programmez!
Le magazine des dévs - CTO & Tech Lead

www.programmez.com

SPÉCIAL ÉTÉ 2025
HORS-SÉRIE #19

GREEN-IT
ECO-CONCEPTION

Langages, infrastructures & serveurs, sites web : des développements plus économes sont possibles

En partenariat avec

Printed in EU - Imprimé en UE - BELGIQUE 7,50 € P928345 - Canada 10,55 $ CAN - SUISSE 14,10 FS - DOM Surf 8,10 € - TOM 1100 XPF - MAROC 59 DH

M 01642 - 19 - F: 6,99 € - RD

PORTAGE

Fiparco Portage

Portage salarial

Votre carrière est une priorité !

Votre contact

Ikram 07 86 45 01 75

fiparco-portage.fr

FIPARCO Sponsor de PROGRAMMEZ !

SOMMAIRE#HS19

Divers

Programmez! est une publication bimestrielle de Nefer-IT. Adresse : 150, rue Lamark, 75018 Paris – France. Pour nous contacter : redaction@programmez.com

Eco-conception, obésité logicielle, utilisation des ressources matérielles… même combat

Petite digression

« *Une grande partie du monde, plus grande que beaucoup ne l'imaginent, pourrait fonctionner sur du matériel obsolète si l'optimisation logicielle était véritablement une priorité* », commentait John Carmack sur X. Tout est parti d'un tweet de LaurieWired : "Et si l'humanité oubliait de fabriquer des processeurs ? Imaginez le Jour Z (Zero Tape-out Day), le moment où plus aucun circuit imprimé ne sera fabriqué. Le design des coeurs les plus avancés serait très mauvais (avec des défauts)"

Une absence de progrès des CPU entraînerait une optimisation des codes et des logiciels sur le matériel courant. Bref, au lieu de se dire, "bah, prenez un PC plus rapide et plus de mémoire, ou provisionnez plus de ressources cloud", on pourrait réellement optimiser les codes, réduire les binaires, se poser les vraies questions sur l'architecture de son application, etc.

Une solution radicale est émise : reconstruire toutes les applications basées sur de l'interprété pour faire du code natif monolithique ! Abonner les patterns actuels pour revenir aux approches plus efficaces des premiers ordinateurs même si l'impact serait important : « *L'innovation des nouveaux produits serait beaucoup plus rare sans les ordinateurs actuels et la montée en puissance régulière.* »

Voilà pour la réflexion et quand Carmack parle, on écoute.

Revenons aux thèmes de ce numéro

Ce n'est pas en mettant l'opprobre sur la technologie, les apps, le cloud, l'IA, les smartphones que nous allons réellement avancer. Les développeurs ont un rôle actif à jouer dans l'éco-conception des sites web, des apps. Les pistes d'optimisation sont nombreuses et très efficaces.

Pourquoi avoir 4 ou 5 pages pour valider un panier de e-commerce (on ne rigole pas, cela existe vraiment) ? Pourquoi multiplier les animations et transitions qui n'apportent rien à l'expérience d'utilisation ? Pourquoi utiliser un CMS complet alors qu'une approche statique peut suffire ? Pourquoi ne pas nettoyer sérieusement les codes, les dépendances et tous les assets techniques & médias de notre projet ?

Il y a qu'à voir le poids moyen d'une page web en 2025 par rapport à 2015, 2005 et même 1995 : il a été multiplié par 180 !

Pourquoi laisser des workloads et des services cloud actifs quand ils ne sont pas utilisés ? Il est possible d'économiser de l'énergie et des milliers d'€.

Encore faut-il laisser le temps aux équipes de bien faire les choses, de sensibiliser aux bonnes pratiques. Modifier un site ou une app déjà en production, c'est toujours compliqué et exige du temps. En revanche, adopter l'éco-conception et/ou la sobriété technologique (incluant tous les aspects : ressources, UX, codes, etc.) dans les nouveaux projets serait déjà un progès.

Le sujet de l'éco-conception n'est pas nouveau à Programmez!. Nos premiers dossiers sur le sujet remontent à plus de 10 ans. Des progrès ont été faits, mais nous sommes seulement au pied de la montagne.

François Tonic

Depuis 24 ans à Programmez!

PROCHAINS NUMÉROS

PROGRAMMEZ! N°270

Disponible à partir du **4 juillet 2025**

PROGRAMMEZ! HORS-SÉRIE 20

Disponible à partir du **26 septembre 2025**

programmez!

Directives de compilation

Programmez! hors-série n°19
ETE 2025

Directeur de la rédaction : **Jean-Christophe Tic**
Rédacteur en chef : **François Tonic**
ftonic@programmez.com

Contacter la rédaction
redaction@programmez.com

Les experts techniques du numéro
Yannick Corre
Raphaël Lemaire
Alexandre Poichet
Guillaume Chervet
Matthieu Vincent
Sylvain Gougouzian
Yoan de Macedo
Nicolas Leseignoux
Hervé Boisgontier
Benjamin Morali
Jérémy Pastouret
Davide Usai
Jérôme Cilly
Kévin Ansard
David de Carvalho

Maquette
Pierre Sandré

Marketing – promotion des ventes
Agence BOCONSEIL - Analyse Media Etude
Directeur : Otto BORSCHA
oborscha@boconseilame.fr
Responsable titre : Anthony Carrée
Téléphone : 09 67 32 09 34

Publicité
Nefer-IT
Tél. : 09 86 73 61 08
ftonic@programmez.com

Impression
Léonce Deprez, France

Dépôt légal
A parution

Commission paritaire
1225K78366

ISSN
2279-5001

Abonnement
Abonnement (tarifs France) : 55 € pour 1 an, 90 € pour 2 ans. Etudiants : 45 €. Europe et Suisse : 65 €
Algérie, Maroc, Tunisie : 64 € - Canada : 80 €
Tom - Dom : voir www.programmez.com.
Autres pays : consultez les tarifs sur www.programmez.com.
Pour toute question sur l'abonnement :
abonnements@programmez.com

Abonnement PDF
monde entier : 45 € pour 1 an.
Accès aux archives : 25 € (1 an)
30 € (2 ans).

Nefer-IT
150, rue Lamarck, 75018 Paris
redaction@programmez.com
Tél. : 09 86 73 61 08

Abonnez-vous à programmez!

Formules	1 an	2 ans	Etudiant	Numérique
Vous recevez le magazine chez vous	1 an 10 numéros (papier) 55 €	2 ans 20 numéros (papier) 90 €	1 an 10 numéros (papier) 45 €	1 an 10 numéros (format PDF) 45 €
Abonnements + accès aux archives	80 € (1 an + archives)	120 € (2 ans + archives)		70 € (1 an + archives)

Tarifs France métropolitaine.
Tarif PDF : valable partout dans le monde

L'abonnement comprend : les numéros normaux et les hors-séries

L'abonnement **numérique** est disponible uniquement sur notre boutique : **www.programmez.com**

Oui, je m'abonne

ABONNEMENT à retourner avec votre règlement à :
PROGRAMMEZ, Service Abonnements
57 Rue de Gisors, 95300 Pontoise

☐ Abonnement 1 an :	55 €
☐ Abonnement 2 ans :	90 €
☐ Abonnement 1 an Etudiant :	45 €
Photocopie de la carte d'étudiant à joindre	

Option : accès aux archives

☐ Pour abonnement 1 an	25 €
☐ Pour abonnement 2 ans	30 €

☐ Mme ☐ M. Entreprise : Fonction :

Prénom : Nom :

Adresse :

Code postal : Ville :

Adresse email indispensable pour la gestion de votre abonnement

E-mail : @

☐ Je joins mon règlement par chèque à l'ordre de Programmez !

☐ Je souhaite régler à réception de facture

* Tarifs France métropolitaine

Offres pouvant s'arrêter à tout moment, sans préavis **PROG HS19**

Yannick Corre
Responsable département Solutions & Numérique Responsable

L' IT responsable, un catalyseur de transformation, les ESN au cœur des pratiques

Les ESN font aujourd'hui face à un double défi : elles doivent répondre à la demande croissante de leurs clients en technologies avancées — IA, cloud, automatisation… — tout en réduisant leur impact environnemental. Mais ces technologies innovantes sont particulièrement énergivores et contribuent fortement à l'augmentation de l'empreinte carbone.

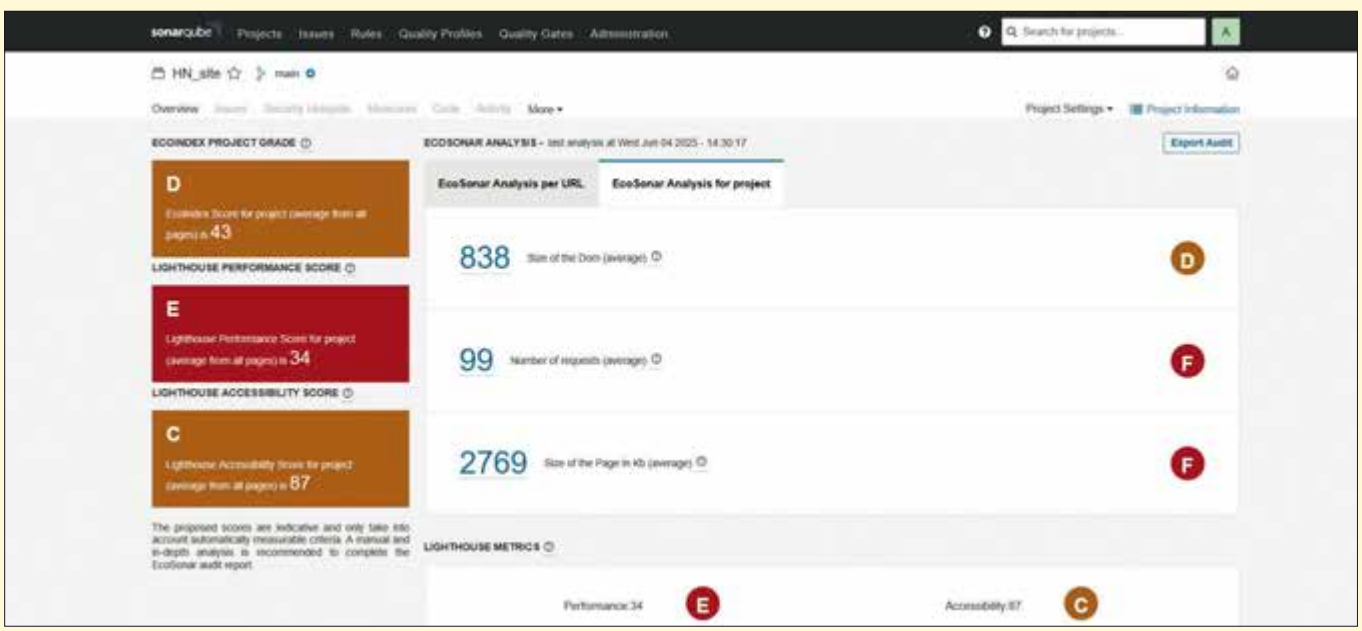
Pour concilier performance technologique et sobriété énergétique, les ESN doivent adopter une approche globale, repensant en profondeur la conception, le développement et la livraison de leurs services. Cela implique l'optimisation des infrastructures, la mise en œuvre de pratiques de développement plus vertueuses et la recherche de stratégies opérationnelles à faible consommation d'énergie. Les solutions expérimentées sont variées : recours à des datacenters alimentés par des énergies renouvelables, optimisation des systèmes de refroidissement et de gestion des ressources serveur, développement de logiciels éco-conçus, ou encore allongement du cycle de vie du matériel via l'utilisation d'équipements réparables, la mise en place de programmes de recyclage et l'établissement de partenariats responsables.

Dans un contexte réglementaire européen de plus en plus strict, les ESN doivent également anticiper les textes qui impacteront directement leur activité. La directive CSRD impose par exemple aux grandes entreprises de déclarer leurs émissions, y compris celles liées à l'IT. Le règlement sur l'écoconception fixe quant à lui des normes d'efficacité énergétique pour les équipements numériques, tandis que les marchés publics intègrent désormais des critères environnementaux renforcés. Enfin, la directive sur le devoir de vigilance (CSDDD), attendue pour 2025, exigera des grandes entreprises qu'elles identifient, préviennent et atténuent les impacts négatifs sur les droits humains et l'environnement tout au long de leur chaîne de valeur. Dans ce cadre, les ESN devront évaluer leurs partenaires et sous-traitants sous l'angle de la durabilité et ajuster leurs pratiques en conséquence.

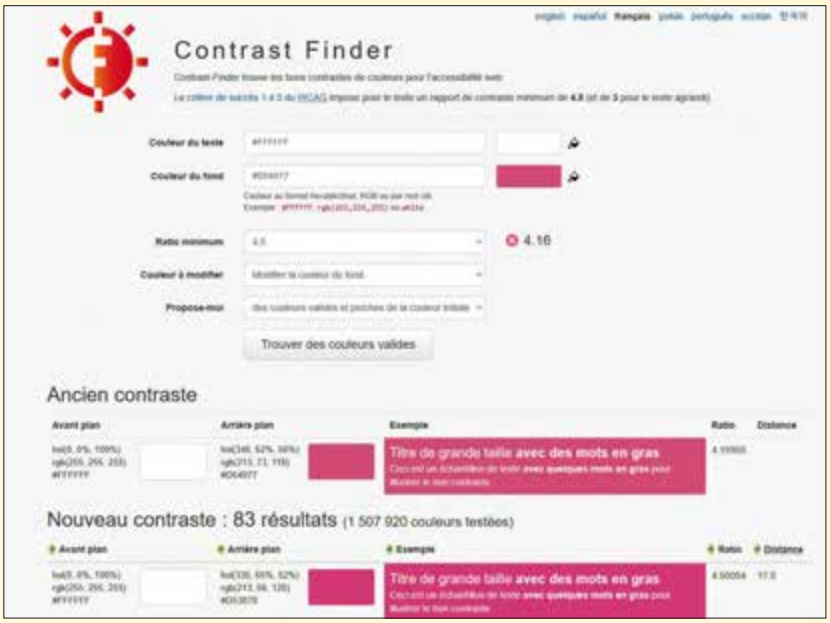
HN Services mène depuis plusieurs années une transformation RSE fondée sur trois piliers : la réduction de son empreinte environnementale, la promotion d'un numérique responsable auprès de ses clients, et la création d'un environnement de travail inclusif et porteur de sens pour ses collaborateurs. Précurseur dans l'obtention du label Ecovadis et membre actif de l'Alliance Green IT (AGIT), l'entreprise affirme ainsi son engagement historique en faveur d'un modèle plus durable, performant et éthique.

Allonger le cycle de vie du matériel

HN Services vise à prolonger la vie des équipements, éliminer les consommations superflues, et sensibiliser chaque collaborateur à une logique d'efficacité fonctionnelle. Cette démarche collaborative a d'ailleurs valu à l'entreprise l'obtention du label Numérique Responsable en juillet 2023. HN Services applique une gestion proactive des équipements. Plutôt que de procéder au renouvellement systématique, l'entreprise a mis en place une politique de maintenance préventive et de reconditionne-



ment. Un ordinateur ne répondant plus aux besoins d'un développeur peut parfaitement satisfaire ceux d'un administratif. Cette adéquation entre besoins réels et ressources techniques permet d'éviter la surconsommation et le gaspillage d'équipements encore fonctionnels. HN Services a également implémenté plusieurs mesures de rationalisation : extinction automatique des machines inactives, réduction des environnements redondants et limitation des équipements non essentiels. Le dimensionnement approprié constitue un autre axe majeur : pourquoi déployer un ordinateur ultrapuissant pour de simples tâches bureautiques ? Un mini-PC peut souvent répondre aux besoins tout en réduisant la consommation énergétique. Cette logique du "juste nécessaire" s'applique à tous les niveaux, y compris dans les infrastructures serveur, HN Services favorise la virtualisation et la mutualisation pour limiter la multiplication des serveurs physiques et optimiser les performances tout en réduisant l'empreinte carbone.



Le choix des fournisseurs représente aussi un levier stratégique. HN Services privilégie systématiquement les fabricants engagés dans des démarches environnementales certifiées ainsi que des data centers sélectionnés pour leur efficacité énergétique, leur approvisionnement en énergies renouvelables et leur gestion responsable des flux thermiques. Ce parti pris permet de partager la responsabilité de la sobriété numérique avec l'ensemble de la chaîne de valeur. HN Services s'inscrit dans une économie circulaire où chaque décision vise à maximiser l'utilité, éviter le gaspillage et prolonger la durée de vie des équipements. En agissant sur l'ensemble du cycle, de l'achat au recyclage, en passant par l'usage, la maintenance et le reconditionnement, l'entreprise démontre qu'il est parfaitement possible de concilier performance opérationnelle et responsabilité environnementale.

Vers un logiciel durable

HN Services optimise la consommation d'énergie de ses architectures pour créer des actifs écoresponsables de manière intelligente, en les réutilisant et en optimisant leur efficacité énergétique.

La virtualisation est un levier central pour améliorer l'utilisation des ressources sans changer radicalement les installations existantes. HN Services réduit le nombre de serveurs physiques nécessaires au support des applications en déployant des environnements virtuels. Dans les environnements de développement, les machines virtuelles peuvent être provisionnées de manière temporaire, uniquement pendant les phases de test, afin d'éviter une consommation énergétique inutile. Des audits réguliers identifient les ressources inactives ou sous-exploitées qui sont ensuite désactivées ou réaffectées. La mutualisation vise aussi à optimiser le taux d'utilisation des équipements IT. Elle repose sur le partage d'infrastructures (plateformes de sauvegarde, serveurs de développement…) entre plusieurs équipes, projets ou clients. L'usage de la conteneurisation (Docker, Kubernetes) permet d'aller encore plus loin en mutualisant les ressources au niveau des processus. Ces technologies facilitent l'automatisation, réduisent les duplications et favorisent une gestion dynamique des charges, avec à la clé une diminution sensible de l'empreinte carbone. Une telle mutualisation des ressources peut entraîner une réduction de l'espace nécessaire pour le stockage des équipements et, par conséquent, moins d'énergie dépensée pour le refroidissement et l'alimentation des serveurs.

Le choix des partenaires d'hébergement a un impact direct sur la performance environnementale des services numériques. Privilégier des data centers à faible PUE (Power Usage Effectiveness), alimentés en énergies renouvelables et engagés dans des démarches de certification (ISO 14001, ISO 50001), permet de réduire significativement les émissions associées au stockage et au traitement des données.

L'obésiciel, ou *software bloat*, la dérive logicielle qui conduit les applications à consommer toujours plus de ressources, souvent de manière injustifiée, se traduit par une utilisation excessive du CPU, une occupation mémoire surdimensionnée, des échanges réseau non optimisés et un recours intensif au stockage. Dans une démarche d'IT responsable, il est essentiel de s'attaquer aux causes profondes de la consommation excessive de ressources numériques. Le code source des applications constitue l'un des principaux leviers d'optimisation. Un code mal conçu peut entraîner des traitements trop longs ou inutiles, solliciter excessivement le processeur ou le disque dur, ou encore provoquer des fuites de mémoire en retenant inutilement des données. C'est pourquoi HN Services intervient dès la racine du problème en réalisant des audits techniques du code source. Ces analyses permettent d'identifier les boucles mal calibrées, les accès disque inefficaces, les objets mal gérés en mémoire, ou encore les logiques d'exécution trop complexes. L'enjeu est d'optimiser ces éléments pour rendre le code plus sobre, plus rapide et moins énergivore.

L'optimisation algorithmique a un impact direct sur le nombre d'opérations que l'ordinateur doit effectuer et donc sur la quantité d'énergie consommée. L'idée est de réduire cette "complexité computationnelle" et le travail nécessaire pour obtenir un résultat sans altérer la performance fonctionnelle, c'est-à-dire la qualité ou la rapidité du service rendu à l'utilisateur. L'allègement du socle applicatif est également stratégique. En réduisant la taille des dépendances, en élimi-

nant les bibliothèques inutilisées et en mettant en œuvre le chargement à la demande, on diminue significativement l'empreinte mémoire et les temps de chargement. Cela permet également d'améliorer l'expérience utilisateur tout en réduisant les besoins en ressources matérielles.

Dans le cadre d'un pipeline DevOps, HN Services peut intégrer des tests de performance énergétique en amont, au même titre que les tests fonctionnels ou de sécurité. Ces tests évaluent l'impact des évolutions logicielles sur la consommation CPU/mémoire et bloquent les déploiements si des seuils critiques sont franchis. Une fois en production, le monitoring applicatif permet de mesurer en continu les performances et les consommations. Couplée à une capacité d'auto-scaling intelligent, cette surveillance autorise une allocation dynamique et efficiente des ressources, limitant ainsi le gaspillage énergétique. En traitant l'obésiciel comme un défaut technique au même titre qu'un bug ou une faille de sécurité, HN Services peut concevoir des logiciels plus propres et performants. Les décisions architecturales, telles que la conception efficace de bases de données, moins d'appels système réseau, ou des systèmes de mise en cache, peuvent alléger les charges de serveurs et de réseaux réduisant la consommation énergétique.

Mesurer l'impact des applications : recherche d'outils dédiés

Dans une approche IT pragmatique orientée résultats, la mesure de l'impact environnemental des applications est un axe stratégique. Il s'agit ici de développer des indicateurs au niveau client afin de garantir des comportements mesurables et comparables dans le temps. HN Services évalue des outils dédiés à l'analyse de la consommation énergétique et des performances environnementales à travers l'utilisation, les flux de données, l'architecture technique et le contexte dans lequel les applications sont exécutées. L'idée est de permettre une lecture rapide et standardisée de cette consommation.

Chez HN Services, l'optimisation de l'empreinte environnementale des applications ne repose pas uniquement sur le code en lui-même. En pratique, les équipes s'appuient sur de nombreuses données d'usage et de fonctionnement collectées depuis différentes sources : par exemple, les statistiques d'activité sur des plateformes comme GitHub (fréquence des commits, ouverture de tickets…), les logs des serveurs, les métriques de charge (comme l'utilisation CPU, RAM, bande passante…), ou encore le nombre et la fréquence des appels API. Toutes ces informations permettent d'identifier les éléments énergivores d'une application, même si le code semble "propre". Cela peut révéler des points de surconsommation comme un processus trop lourd, une mauvaise gestion des appels réseau, ou encore une infrastructure cloud mal dimensionnée ou mal localisée. Une fois ces données récoltées, elles sont analysées à la lumière de référentiels comme le RGESN (Référentiel Général d'Écoconception de Services Numériques), qui donne des repères objectifs pour évaluer l'impact environnemental d'un service numérique. Cela permet ensuite de définir des actions prioritaires : faut-il migrer vers un hébergement plus vert ? Réécrire une partie du code trop complexe ? Même dans des contextes techniques contraints, cette approche permet de poser un diagnostic fiable et d'orienter des efforts concrets vers plus de sobriété numérique.

L'objectif n'est pas d'imposer des refontes totales, mais bien d'accompagner la prise de décision. Cette transparence est d'autant plus utile dans des contextes où les applications évoluent sur plusieurs années et où la pile technologique s'accroît tout en étant souvent mal documentée. Grâce à ces outils, HN Services aide ses clients à mieux comprendre ce que "coûte" réellement une application, non seulement en ressources financières ou en bande passante, mais aussi en empreinte carbone, en durée de vie des infrastructures et en mobilisation de ressources partagées.

Éco-conception : nouveaux projets vs existants

L'éco-conception est un enjeu majeur pour HN Services, qui cherche à intégrer des pratiques responsables dès la conception de nouvelles solutions tout en optimisant les projets existants en tenant compte des contraintes liées à l'infrastructure et au code source legacy. L'éco-conception est un processus continu qui implique une réévaluation constante des projets en fonction de l'évolution des technologies et des besoins. Cela passe par plusieurs choix techniques : adopter des pratiques de développement optimisé, concevoir des applications et bases de données plus compactes, et utiliser des services cloud moins gourmands en énergie. L'objectif est de limiter les besoins en calcul, en stockage ou en bande passante dès la conception, plutôt que de corriger après coup. La simplification et l'optimisation du code vont le rendre plus maintenable et modifiable, ce qui va optimiser les coûts de projet et de maintenance. L'éco-conception est une version 2.0 des bonnes pratiques de codes.

HN Services mise sur des technologies plus sobres comme les micro-services. Cette approche, qui consiste à découper une application en petits modules autonomes, permet de ne mobiliser que les ressources strictement nécessaires à un instant donné. Résultat : moins d'énergie consommée et une infrastructure plus agile, capable de s'adapter facilement aux besoins sans gaspillage.

Les optimisations des projets existants peuvent prendre diverses formes. Elles incluent, par exemple, la réorganisation des processus métiers pour améliorer l'efficacité et réduire les gaspillages de ressources. L'objectif est de rationaliser les tâches et d'optimiser les flux de travail pour réduire la consommation d'énergie et les coûts opérationnels. La mise en place de mécanismes de suivi et de contrôle pour identifier rapidement les processus inefficaces ou énergivores permet de mettre en œuvre des actions correctives dans le cadre des limitations des systèmes hérités.
L'adaptation des interfaces utilisateurs (UX) représente également un axe important dans l'éco-conception des projets existants. En améliorant l'ergonomie et en simplifiant les interactions avec les systèmes, HN Services permet de réduire la consommation des utilisateurs finaux et d'optimiser les res-

sources nécessaires pour accéder aux services. Une interface plus intuitive permet non seulement d'améliorer l'expérience utilisateur, mais aussi de favoriser une utilisation plus efficace des systèmes, réduisant ainsi les besoins en énergie pour des actions complexes ou répétitives. Il convient de noter que parler de numérique responsable ne signifie pas seulement réduire l'empreinte environnementale des services numériques au travers d'une IT plus responsable ou de l'éco-conception. Cette démarche englobe également des enjeux sociaux et éthiques, au premier rang desquels figure l'accessibilité numérique. Trop souvent reléguée au second plan, l'accessibilité devient pourtant incontournable avec la mise en application, le 28 juin 2025 prochain de la directive européenne sur l'accessibilité des services numériques, encadrée en France par l'ARCOM. Elle impose aux acteurs concernés (e-commerce, banques, opérateurs télécoms, services audiovisuels, plateformes numériques, etc.) de garantir un accès équitable à leurs services, y compris pour les personnes en situation de handicap. Dans ce contexte, intégrer l'accessibilité dès la conception des interfaces n'est plus une option, mais une exigence, au même titre que la sobriété fonctionnelle. Un numérique véritablement responsable doit aussi être un numérique inclusif.

Autre partie d'une démarche IT responsable, le refactoring joue un rôle clé pour améliorer la performance et la sobriété du code. En éliminant la redondance et en factorisant les fonctions ou modules récurrents, les équipes de développement réduisent la taille et la complexité des applications, ce qui limite les traitements inutiles et optimise la consommation de ressources. Cette bonne pratique améliore non seulement la lisibilité et la maintenabilité du code, mais contribue aussi à prolonger la durée de vie des logiciels et à réduire leur empreinte énergétique. Chez HN Services, nous intégrons systématiquement cette logique dans nos processus de développement pour allier efficacité, durabilité et qualité logicielle.
Lorsque la modification du code source est limitée, notamment en raison de l'architecture des systèmes existants, HN Services offre des conseils d'architecture permettant à ses clients d'optimiser la gestion des ressources et des données. Cette optimisation passe par la mise en place de solutions efficaces pour le stockage, le traitement et le transfert des données. L'ESN recommande, par exemple, l'utilisation de bases de données plus performantes et éco-responsables, ainsi que l'adoption de systèmes de gestion de la mémoire qui réduisent la charge sur les serveurs. La rationalisation des systèmes de stockage est également un point clé de l'approche d'éco-conception. L'ESN conseille ses clients sur l'optimisation des systèmes de gestion de données, en favorisant des solutions de stockage à faible consommation énergétique et en minimisant la duplication des données. HN Services guide également ses clients dans le choix d'architectures permettant de limiter le gaspillage des ressources. Cela inclut la réduction de l'utilisation des ressources informatiques non nécessaires, et la mise en place de mécanismes de gestion automatique des ressources qui permettent d'ajuster la consommation énergétique en fonction des besoins réels.

Le rôle des développeurs dans l'éco-conception

Chez HN Services, les développeurs occupent une place centrale dans la stratégie d'éco-conception, acteurs clés de la transformation des pratiques de développement pour répondre aux exigences d'une informatique plus responsable. Cette responsabilité est d'autant plus importante que, dans de nombreux cas, le code source des applications n'est pas directement accessible ou modifiable, notamment lorsqu'il s'agit de projets hérités de type mainframe ou de solutions éditeurs. HN Services privilégie ici une approche proactive visant à renforcer les compétences des développeurs et à les outiller pour qu'ils puissent agir à chaque niveau du cycle de vie logiciel, même dans des contextes contraints.
La formation continue constitue une première brique essentielle de cette stratégie. Les équipes techniques sont régulièrement sensibilisées aux enjeux du numérique responsable, à la sobriété numérique et aux pratiques d'optimisation énergétique. Ces formations permettent aux développeurs de mieux comprendre l'impact environnemental des lignes de code qu'ils produisent, mais aussi de développer une véritable culture de l'efficience, intégrée dès les premières étapes

de réflexion sur un projet. L'objectif est d'en faire des ambassadeurs de l'éco-conception, capables de relayer les bonnes pratiques auprès des clients et au sein des équipes projet. D'ailleurs à l'échelle plus globale, HN met en place des « Fresques du Numérique et du Climat », visant à sensibiliser et à engager les équipes dans une démarche collective en faveur d'un avenir numérique plus respectueux de l'environnement. Depuis 2023, 115 collaborateurs ont été formés.

Pour structurer cette démarche, HN Services s'appuie sur des règles spécifiques issues du référentiel du numérique responsable (notamment RGESN, Référentiel général d'écoconception des services numériques) et des outils d'analyse du code source comme SonarQube. Cet outil permet de détecter automatiquement les mauvaises pratiques de codage, les redondances inutiles, les appels systèmes énergivores ou les surconsommations de ressources mémoire. Le refactoring du code est encouragé dès que cela est possible, afin de supprimer les duplications, optimiser les algorithmes, alléger les traitements et limiter le recours aux bibliothèques lourdes ou obsolètes. Cette vigilance constante permet de garantir que chaque mise à jour de code contribue à une amélioration de la performance énergétique de l'application.

Même lorsque le champ d'action sur le code est restreint, les développeurs peuvent travailler avec les architectes sur des solutions alternatives, en proposant par exemple des patterns d'optimisation. Cela permet de réduire la consommation globale de l'application sans nécessairement modifier son cœur fonctionnel. Dans les nouveaux projets, les développeurs adoptent des choix technologiques orientés sobriété, en sélectionnant des langages et frameworks plus légers, en limitant les appels API, en adaptant la fréquence des mises à jour des données ou encore en réduisant les temps d'exécution.

Par ailleurs, HN Services encourage une approche DevSecOps renforcée par des dimensions « green », souvent désignées sous le terme DevGreenOps. L'idée est d'intégrer les préoccupations environnementales dès les phases de conception et de développement, tout comme la sécurité est prise en compte dès les premières lignes de code. Cette approche permet une meilleure anticipation des impacts énergétiques futurs du produit, et encourage l'utilisation d'environnements de test plus sobres, de pipelines CI/CD optimisés, et de méthodes de déploiement qui évitent la surconsommation de ressources.

En interne, HN Services développe également des applications pour analyser et objectiver l'impact des logiciels en production. Ces solutions permettent d'évaluer, de manière concrète, ce que « coûte » une application en termes de consommation énergétique ou de mobilisation des ressources. Ces outils permettent de donner une note environnementale aux applications en croisant des données d'usage réelles (comme celles issues de GitHub ou des environnements de production) avec des métriques de consommation énergétique. L'objectif est de donner à la fois une visibilité aux équipes internes et aux clients, et de créer un référentiel d'amélioration continue.

Enfin, cette expertise technique est partagée avec les clients : HN Services ne se limite pas à appliquer ces pratiques pour ses propres développements, mais les promeut activement auprès de ses clients et partenaires. L'entreprise conseille ses clients sur la mise en œuvre de processus de développement plus sobres, les accompagne dans la refonte de leurs workflows, et les aide à adopter des pratiques d'architecture orientées sobriété. Elle les forme également à identifier les bons moments pour intervenir : pas besoin de tout réécrire, mais plutôt de profiter des cycles naturels de maintenance, d'évolutions fonctionnelles ou de migrations techniques pour introduire des optimisations ciblées.
Ainsi, les développeurs chez HN Services ne sont pas de simples exécutants techniques. Ils sont les vecteurs d'un changement structurel dans les modes de production numérique. Grâce à leur formation, leurs outils, et une culture partagée de la sobriété numérique, ils participent pleinement à la réduction de l'empreinte environnementale des systèmes d'information. En intégrant l'éco-conception dans leur quotidien, ils contribuent à faire de chaque ligne de code une opportunité de développement durable.

HN Factory : levier d'un IT collectif et responsable

Au cœur de la stratégie écoresponsable de HN Services se trouve HN Factory, son centre de services mutualisé apportant des solutions adaptées (développement, TMA, RPA, projet de migration, renfort technique...) et qui joue un rôle central dans la mise en œuvre d'une démarche collective, cohérente et mesurable en matière de numérique responsable.

HN Factory, qui regroupe des équipes pluridisciplinaires mutualisées, permet à HN Services d'optimiser les ressources humaines, techniques et logicielles dans une logique de rationalisation et de réduction d'impact. En centralisant certaines activités de développement, de test et de maintenance, HN Services évite la démultiplication des environnements de travail souvent énergivores, favorise le partage de composants logiciels réutilisables et réduit les redondances inutiles dans les projets. Par sa nature même, HN Factory encourage une logique d'efficacité fonctionnelle : allouer les bons outils, au bon moment, aux bonnes personnes, selon un usage raisonné. Cela se traduit par une meilleure gestion des postes informatiques, une répartition des équipements en fonction des usages réels et la mise en place de bundles IT adaptés à chaque métier, évitant ainsi la surconsommation matérielle et logicielle.

Cette centralisation permet aussi d'appliquer plus efficacement les principes de sobriété numérique dans les projets clients. À travers des référentiels partagés, des outils de mesure communs et une gouvernance technique unifiée, HN Factory intègre de manière systématique des critères environnementaux dans l'architecture logicielle, l'organisation des flux de données, ou encore la fréquence de déploiement et de mise à jour des applications. Les choix techniques s'appuient sur des indicateurs concrets, notamment ceux issus du RGESN qui fournit un cadre clair pour évaluer et améliorer l'impact environnemental des services numériques tout au long de leur cycle de vie.

Les impacts des IA génératives et comment écoconcevoir avec ces outils

Raphaël Lemaire
Responsable offres Green IT chez Zenika, consultant et formateur, spécialisé dans le numérique responsable.

L'intelligence artificielle (IA) générative est à la mode. Tout le monde en met partout, il y a des boutons IA dans toutes les applications, et votre patron en demande dans votre projet. Pour être juste, ces outils peuvent être utiles et des cas d'usages pertinents commencent à apparaître. En parallèle, les entreprises ont des objectifs de réduction de leurs impacts environnementaux. Et nous sommes tous sensibles au sujet des crises écologiques dont nous voyons les effets au jour le jour dans l'actualité. L'IA générative a la réputation d'avoir des impacts environnementaux importants. Qu'en est-il vraiment ? Et comment l'utiliser au mieux ?

IA générative ?

Rappelons d'abord que le terme « intelligence artificielle » est utilisé au moins depuis les années soixante pour désigner différentes technologies : des simples algorithmes minimax ou A* des jeux à la reconnaissance d'images, en passant par la logique formelle, la traduction de texte ou les moteurs de recommandations.

En fait, le terme d'« Intelligence artificielle » est essentiellement une appellation marketing autour de programmes performants pour leur époque. Une fois ceux-ci devenus banals, on cesse de les qualifier d'intelligents.

L'engouement autour de l'IA depuis la sortie de ChatGPT tourne autour de l'IA générative, des outils capables de générer du contenu (texte, images, vidéo) à partir d'un prompt, en général textuel. C'est sur ce type de programmes que nous allons nous focaliser dans cet article.

Vocabulaire

LLM : *Large Langage Model*, un type de programme basé sur des réseaux de neurones et entraîné avec une grande quantité de contenu textuel.

GPT : Pour *Generative Pretrained Transformer*. *Transfomer* est un type d'architecture utilisant des réseaux de neurones de façon complexe (en série, de façon répétée, etc.), *pretrained*, signifie simplement pré entraîné, et générative indiquant que le modèle peut générer du contenu.

Prompt : instruction textuelle envoyée au LLM pour générer du contenu

Token : Les LLM ne travaillent pas directement avec les mots, mais avec des *tokens*, qui peuvent être plus petits. Par exemple « microprocesseur » peut être découpé en deux *tokens* : « micro » et « processeur ».

Prompt engineering : Le terme « *prompt engineering* » regroupe des stratégies d'écriture de prompt permettant d'obtenir de meilleurs résultats du modèle.

Entraînement : ajustement des paramètres d'un programme à partir d'un jeu de données, par exemple afin qu'il puisse bien catégoriser d'autres données plus tard, ou générer des contenus similaires.

Paramètres : dans le cas des IA génératives, quand il est question parle du nombre de paramètres (généralement exprimé en milliards, par exemple « 8B » comme « 8 billions parameters »), il s'agit du nombre de paramètres des réseaux de neurones associés, c'est à dire les poids liés aux connexions entres les neurones virtuels. Cette métrique donne une idée de la taille du modèle utilisé.

Quels sont vraiment les impacts de ces outils ?

Peut-être avez-vous entendu diverses informations sur les impacts environnementaux de l'IA. Comme souvent, des articles sortent quasi quotidiennement, avec des chiffres rarement mis en perspective, et il n'est pas facile d'avoir une bonne vision sur le sujet.

La trajectoire que nous devrions suivre

Nous sommes dans une situation très grave au niveau des impacts environnementaux de l'humanité. La plupart des limites planétaires(1), que nous devrions respecter pour ne pas compromettre les conditions dans lesquelles notre civilisation a pu se développer, sont dépassées. *Nous devons faire décroître fortement nos impacts*. Or ceux-ci sont en croissance.

Si on se focalise sur le numérique, la Stratégie Nationale Bas Carbone en France vise -45% d'émissions de gaz à effets de serre d'ici 2030(2), alors que le scénario tendanciel de l'ADEME (établi sans compter avec l'IA générative) prévoit

(1) https://fr.wikipedia.org/wiki/Limites_plan%C3%A9taires

(2) https://commission.europa.eu/document/download/ab4e488b-2ae9-477f-b509-bbc194154a30_fr

une croissance de 45%(3). Clairement ces trajectoires ne sont pas en phase.

Des signaux globaux très inquiétants

Les annonces autour des impacts environnementaux de l'IA générative sont effectivement très inquiétantes.

Les équipements, comme les GPU et CPU de Nvidia augmentent en puissance (je parle ici électrique) : la GB200 est à 1200 Watts et la Rubin à 1800 Watts(4). À titre de comparaison, la puissance d'un appareil à raclette moyen est de 850 Watts(5). **Figure 1**

Ces équipements sont rassemblés dans des baies comme les NVL 72 et NVL 576, toujours chez Nvidia, qui consomment respectivement 120 kW et 600 kW, soit autant que plusieurs dizaines de foyers français. Pour une seule baie.

Pour supporter ces équipements, les data centers ont besoin de plus d'énergie et de plus d'eau que précédemment. Dans certains cas, il est même nécessaire de démolir et de reconstruire le bâtiment pour adapter la climatisation à la puissance des serveurs.

Des projets pharaoniques se multiplient. En France par exemple, il est question de dépenser beaucoup d'argent pour construire des data centers dont la puissance pourrait atteindre 1GW, soit plus que la production d'un réacteur nucléaire typique, qui est de 800 MW. Et ce projet est encore très petit par rapport aux annonces américaines. Les besoins de ces structures impliquent la prolongation de la vie de centrales à charbon(6), voire des investissements dans des centrales au gaz(7).

Ce constat amène divers acteurs à faire des projections très alarmistes sur la consommation d'énergie et de matières premières des centres informatiques et leurs impacts environnementaux dans le futur, qui pourraient aller jusqu'à tripler, voire quadrupler.

Ces impacts accrus sont à mettre en perspective avec le service rendu. Les IA génératives s'ajoutent à tous les autres services déjà rendus par le numérique : messageries, réseaux sociaux, e-commerce, paiement, banque, assurance, musique, vidéo… La perspective régulièrement présentée est donc de doubler voire quadrupler les impacts, pour un seul cas d'usage. Cela en vaut-il le coup ?

Une grande variabilité selon les tâches et les modèles

D'un autre côté, si on se place au niveau des programmes et des conversations, on constate une très grande variabilité suivant le modèle et la tâche choisie. Il a été montré que générer une seule image peut consommer jusqu'à la moitié de la charge d'un smartphone(8) et certains exemples issus de benchmarks sont assez affolants avec des consommations qui vont jusqu'à 1785 kWh pour une seule tâche(9).

À l'opposé, Epoch.AI estime de façon optimiste une conversation typique sur ChatGPT(10) à 0.3Wh au lieu de 3 Wh et la startup chinoise DeepSeek a fait beaucoup de bruit en déployant un modèle très efficace.

La startup Hugging Face a réalisé des estimations d'impacts de conversations avec différents modèles(11), montrant les écarts entre les différentes architectures. Il y a ainsi un facteur 250 de différence entre Chat GPT 4 et Mistral ! Grosse limitation cependant : dans le cas des différentes versions de Chat GPT, il s'agit d'estimations, faute de transparence d'Open AI. L'écart semble toutefois cohérent avec l'ampleur des investissements d'Open AI qui a basé sa stratégie sur le gigantisme de ses modèles.

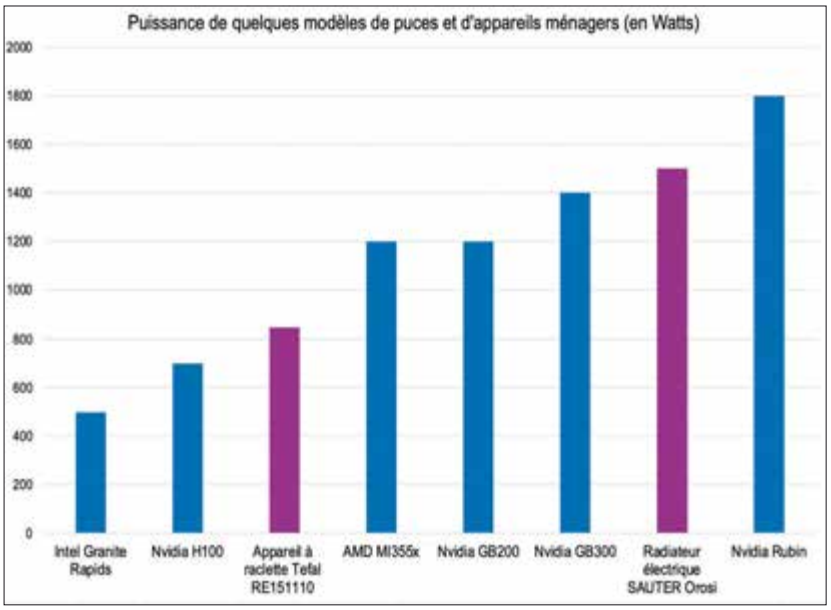
Il est possible de mettre en perspective ces chiffres avec ceux de l'ADEME sur les usages numériques ou sur le transport et l'alimentation, en gardant bien à l'esprit que les périmètres

(3) https://www.arcep.fr/uploads/tx_gspublication/dossier-presse-Etude-Ademe-Arcep-lot3_mars2023.pdf?utm_source=pocket_saves

(4) https://semianalysis.com/2025/02/13/datacenter-anatomy-part-2-cooling-systems/ via Benoit Petit https://www.youtube.com/watch?v=iVtF5F7k2mE

(5) https://www.carrefour.fr/p/appareil-a-raclette-plancha-crepiere-6-personnes-re151110-tefal-3168430359147, et pour un radiateur électrique : https://www.leroymerlin.fr/produits/radiateur-electrique-a-inertie-fluide-1500-w-sauter-orosi-horizontal-blanc-80162919.html

Figure 1



(6) https://intelligence-artificielle.developpez.com/actu/371121/Pour-repondre-aux-besoins-des-centres-de-donnees-d-IA-Trump-exempte-70-centrales-au-charbon-de-la-regle-de-l-ere-Biden-des-exigences-federales-visant-a-reduire-les-emissions-de-produits-chimiques-toxiques/

(7) https://medium.com/@benjamin.davy/sustainability-implications-of-the-ai-revolution-b10ccb97435b

(8) https://arxiv.org/pdf/2311.16863

(9) https://www.linkedin.com/posts/th%C3%A9o-alves-da-costa-09397a82_encore-un-superbe-effet-rebond-de-limpact-activity-7292100769648963584-5C5I?utm_source=share&utm_medium=member_desktop&rcm=ACoAAAJIBccBgSQxC6Zk4kfX9ogZlpHe4t_Hn5U

(10) https://epoch.ai/gradient-updates/how-much-energy-does-chatgpt-use

(11) https://huggingface.co/spaces/genai-impact/ecologits-calculator

et les méthodes de calcul sont très différents(12). **Figure 2**

Perspectives

Nous avons donc des projets pharaoniques et des équipements qui consomment énormément d'un côté, et de l'autre des modèles pas si consommateurs et qui s'optimisent.

Il est possible que l'ambition des investissements soit revue à la baisse parce que les modèles actuels ne réclament plus le gigantisme initialement appliqué et prévu pour avoir de bonnes performances. Cela semble déjà se concrétiser puisque Microsoft et Amazon réduisent leurs plans d'exploitation de datacenters(13) et que des installations dédiées à l'IA en Chine ne sont pas utilisées(14). Mais d'autres signaux vont dans le sens contraire comme les images de la construction de l'énorme Datacenter Stargate d'OpenAI, à Abilene, au Texas(15).

Les impacts des data centers devraient cependant beaucoup augmenter dans les années à venir, même sans tripler ou quadrupler. L'Agence Internationale de l'Énergie table sur un doublement de la consommation d'électricité, passant de 415 TWh en 2024 à 945 TWh en 2030, dont 70% à cause de serveurs dédiés à l'IA. Cette trajectoire n'est pas responsable, nos impacts devant se réduire fortement et non augmenter.

Comment réduire ces impacts ?

Se pose maintenant la question de savoir comment développer nos produits logiciels sans que ceux-ci aient un impact excessif par rapport aux besoins auxquels ils répondent.

L'écoconception de services numériques

L'écoconception (un terme qui n'est pas propre au numérique), est une démarche itérative consistant à optimiser à la baisse les impacts environnementaux d'une solution tout en continuant à répondre au besoin initial. Il est possible d'écoconcevoir un objet (une tasse, un smartphone…), un bâtiment, une action (transporter une personne sur 10 km…), ou encore un service.

L'écoconception de services numériques existe depuis longtemps, et dispose d'une large documentation (référentiels, exemples, tutoriels, livres…).

Les services numériques utilisant des IA génératives sont d'abord des services numériques

Les services numériques utilisant des IA génératives sont d'abord des services numériques. Cela signifie que, globalement, les mêmes concepts, actions et outils peuvent être utilisés.

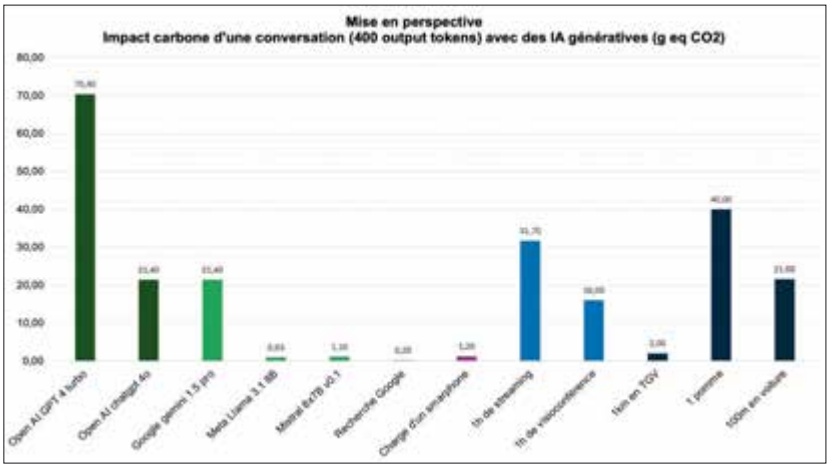


Figure 2 - *Sources : hugging face, étude de Alex de Vries, ADEME*

Dans le cas des services numériques, les actions clés de l'écoconception se déclinent autour de deux axes :

- Utiliser le **moins d'appareils** (serveurs, smartphones, laptops…) possible et les faire durer le **plus longtemps** possible.
- Utiliser **moins de ressources numériques** (réseau, stockage, CPU, RAM…) pour limiter les besoins d'infrastructure.

Concrètement, on va donc :

- Assurer la **compatibilité** avec le parc d'appareils cible et ne pas les pousser à l'obsolescence en les faisant « ramer », ce qui implique une attention importante aux **performances front**.
- Mettre en place dans la conception une démarche de **sobriété fonctionnelle** permettant d'éviter de faire grossir le produit.
- Utiliser une **architecture scalable et modulaire** permettant de dimensionner les composants (VM, conteneurs) au plus juste de limiter leur nombre en suivant la courbe de charge.
- **Limiter la quantité de données stockées** en prêtant une attention particulière au cycle de vie des données.
- **Optimiser les applications** aux divers niveaux possibles : algorithmes, structures de données, requêtes aux bases de données…

À ces points s'ajoute la localisation de l'hébergement de l'application. Les différents pays du monde ne produisant pas leur électricité de la même façon, les impacts associés à la consommation d'énergie des serveurs seront différents entre, par exemple, l'Irlande et la France. Les pays ayant un mix électrique majoritairement basé sur les énergies renouvelables et le nucléaire (comme la France ou la Suède) sont ceux qui ont l'électricité ayant le moins d'impact.

Les outils qui peuvent être utilisés dans le cas d'une démarche d'écoconception sont, bien entendu, tous les outils permettant d'observer le comportement des applicatifs et d'améliorer leurs performances, , ainsi que ceux qui permettent de réaliser des estimations des impacts à partir de critères techniques.

À partir de ces points clés, une multiplicité d'acteurs ont créé des documents listant des critères et pratiques allant dans le

(12) https://impactco2.fr/outils/usagenumerique

(13) https://gizmodo.com/amazon-follows-microsoft-in-retreat-from-ambitious-ai-data-center-plans-2000592217

(14) https://www.technologyreview.com/2025/03/26/1113802/china-ai-data-centers-unused/

(15) https://www.youtube.com/watch?v=fUiI03X6DQc

sens attendu et qui sont également des bonnes pratiques de conception et de développement. Ce sont les référentiels d'écoconception de services numériques. Plusieurs d'entre eux traitent au moins en partie d'IA et seront cités plus loin.

Envisager des alternatives à l'IA [générative]

L'IA générative qui a le moins d'impact est celle qui n'est pas utilisée.

Certes ces outils peuvent répondre à beaucoup de cas d'usage, mais souvent, un programme plus classique et plus simple, ou même un autre algorithme de Machine Learning (arbre de décision, clustering…), répondra au besoin.

Une partie au moins des déploiements de projets comprenant de l'IA générative provient de la peur de rater une opportunité technologique plus que d'un vrai besoin et d'un vrai alignement du cas métier avec ce type de programme, comme le montre une étude d'IBM(16).

De plus, même pour des cas d'usage « intelligents », impliquant du Machine Learning, l'IA générative n'est sans doute pas le meilleur choix. Le Dr Sasha Luccionni et son équipe ont ainsi montré que pour des tâches comme la classification d'image ou de texte, la génération de légende pour des images ou encore la génération de résumés, les IA spécifiques pouvaient consommer jusqu'à 30 fois moins d'énergie(17) que les IA génératives.

Prenons pour illustrer ces points l'exemple d'une entreprise chargée de gérer le transport de personnes avec des trains sur tout un pays et imaginons qu'ils aient réalisé un atelier pour générer des idées d'utilisation d'IA générative dans le cadre de leurs services. Voici un tableau listant des idées possibles :

Cas d'usage	Solution : avec ou sans IA générative ?
Fournir des informations personnalisées sur les retards et perturbations	Il y a peu de personnalisation possible au-delà de la ligne ou du train utilisé par le voyageur. Les informations associées à ces niveaux de granularité sont déjà fournies par les services.
Prévoir les prochains voyages d'un utilisateur.	Il est possible de faire des statistiques sur ses voyages passés pour identifier des trajets fréquents, sans aucune IA.
Prévoir les prochains retards et perturbations.	Du machine Learning « classique » peut être réalisé à partir de paramètres comme l'âge des équipements (trains, feux de signalisation…), la performance historique des lignes et des équipes, la météo, etc. L'IA générative peut aider à extraire les informations de documents non structurés qui pourront être ajoutés à une base structurée.
Conseil de lectures ou de séries et films pour occuper le temps du voyage et de l'attente en gare.	Peut se faire avec des moteurs de recommandations, voire simplement l'actualité culturelle du moment, sans IA générative.
Assister les opérateurs du support client dans la rédaction de leurs réponses.	Cas d'utilisation classique de l'IA générative, mais qui était déjà largement gérée par des chatbots plus légers et des templates de messages préparés par les équipes support. Des humains doivent rester impliqués pour éviter les problèmes dus à de potentielles hallucinations. L'entreprise Klarna, qui avait misé sur l'IA pour son service client et avait licencié 1800 personnes en 2023, réembauche aujourd'hui, car « il s'avère que les gens préfèrent parler à d'autres personnes »(18).
Personnifier les clients dans le cadre de la conception d'application (persona)	Ce cas d'usage d'IA générative me paraît très pertinent et relativement peu cité par rapport à son intérêt.

Dans cet exemple fictif, il est possible que l'utilisation d'IA générative puisse permettre de répondre à ces différents cas d'usage, mais dans le cas général, ce n'est pas le meilleur choix.

Optimiser les appels d'API

Vous êtes maintenant convaincu que l'IA générative est le meilleur choix possible pour votre cas d'usage. Comment limiter les impacts environnementaux associés ?

Dans le cas (le plus répandu), où l'application fait des appels directs à un service via une API (par exemple l'API d'OpenAI ou celle de Mistral AI), trois axes d'optimisation sont possibles :

- Choisir le service et le modèle
- Limiter le nombre d'appels
- Réduire les ressources utilisées à chaque appel.

Choix du service et du modèle

Si vous avez bien regardé le graphique comparant les impacts de différents modèles plus haut dans cet article, vous l'avez sans doute deviné : l'action qui a, de loin, le plus d'influence sur la consommation de ressources des IA génératives est le **choix du modèle**.

Les critères de choix sont :

- **Choisir des modèles performants**. Le comparateur de hugging face peut être utilisé pour avoir une idée des impacts associés à un modèle. Le modèle Meta Llama 3.1 8B consomme par exemple 75 fois moins d'énergie que GPT

(16) https://www.theregister.com/2025/05/06/ibm_ai_investments/

(17) Papier d'origine : https://arxiv.org/pdf/2311.16863 et interview dans le monde https://www.lemonde.fr/pixels/article/2024/03/25/intelligence-artificielle-le-bilan-carbone-de-la-generation-d-images-de-textes-ou-de-sous-titres_6224138_4408996.html

(18) https://gizmodo.com/klarna-hiring-back-human-help-after-going-all-in-on-ai-2000600767

4 Turbo (attention les chiffres des impacts des modèles d'Open AI sont des estimations et l'écart est peut-être moins important).
- **Utiliser des modèles plus petits** (avec moins de paramètres), ce qui va amener à une consommation de ressources moindre. Des modèles plus petits seront potentiellement moins performants, mais cela peut être compensé en partie par du *prompt engineering*, voire du *fine tuning*.
- **Choisir des modèles hébergés en France**. L'électricité en France a moins d'impact qu'en Irlande ou aux États-Unis. Choisir par exemple d'utiliser les modèles de Mistral AI est une optimisation possible, qui converge avec les questions de souveraineté et d'indépendance technologique.

Il existe des benchmarks comparant les performances des modèles, comme celui de Artificial Analysis(19), permettant de trouver le meilleur compromis entre ces différentes dimensions. Des tests peuvent être effectués selon votre cas d'usage pour sélectionner un modèle qui répond assez bien au besoin tout en étant dimensionné au plus juste.

Limiter le nombre de requêtes aux API.
Une fois le modèle choisi, limiter le nombre d'appels permet de consommer moins de ressources numériques.

Un appel à une API d'IA générative n'est pas très différent de n'importe quel appel d'API coûteux et des stratégies similaires peuvent être mises en place pour en réduire le nombre et les coûts associés.
- **Mettre en cache** les réponses si applicable.
- **Regrouper les requêtes similaires** : dans le même ordre d'idée, bufferiser les requêtes et fusionner celles qui sont identiques pour ne faire qu'un appel.
- **Fixer des quotas** si les appels sont déclenchés par des actions utilisateur.
- **Regrouper les tâches à effectuer** : demander plusieurs résultats à l'API en une seule requête si possible. Par exemple le prompt « *Donner les publications de Ada Lovelace, ses apports à l'informatique et ses principaux collaborateurs* » contient trois tâches.

De manière plus spécifique aux IA génératives, il est possible d'optimiser les requêtes via des techniques de ***« prompt engineering »*** pour obtenir une information pertinente directement, sans avoir à réinterroger le modèle.

Voici deux exemples de stratégie de « *prompt engineering* » :
- **Few-shot prompting** : fournir des exemples au modèle lui permettant de mieux « comprendre » le résultat attendu.
- **Chain-of-thought prompting** : guider le modèle avec des étapes de raisonnement, ce qui permet à l'outil de suivre le même processus sur d'autres données.

Exemple de *Few-shot prompting* :

Prompt :
Ada Lovelace est née en 1815.
Grace Hopper est née en 1906.
Hedy Lamarr est née en 1914.

Compléter dans le même esprit :
Margaret Hamilton est née en ?

Exemple de *Chain-of-thought prompting* :
Prompt :
Problème : Si un train quitte Paris à 8h00 et roule à 120 km/h, et un autre train quitte Lyon à 9h00 en direction de Paris à 100 km/h, à quelle heure se croiseront-ils, sachant que la distance entre Paris et Lyon est de 480 km ?
Chain-of-Thought :
1. Le train de Paris a une heure d'avance.
2. À 9h00, le train de Paris a déjà parcouru 120 km (120 km/h * 1h).
3. La distance restante entre les deux trains à 9h00 est de 480 km - 120 km = 360 km.
4. La vitesse combinée des deux trains est de 120 km/h + 100 km/h = 220 km/h.
5. **Le temps nécessaire pour parcourir 360 km à 220 km/h est de 360 km / 220 km/h ≈ 1,636 heures.**
6. **Convertir 0,636 heure en minutes : 0,636 * 60 ≈ 38,18 minutes.**
7. Les trains se croiseront environ 1 heure et 38 minutes après 9h00.
8. Ajouter ce temps à 9h00 donne environ 10h38.

Réponse : Les trains se croiseront à environ 10h38.

Réponds maintenant au problème : Si un vélo quitte Paris à 9h00 et roule à 10 km/h, et un autre vélo quitte Lille à 9h00 en direction de Paris à 10 km/h, à quelle heure se croiseront-ils, sachant que la distance entre Paris et Lille est de 200 km ?

Réduire le nombre d'appels a aussi pour avantage de potentiellement réduire les coûts opérationnels associés, puisque la facture dépend du nombre de *tokens* reçus et envoyés.

Réduire la quantité de ressources associées aux requêtes
Une autre piste pour réduire les ressources dépensées est d'optimiser les requêtes pour qu'elles soient moins coûteuses.

D'abord, **éviter de générer du contenu multimédia** (image ou vidéo), une tâche beaucoup plus lourde pour ces outils que générer un texte. Vous l'avez peut-être constaté en expérimentant avec les chabots : le temps de réponse pour la génération d'image est bien plus grand que pour de la génération de texte. Des limites assez basses sont d'ailleurs en place sur les comptes non professionnels.

Une fois ces deux points posés, est-il possible de modifier les requêtes afin de diminuer les ressources utilisées côté serveur ? La réponse n'est pas évidente, car il y a peu de transparence de la part des acteurs sur ces sujets et assez peu de ressources disponibles. Les paragraphes qui viennent sont donc très spéculatifs et sujets à être challengés dans le futur.

En explorant les documentations, il est toutefois possible de faire des suppositions, notamment avec une **métrique proxy intéressante : la latence**. Le temps de réponse de l'outil est en effet lié au temps de travail du modèle côté serveur et donc aux ressources dépensées pour répondre à la requête.

(19) https://artificialanalysis.ai/models

Dans sa documentation, OpenAI propose différentes techniques pour diminuer la latence(20), parmi lesquelles :

- **Utiliser des modèles plus petits**, et compenser avec du « *prompt engineering* »
- **Limiter la taille des réponses** : si l'outil a moins de données à générer, cela consomme moins de ressources.

Voici par exemple une façon de choisir un modèle et limiter la taille par défaut des réponses avec la librairie java langchain4j :

```
MistralAiChatModel model = MistralAiChatModel.builder()
    .apiKey(ApiKeys.MISTRALAI_API_KEY)
    .modelName(MistralAiChatModelName.MISTRAL_MEDIUM_LATEST)
    .maxTokens(250)
    .build();
```

D'autres moyens d'accélérer l'obtention de résultats sont cités, comme la parallélisation des requêtes ou le streaming des réponses, mais ce sont les deux points ci-dessus qui semblent exercer une influence sur la quantité de calcul réalisée.

Toujours dans la documentation d'OpenAI, il est intéressant de noter qu'il existe un **cache des fragments de prompts** répétitifs qui peut être exploité(21). Pour des prompts très larges, contenant du contexte fourni au LLM et des instructions communes entre les requêtes, il est donc intéressant de structurer le texte en donnant aux fragments répétés la même forme et le même emplacement. Cette stratégie permettrait de réduire la latence jusqu'à 80 % et le coût de 50 % d'après Open AI. **Figure 3**

Utiliser la latence comme métrique proxy est une stratégie intéressante, mais limitée. Le temps de calcul n'est pas le seul paramètre pouvant influer sur la consommation de ressources du modèle.

Il existe au moins un benchmark analysant l'influence de la forme du *prompt* sur la consommation de ressources, par Adamska et al.(22) de l'université de Lancaster. L'équipe a expérimenté avec divers types de *prompts* et des modèles *open source* pour voir ce qui était le moins consommateur d'énergie. Parmi les résultats, ils montrent que :

- **La longueur de la requête a peu d'importance**, au contraire de celle de la réponse. Ce travail confirme donc l'intérêt de **limiter la taille des réponses**.
- Certaines tâches sont plus complexes à réaliser. Les mots clés comme « *analyse* » ou « *explain* » amènent ainsi à deux fois plus de dépense de ressources que « *classify* » ou « *translate* ».

En résumé, pour optimiser les ressources dépensées par les appels aux API d'IA génératives, il faut :

- Éviter la génération de contenu multimédia
- Limiter la taille des réponses
- Utiliser du « *prompt engineering* » pour augmenter la qualité des résultats et minimiser le nombre de requêtes
- Maximiser le « *prompt caching* »

Optimiser l'utilisation des modèles en local

Dans le cadre de la réalisation d'un service numérique basé sur l'IA, il est également possible d'utiliser un modèle en local et non via une API.

Dans ce cas, le cycle de développement du produit (conception, développement, test…) est augmenté d'une nouvelle phase propre au Machine Learning : l'*entraînement*, permettant d'ajuster le programme aux besoins de l'application. L'entraînement des modèles peut être très coûteux, et c'est donc sur cette phase que se concentrent les conseils fournis par les référentiels d'écoconception traitant d'IA.

Notons que les points listés plus haut sur l'optimisation des requêtes sont toujours pertinents pour un modèle auto-hébergé. D'autant plus que les conséquences réelles de ces optimisations pourront être mesurées sur les composants déployés.

Les référentiels d'écoconception disponibles

Il existe actuellement au moins trois référentiels comprenant des points liés à l'écoconception de services numérique utilisant de l'IA :

- Le **RGESN(23)**, référentiel général d'écoconception de services numériques de l'ARCEP (Autorité de régulation des communications électroniques, des postes et de la distribution de la presse). Ce document a été commandé par la loi REEN et comporte toute une section, improprement appelée « algorithmie », qui traite de Machine Learning.
- L'**AFNOR Spec 2314(24)**, « référentiel général pour une IA frugale », issue de l'AFNOR (Association française de normalisation) un document qui comporte une trentaine de

(20) https://platform.openai.com/docs/guides/latency-optimization

(21) https://platform.openai.com/docs/guides/prompt-caching

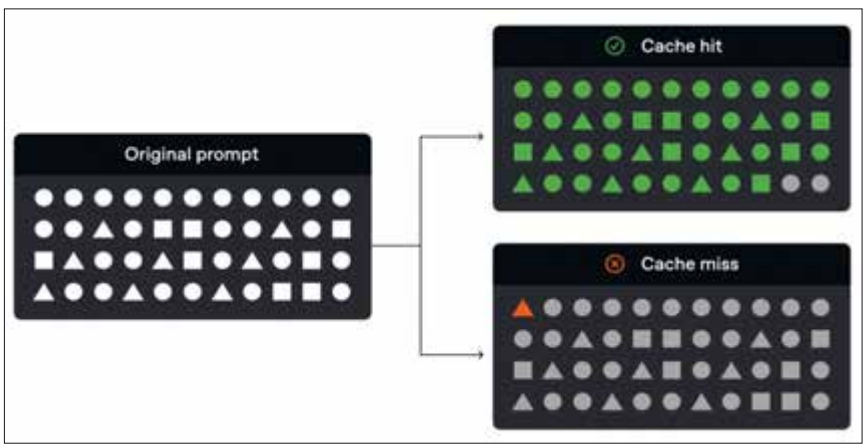


Figure 3 : *Image issue de la documentation d'Open AI*

(22) https://arxiv.org/pdf/2503.10666

(23) https://www.arcep.fr/mes-demarches-et-services/entreprises/fiches-pratiques/referentiel-general-ecoconception-services-numeriques.html

(24) https://www.boutique.afnor.org/fr-fr/norme/afnor-spec-2314/referentiel-general-pour-lia-frugale-mesurer-et-reduire-limpact-environneme/fa208976/421140

fiches donnant des conseils pour une frugalité technique et fonctionnelle des solutions basées sur l'IA.
- La section « *artificial intelligence* » du catalogue de **Green Software Patterns**(25) de la Green Software Foundation. Ce référentiel comprend aussi des sections « Cloud » et « Web ».

Les pratiques encouragées

Ces différents documents, malgré leurs formes différentes, convergent dans leurs recommandations. Le premier point mis en avant est, comme déjà mentionné, de chercher une solution plus légère que l'IA pour répondre au besoin.

Une fois cette première étape passée, ils se focalisent sur la phase d'entraînement, qui peut être très coûteuse. Divers moyens peuvent être mise en œuvre pour limiter son impact :
- **Utiliser des modèles déjà entraînés.** Le *fine tuning*, qui consiste à ajuster le modèle avec de nouvelles données peut être utilisé.
- **Définir des critères stricts justifiant un réentraînement.** Afin de minimiser le nombre de répétitions de cette phase coûteuse. Quel est le niveau de qualité attendu du modèle ? À quel moment est-on satisfait ? À quel moment ne l'est-on plus ?
- **Minimiser la taille des jeux de données d'entraînement.** Cela peut se faire en choisissant soigneusement les données les plus pertinentes. Un jeu de données plus petit permettra de réduire le volume d'entraînement associé.

Un autre axe abondamment traité est l'optimisation des modèles.

- **Choisir un modèle efficient.** Comme déjà vu plus haut, il existe de grandes différences en termes de ressources consommées entre les différents modèles pour des performances parfois proches.
- **Décomposer les modèles en plus petits modèles.** Pour réduire leur taille et les ressources nécessaires pour l'entraînement et l'inférence.
- **Réduire la taille des modèles et les compresser(26).** Il existe de nombreuses méthodes permettant d'améliorer les performances des modèles :
  - **Transfer Learning** : Réutilisation d'un modèle préalablement entraîné pour de nouvelles tâches en retirant certaines en remplaçant des couches du réseau de neurones.
  - **Knowledge distillation** : Transfert des connaissances d'un modèle d'IA « professeur » déjà formé vers un modèle « élève » plus petit.
  - **Pruning** : Méthode de compression qui consiste à supprimer les connexions ayant le moins de poids dans un réseau de neurones (jusqu'à 90%).
  - **Quantification** : Réduction de la précision des données, en réduisant le nombre de bits utiles à la réalisation des opérations ou au décryptage des données.

L'AFNOR Spec 2314 insiste beaucoup sur la mutualisation des modèles et des jeux de données afin de réduire les coûts environnementaux. Cela peut se faire en interne dans le cadre d'une grande entreprise avec une bibliothèque contenant les travaux déjà réalisés.

De façon plus globale, de nombreux modèles sont disponibles publiquement, par exemple sur les sites de Hugging Face(27), Kaggle(28) ou Tensor Flow(29). De même, il existe des jeux de données *open sources*, encore une fois chez Hugging Face(30), Kaggle(31) et Tensor Flow(32), mais aussi via le portail open data de l'ADEME(33).

En résumé

Au moment où j'écris ces lignes, le sujet des impacts des IA génératives reste brûlant et des actualités contradictoires se succèdent. Les modèles tendent à devenir plus légers, ce qui pourrait rendre optimiste sur la croissance des impacts qui ne devrait pas être trop importante.

Dans le cas où vous avez un cas d'utilisation pertinent de ces outils dans votre projet, utilisez des outils comme le comparateur d'hugging face pour bien choisir votre modèle en équilibrant les performances et les impacts associés. Puis appliquez des techniques classiques pour limiter le nombre d'appels (cache, quotas, batchs…) et optimisez ceux-ci en demandant des réponses plus courtes.

Les référentiels de l'ARCEP, de l'AFNOR et de la Green Software Foundation peuvent être utilisés dans le cadre d'une démarche d'écoconception de service numérique incluant un ou des modèles d'IA générative (ou plus généralement de Machine Learning) nouveau ou adapté d'un modèle existant.

(25) https://patterns.greensoftware.foundation/catalog/ai/

(26) https://france-science.com/lia-frugale-bouleverse-les-codes-technologiques-decryptage-des-solutions-techniques-innovantes-depuis-la-silicon-valley/

(27) https://huggingface.co/models

(28) https://www.kaggle.com/models

(29) https://www.tensorflow.org/resources/models-datasets

(30) https://huggingface.co/docs/datasets/index

(31) https://www.kaggle.com/datasets

(32) https://www.tensorflow.org/resources/models-datasets

(33) https://data.ademe.fr/

Flutter Eco Mode : un plug-in pour créer des apps mobiles plus éco-responsable

Alexandre Poichet
Developpeur FullStack chez SNCF Connect & Tech

J'ai découvert Flutter en 2019 et depuis j'ai du mal à le lâcher... Bien conscient du potentiel que peuvent apporter des technologies cross plateforme comme Flutter, je suis convaincu de l'importance d'apporter une dimension environnementale dans l'utilisation de ces frameworks.

Comment se comportent nos apps quand on a presque plus de batterie ? Un faible réseau ? Quand on a un smartphone vieillissant ? En tant que développeur mobile, on peut désormais apprendre à gérer ça…Cet article présente le plug-in Flutter Eco Mode, un outil pour créer des applications mobiles plus éco-responsables tout en favorisant l'expérience utilisateur.

Éco conception et Produit

La pratique d'un numérique plus responsable est souvent perçue comme un simple enjeu technique visant à rationaliser les coûts. Mais non ! Il s'agit aussi d'une question de conception fonctionnelle : comment concevoir des applications qui s'adaptent à notre environnement et de manière éco responsable ? Mais pourquoi tant de difficultés ? C'est là qu'entre en jeu ce que j'appelle le ***Paradoxe Produit***. En tant que produit, on veut s'adapter de façon plus responsable aux utilisateurs en fin de cycle, mais il est bien trop tard donc on abandonne… ça vous parle ? Tant mieux ! Car rappelez-vous de l'accessibilité, c'était la même histoire, à force de traiter cet élément en bout de chaîne, on accumulait de la dette technique et fonctionnelle impossible à rattraper. Il est temps d'introduire des usages éco-responsables de nos applications dès la phase de conception du produit. **Figure 1**

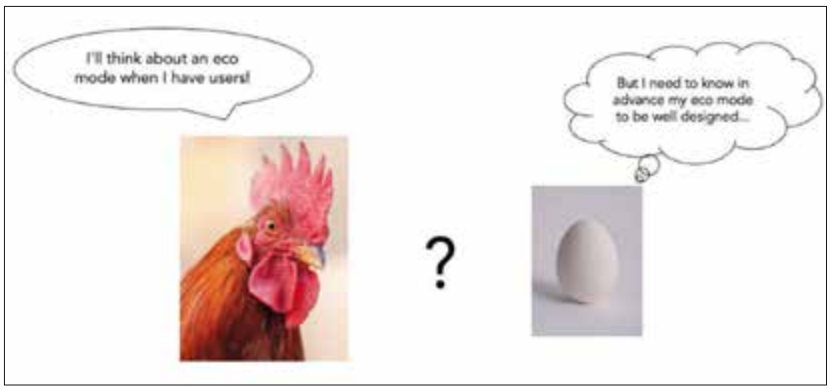


Figure 1 : ***Paradoxe Produit*** *: le paradoxe de l'œuf logiciel et la poule produit*

Au commencement il n'y avait rien… Puis la lumière jaillit avec les tests : différentes stratégies de tests pour concevoir un produit viable. À la suite de cela, nous avons décidé de prendre en considération la taille des écrans des appareils. Ensuite, soucieux de notre prochain, nous concevons désormais des applications accessibles à tous. **Demain, soucieux de notre environnement, nous finirons par concevoir des produits qui s'adaptent** *de* manière plus responsable **à nos usages tout en favorisant l'expérience utilisateur.**

Figure 2

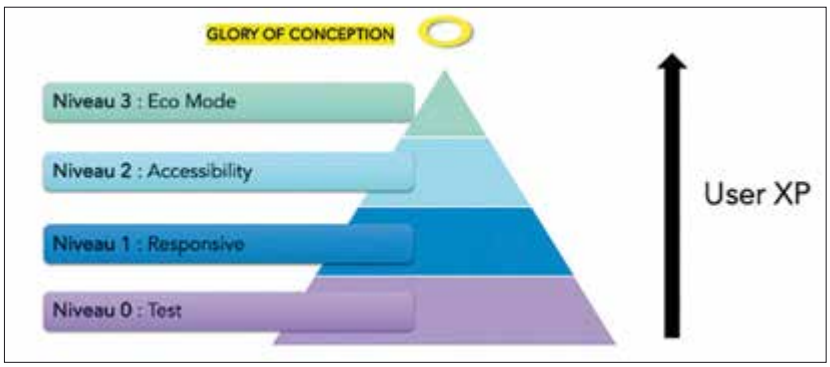


Figure 2 : ***Pyramide de conception*** *: niveau 0 le Test, niveau 1 le Responsive, niveau 2 l'Accessibilité et niveau 3 le Mode Eco*

Éco mode et Fonctionnalités

Low-End Device.

Et si on était capable de détecter le caractère *vieillissant* des smartphones que nous utilisons ? On pourrait alors adapter un parcours fonctionnel dédié, qui demanderait moins de ressources physiques, entre autres. Mais alors, c'est quoi un ***"low-end device"*** ? Difficile de trouver dans la littérature une définition précise, mais nous partirons du principe que c'est un smartphone ***pauvre*** **en ressources et avec des capacités faibles,** telles que la RAM, le stockage, le processeur, la version de l'OS, etc. On va donc aller chercher ces informations sur le terminal en utilisant les méthodes natives, puis calculer un éco score pour le renvoyer à l'utilisateur du plug-in.

- iOS (Swift) : **figure 3**
- Android (Kotlin) : **figure 4**

On peut ainsi définir s'il s'agit d'un low-end device ou non en Dart, côté Flutter : **figure 5**

Battery Eco Mode.

On peut savoir quand l'utilisateur passe en mode économie d'énergie, mais également prendre le relais à sa place. En effet, **si la batterie est en dessous de X% et en décharge, on**

Figure 3

```
func getEcoScore() throws -> Double? {
        let nbrParams = 3
        var maxScore = nbrParams

        let totalMemory = try getTotalMemory()
        let processorcount = try getProcessorCount()
        let totalStorage = try getTotalStorage()

        if (totalMemory <= 1_000_000_000) {  maxScore = maxScore - 1 }
        if (processorcount <= 2) {  maxScore = maxScore - 1 }
        if (totalStorage <= 16_000_000_000) {  maxScore = maxScore - 1 }

        return Double(maxScore / nbrParams)
    }
```

peut activer un mode économie de batterie pour l'utilisateur. On va donc écouter ces infos batteries, y ajouter nos propres règles pour définir un mode économie de batterie (Flutter-Dart). **Figure 6**

Connectivity.

Il est possible de lire les informations réseau sur le terminal et les restituer à l'utilisateur. Avec ces infos, on peut définir si le réseau est absent, faible ou bon. Bien sûr, ce n'est pas aussi fiable qu'un ***"speed test"***, mais cela correspond plus à la philosophie de l'outil : éviter de joindre constamment un serveur pour analyser des temps de réponses, ce qui peut représenter un impact écologique non négligeable… On préfère donc s'appuyer directement sur les informations fournies par l'appareil lui-même. On commence par catégoriser le type de réseau entre l'Ethernet, le wifi et le cellulaire. Dans le cas du cellulaire, on peut aller plus loin et distinguer la 2G, 3G, 4G, 5G. Et pour le wifi, on peut également aller chercher la puissance du signal. Le réseau combiné à la puissance du signal nous donne la connectivité.

Sur Android (Kotlin) :

```
fun ConnectivityManager.getNetworkType(
  networkCapabilities: NetworkCapabilities? = null,
  telephonyManager: TelephonyManager,
): ConnectivityType {
    when {
        networkCapabilities?.hasTransport(TRANSPORT_ETHERNET) ==
 true -> ETHERNET
        networkCapabilities?.hasTransport(TRANSPORT_WIFI) == true -
> WIFI
        networkCapabilities?.hasTransport(TRANSPORT_CELLULAR) ==
true -> telephonyManager.networkType()
        else -> NONE
    }
}

fun TelephonyManager.networkType(): ConnectivityType {
  when (dataNetworkType) {
    NETWORK_TYPE_GPRS, NETWORK_TYPE_EDGE, NETWORK_TYPE
_CDMA,
    NETWORK_TYPE_1xRTT, NETWORK_TYPE_GSM
    -> return MOBILE2G

    NETWORK_TYPE_UMTS, NETWORK_TYPE_EVDO_0, NETWORK_TYPE
_EVDO_A,
    NETWORK_TYPE_HSDPA, NETWORK_TYPE_HSUPA, NETWORK_TYPE
_HSPA,
    NETWORK_TYPE_EVDO_B, NETWORK_TYPE_EHRPD, NETWORK_TYPE
_HSPAP,
    NETWORK_TYPE_TD_SCDMA
    -> return MOBILE3G

    NETWORK_TYPE_LTE
    -> return MOBILE4G

    NETWORK_TYPE_NR
    -> return MOBILE5G

    else -> return UNKNOWN
  }
}
fun NetworkCapabilities.getWifiSignalStrength(): Int? = let { transportInfo
 as? WifiInfo }?.rssi
```

```
override fun getEcoScore(): Double {
        val nbrParams = 4.0
        var maxScore = nbrParams

        if (Build.VERSION.SDK_INT < Build.VERSION_CODES.LOLLIPOP) maxScore--
        if (getTotalMemory() <= 1_000_000_000) maxScore--
        if (getProcessorCount() <= 2) maxScore--
        if (getTotalStorage() <= 16_000_000_000) maxScore--

        return maxScore / nbrParams
    }
```

Figure 4

```
DeviceEcoRange _buildRange(double ecoScore) {
    switch (ecoScore) {
      case > minEcoScoreMidRangeDevice:
        return DeviceEcoRange.highEnd;
      case > minEcoScoreLowEndDevice:
        return DeviceEcoRange.midRange;
      default:
        return DeviceEcoRange.lowEnd;
    }
  }
```

Figure 5

```
Stream<bool?> get isBatteryEcoModeStream => CombineLatestStream.list([
        _isNotEnoughBatteryStream(),
        lowPowerModeEventStream.withInitialValue(isBatteryInLowPowerMode()),
      ]).map((event) => event.any((element) => element)).asBroadcastStream();

  Stream<bool> _isNotEnoughBatteryStream() => CombineLatestStream.list([
        batteryLevelEventStream.map((event) => event.isNotEnough),
        batteryStateEventStream.map((event) => event.isDischarging),
      ]).map((event) => event.every((element) => element)).asBroadcastStream();

extension on BatteryState {
  bool get isDischarging => this == BatteryState.discharging;
}

extension _BatteryLevel on double {
  bool get isNotEnough => this < minEnoughBattery;
}
```

Figure 6

Plugin Flutter

Flutter est une technologie qui permet de réaliser des applications cross plateformes sur iOS et Android, mais pas seulement ! On peut aussi produire une application web et desktop avec la même base de code.

Le plug-in est ainsi architecturé en "platform channels" pour favoriser la mise en place d'un éco mode pour d'autres supports que le mobile. Pour communiquer avec le natif, on fait des appels asynchrones en "methods channels", on passe par les "events channels" pour l'écoute active sous forme de flux de données. Comme évoqué, on applique différentes règles métier pour déterminer un mode éco. Mais en tant qu'utilisateur de l'outil, on peut ne pas être convaincu ou en désaccord avec ces règles. **Architecture du plug-in** : "Platform Channels" pour le support, "Methods Channels" pour les échanges asynchrones et "Events Channels" pour l'écoute active des données (**Figure 7**)

C'est pourquoi nous avons choisi de rendre accessibles toutes les méthodes internes utilisées, afin de donner la possibilité de créer soi-même son mode éco à partir des informations

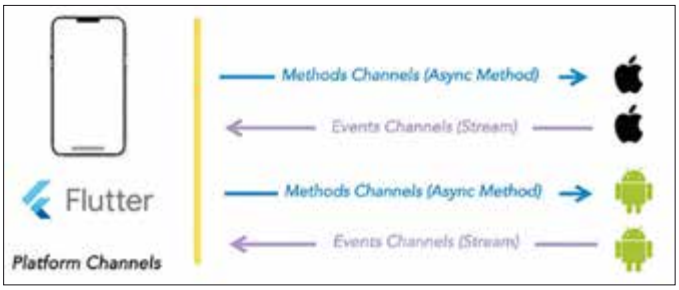


Figure 7

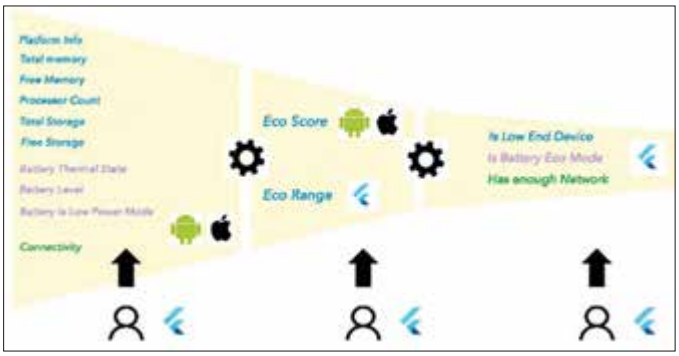


Figure 8

```
Future<bool> isEcoModeEnabled() async {
    final deviceRange = await flutterEcoMode.getDeviceRange();
    final isLowEndDevice = deviceRange?.isLowEndDevice == true;
    final isBatteryEcoMode = await flutterEcoMode.isBatteryEcoMode() == true;
    return isLowEndDevice || isBatteryEcoMode;
  }

  Future<void> init() async {
     bool isEcoModeEnabled = await _ecoModeService.isEcoModeEnabled();
     if (isEcoModeEnabled) {
        accuracy = locationAccuracyEcoMode;
     }
     await _ecoModeService.registerIsBatteryEcoModeStreamEvents(
        switchOnCallback: () => accuracy = LOCATION_ACCURACY,
        switchOffCallback: () => accuracy = locationAccuracyEcoMode
     );

  }
```

Figure 9

natives. Sinon, on peut faire confiance à l'outil et directement appeler nos méthodes relatives au mode éco.

Exposition des méthodes : Liste des méthodes exposées par le plug-in pour les utilisateurs (**Figure 8**)

Pour finir, on sait qu'une écoute active peut faire baisser le niveau de la batterie, c'est pourquoi les données sont également accessibles de façon asynchrone, c'est à dire disponible uniquement à la demande de l'utilisateur du plug-in.

Cas d'usages

Il est temps d'utiliser le plug-in pour créer des parcours utilisateurs éco responsable en activant ou désactivant des fonctionnalités. Ci-dessous une liste d'exemples non exhaustive pour des applications mobiles :

- Stopper les animations
- Passer en Dark Mode (selon le type d'écran)
- Mode hors ligne
- Réduire la précision de géolocalisation
- …

Le plug-in flutter éco mode est actuellement en phase d'expérimentation dans l'application SNCF Connect.

Code en Flutter Dart : **Figure 9**

Pour cette première intégration, nous réduisons la précision de géolocalisation quand la batterie passe en mode éco ou que l'appareil est identifié comme un low-end device. Pour tester cette nouvelle fonctionnalité, il faut accéder au mode bêta de l'application (uniquement ouvert en interne), activer la détection du éco mode depuis les paramètres utilisateurs et bien sûr accepter la géolocalisation.

Réglementation

En France, il existe la RGSEN : Référentiel général d'écoconception de services numériques. On y retrouve plusieurs recommandations dans le domaine du numérique responsable. Les objectifs sont de réduire la consommation de ressources informatiques et énergétiques et la contribution à l'obsolescence des équipements, qu'il s'agisse des équipements utilisateurs ou des équipements réseau ou serveur. Le plug-in s'inscrit pleinement dans la réglementation, car il prend compte le vieillissement de nos terminaux et tend à adapter nos usages de façon plus responsable.

Conclusion

C'est déjà la fin de cet article, mais on peut le résumer en trois parties :

- Construire un **mode éco avec le produit.** Il faut concevoir à plusieurs le mode éco de votre application et le plus tôt sera le mieux. Aujourd'hui ce ne sont que des recommandations, mais demain qui sait ? On va sûrement devoir proposer des alternatives dans nos apps, et toujours dans l'optique d'améliorer l'expérience utilisateur et non de la dégrader.
- Un outil et **pas directement un résultat**. Le plug-in n'est pas magique, il ne va pas faire une application aux usages éco responsables tout seul. C'est bien à vous d'exploiter son potentiel selon vos cas d'usages. Pensez à adopter une démarche disruptive et frugale pour vos utilisateurs, vous obtiendrez de meilleurs résultats.
- Plugin **ouvert à la communauté**. C'est un plug-in open source, disponible sur notre repository. Vous l'avez vu, les règles métiers du mode éco sont encore succinctes et perfectibles, nous avons besoin de vous, de votre expertise pour rendre ces règles plus pertinentes et cohérentes pour ainsi créer le meilleur outil qui soit.

Aller plus loin

Si vous n'utilisez pas Flutter, il existe d'autres bonnes pratiques à mettre en place pour adopter une démarche éco responsable :

- Favoriser les technologies cross plateforme qui permettre d'avoir une même base de code pour plusieurs supports utilisateurs (desktop, web, mobile, etc.)
- Migrer sa logique métier côté serveur back end, pour faciliter les mises à jour et les déploiements tout en réduisant la taille des binaires front end.
- Désactivation ou activation des fonctionnalités à distance, ce qui permet de mieux contrôler sa production, pour éviter de redéployer une nouvelle version tout de suite en cas de problème.

À vous de jouer...

Liens utiles

Conférence DEVOXX 2025 : https://www.youtube.com/watch?v=l9zYcJJs-VI

Autres :

https://github.com/sncf-connect-tech/flutter_eco_mode

https://pub.dev/packages/flutter_eco_mode

https://docs.sentry.io/concepts/search/searchable-properties/#device-classification

https://ecoresponsable.numerique.gouv.fr/publications/referentiel-general-ecoconception/#uxui

J'ai perdu du poids sur Kubernetes avec SlimFaas

Savez-vous que dans les clusters Kubernetes il y a entre 30% à 70% des CPU non utilisés et 20% à 60% de RAM non utilisées en environnement de production. Et encore plus, avec entre 60% à 90% CPU non utilisés et 50 à 80% de RAM non utilisées en environnement de développement, recette et préproduction.

Guillaume Chervet

Passionné par l'informatique, tout simplement !

Principal ML Engineer at AXA France - Microsoft MVP

Autant de ressources qui consomment de l'électricité et augmentent les coûts (et donc la facturation) pour rien. Heureusement, SlimFaas est là pour vous aider, c'est un tout petit proxy HTTP qui se connecte sans couplage à votre architecture existante et permet d'éteindre vos instances de PODS non sollicitées. L'impact GreenIT que vous pouvez obtenir en quelques minutes est tout simplement incroyable ! Cela nécessite tout de même que vous comprenez comment cela fonctionne « sous le capot » et c'est tout l'objectif de cet article.

Pourquoi SlimFaas ?

Prenons un exemple et imaginez que :

- Vous avez en production sur un namespace Kubernetes une application statique HTML + JavaScript servie par un serveur nginx, et que cette application web appelle une API via le protocole HTTP.
- Votre application est utilisée principalement du lundi au vendredi de 8h à 19h uniquement.
- Pour être résilient aux pannes, vous avez doublé le nombre d'instances de pods.
- Un pod nginx nécessite 40 Mo de RAM.
- Un pod de l'API consomme 1 Go de RAM.
- Vous payez 92 € par mois par Go de RAM. Pourquoi la RAM ? Parce que le CPU est partageable entre les instances de POD alors que la RAM ne l'est pas. C'est le plus souvent le goulot d'étranglement. **Figure 1**

Votre infrastructure vous coûte : (1 Go * 2 Pods + 0,04 Go * 2 Pods) * 92 € = **191** € / **mois**

Sachant que l'on a 4 environnements (Développement, recette, préproduction, production) le coût réel est : 191 * 4 = **764** € / **mois** soit **9 168** € / **an** ! **Figure 2**

Imaginez maintenant que l'on éteint votre infrastructure et que vous ne la payez pas entre 19h et 8h du matin ainsi que le week-end. Théoriquement, cela représente une économie de 13h par jour ouvré et 48h le week-end. Votre facture passe de **764** € / **mois** à **255** € / **mois** soit environ **66%** d'économie !

Et vous allez voir qu'avec SlimFaas, vous pouvez économiser encore beaucoup plus !

Comment fonctionne SlimFaas ?

SlimFaas se déploie comme vos propres images Docker, dans le même namespace que votre API. Il agit comme un proxy qui se met en façade de vos API.

Si votre API Kubernetes s'appelle « fibinacci » et possède une route HTTP POST /compute, vous pourrez l'appeler en passant par Slimfaas, par exemple :

HTTP POST http://slimfaas /function/fibinacci/compute {"Input":29}

Slimfaas re-routera alors l'appel à votre pod fibinacci :

HTTP POST http://fibinacci /compute {"Input":29}

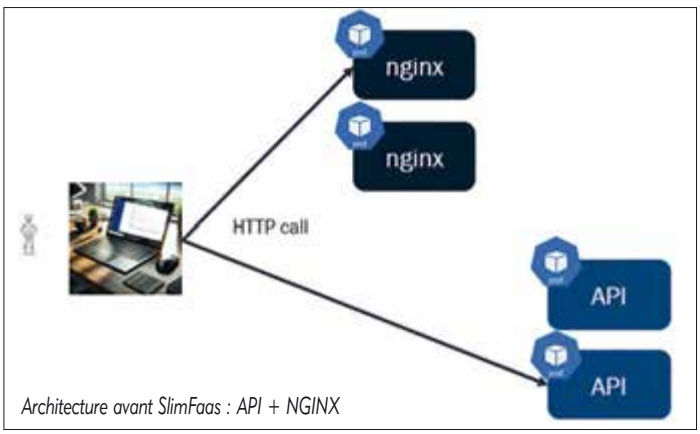
SlimFaas va donc voir passer toutes vos requêtes, et ainsi être capable de savoir lorsqu'un pod n'a pas été appelé depuis une période définie (par exemple 5 minutes) et pourra alors décider de l'éteindre afin d'économiser les ressources inutilisées (scale to 0).

En cas de nouvelle requête, si aucune instance du pod fibinacci n'est allumée, SlimFaas va pouvoir garder la requête en mémoire, lancer le démarrage d'une instance du pod fibinacci, et lorsque celui-ci sera démarré et en état « Ready » (c'est-à-dire répondant à un ping sur une route de type /heath utilisée pour le health check), SlimFaas enverra la requête à « fibinacci ». Dans le monde Kubernetes, une instance d'un pod s'appelle un « replica ».

Son architecture

En production, pour être résiliant aux pannes, il vous faut au minimum 2 replicas pour chaque pod. Le problème avec cela, c'est que chaque replica de SlimFass a besoin d'avoir une vision complète des requêtes qui passent et d'être parfaitement synchronisés, car une requête peut passer par n'importe quelle instance du pod de SlimFaas.

La Solution ? SlimFaas est à la fois un proxy ainsi qu'une



Architecture avant SlimFaas : API + NGINX

Figure 1

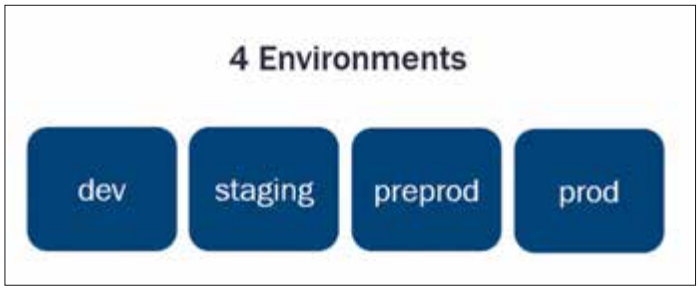


Figure 2

vraie base de données afin d'atteindre un seul et même objectif : être plug and play !
Vous avez une infrastructure existante sur Kubernetes ? Déployez SlimFass, ajoutez **une** seule annotation à vos pods existants et c'est bon : vous pouvez jouer avec !

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
 name: fibinacci
 namespace: slimfaas-demo
spec:
 selector:
  matchLabels:
   app: fibinacci
 template:
  metadata:
   labels:
    app: fibinacci
   annotations:
    SlimFaas/Function: "true" # Declare votre pod a SlimFaas
    SlimFaas/TimeoutSecondBeforeSetReplicasMin: "5" # Eteindre après
5 secondes d'inutilisation
  spec:
   serviceAccountName: default
   automountServiceAccountToken: false
   containers:
    - name: fibinacci1
      image: axaguildev/fibinacci-api:latest
      resources:
       limits:
        memory: "96Mi"
        cpu: "50m"
       requests:
        ephemeral-storage: "200Mi"
        memory: "96Mi"
        cpu: "10m"
      ports:
       - containerPort: 5000
```

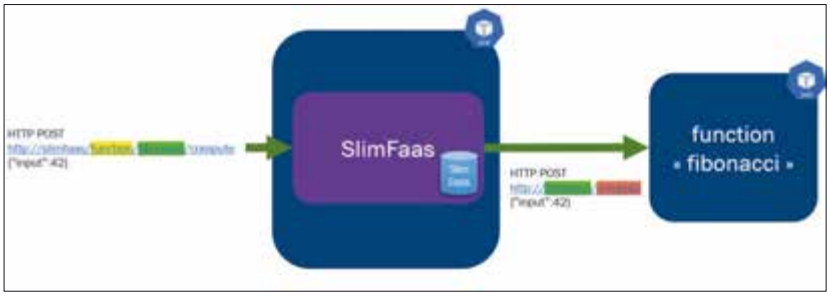


Figure 3 : *Schéma du fonctionnement du Proxy SlimFaas*

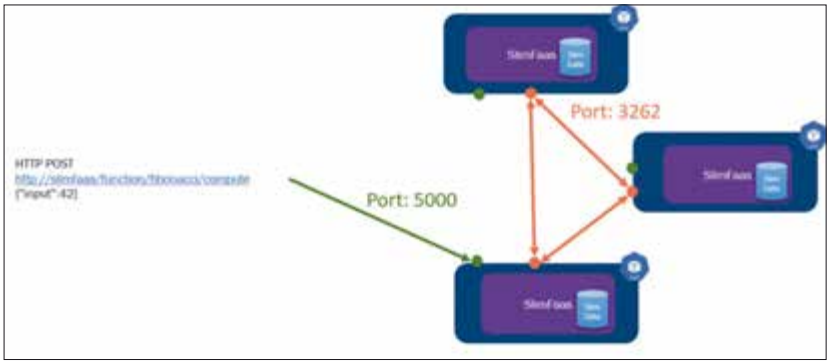


Figure 4 : *Schéma de la communication SlimData entre les replicas SlimFaas*

Configuration Kubernetes d'un Pod Fibinacci avec l'annotation SlimFaas/Function. Figure 3

SlimData est une base de données simple de type Redis incluse dans l'exécutable SlimFaas. Il est basé sur le protocole RAFT. C'est le même protocole qui est aussi utilisé pour les bases de type SQL (ainsi que Redis), avec un nœud principal qui prend les écritures et des nœuds qui réplique la donnée. Par défaut, SlimData utilise un second port HTTP 3262 que les nœuds utilisent pour communiquer entre eux. Pour des raisons de sécurité, ne l'exposez pas en dehors de votre namespace. SlimFaas nécessite au moins 3 nœuds en production pour être résiliant aux pannes. 2 nœuds sont nécessaires pour maintenir la base de données dans un état cohérent. Bien sûr, en développement, un seul nœud suffit. **Figure 4**

Pour être utile, SlimFaas doit consommer le moins de RAM et CPU possible. Aujourd'hui, je configure en production SlimFaas à **80 Mo** de mémoire et **200 m** en CPU (0,2 cœur d'un CPU). Avec notre exemple précédent, SlimFaas coûte : 3 * 0,08 Go * 92 € = **22,08** € / **mois**.

SlimFaas a été codé en .NET 9 et est compilé en mode natif, c'est-à-dire en exécutable indépendant de la Machine Virtuelle .NET 9. Il est actuellement compilé pour les OS x64 et ARM64 (pratique sur un Raspberry).

Problème du démarrage via un trigger http : Planet Saver

Reprenons notre exemple du tout début et ajoutons SlimFaas à notre architecture. L'avantage, c'est que l'on va pouvoir éteindre notre API lorsqu'elle n'est plus utilisée. Afin que SlimFaas ait un véritable intérêt, il faut que votre instance de Kubernetes soit configurée pour pouvoir provisionner des nœuds dynamiquement (c'est-à-dire des machines virtuelles dans lesquelles seront déployés vos pods).

Le démarrage de 0 à 1 replica n'est en réalité pas si simple que cela ! Il peut y avoir un problème. Si une Machine Virtuelle est disponible et que votre API est rapide à démarrer, il n'y a pas de souci. Si, au contraire, l'API est lente à démarrer (par exemple une minute) ou qu'aucune Machine Virtuelle n'est disponible et qu'il faut en provisionner une nouvelle, vos requêtes risquent de tomber en timeout ! Cela entraînera un dysfonctionnement de votre application. Pour information, le provisionning dynamique par une infrastructure Kubernetes d'une machine virtuelle sur Azure nécessite environ 7 minutes !

C'est là où intervient la librairie JavaScript @axa-fr/SlimFaasPlanetSaver, qui utilise des routes spécifiques à SlimFaas pour communiquer des informations au front JavaScript sur l'état de l'infrastructure. L'idée de ce module est de changer les mentalités, un « **Mind Changer** ». Avant SlimFaas, on utilisait minimum deux replicas du pod API afin d'être résiliant aux pannes. Avec SlimFaas et PlanetSaver, l'idée est de transférer cette résilience d'architecture « **Physique** » qui coûte chère à la planète par une résilience

« **UX » (User eXpérience)** côté navigateur. Le petit module JavaScript est écrit en pur JavaScript. Il est compatible avec tous les Frameworks du marché : react, angular, etc. Il appelle une route spécifique de SlimFaas qui communique des informations très sommaire sur l'infrastructure. Exemple :
HTTP POST http://slimfaas/status-functions
qui retourne un tableau JSON du type:
[{"NumberReady":0,"NumberRequested":0,PodType":"Deployment","Visibility":"Public","Name":"fibinacci"}]
Si l'environnement est éteint, le composant JavaScript réalise l'appel à une route spécifique afin de démarrer un replica de l'api et il affiche un loader qui informe l'utilisateur que l'infrastructure démarre.
HTTP POST http://slimfaas/wake-function/fibinacci **Figure 5**
L'avantage de ce composant est que si l'API s'éteint pendant la saisie d'un formulaire par exemple, le « loader » SlimPlanet va s'afficher, mais vous ne perdrez pas votre contexte JavaScript ; vous retrouverez votre formulaire dans l'état où il était auparavant. **Figure 6**
Si l'on refait les calculs, en éteignant votre environnement entre 8 h et 19 h ainsi que tout le week-end, vous économisez environ 66 % des ressources, mais maintenant avec SlimFaas + PlanetSaver, vous pouvez en production utilisez seulement qu'un replica de votre API au lieu de deux à chaque fois. Vous réalisez maintenant **minimum 80%** d'économie ! Votre infrastructure vous coûtera **1 834** € par an au lieu de **9 168** €, soit **7 334** € d'économies !
Si vos utilisateurs ne veulent vraiment jamais attendre, vous pouvez démarrer les replicas de vos pods en avance de phase en utilisant l'annotation « SlimFaas/Schedule » qui s'ajoute sur les scripts Kubernetes de déploiement de votre API. Dans l'exemple ci-dessous, votre API démarre à 7 heures du matin et ne s'éteindra pas de la journée. Cela laisse 1h à votre machine virtuelle pour démarrer si besoin ! Les utilisateurs qui commencent à travailler à 8 h du matin n'auront alors jamais de temps d'attente dans la journée. L'API s'éteindra ensuite à partir de 19h en cas de non-sollicitation. **Figure 7**
Voir tableau ci-contre

Soyez les prochains à témoigner !

Perdre du poids sur Kubernetes est à votre portée. Cela demande bien sûr un investissement ; cependant, SlimFaas met en place les recettes qui vous aideront ! Cet article ne parle pas de tous les ingrédients et fonctionnalités offertes par SlimFaas, il y a aussi par exemple :

- Les fonctions asynchrones, un vrai système de « queue » grâce à SlimData.
- Le système de publication d'événements via HTTP (de type publish-subscribe), sans nécessité de driver qui couple votre code.
- Le démarrage de « Jobs » via des appels API et de manière sécurisée,
- Etc.

A l'origine, créé en Open Source par AXA France, SlimFaas a rejoint début 2025 la Sandbox de la Cloud Native Computing Foundation (CNCF). C'est une fondation sous l'égide de la **Linux Foundation** qui vise à promouvoir l'adoption et le développement des technologies **cloud-native**. Elle héberge et soutient des projets open source essentiel à la construction, au déploiement et à la gestion d'applications modernes.

Figure 5 : *Loader du composant Javascript SlimFaasPlanetSaver*

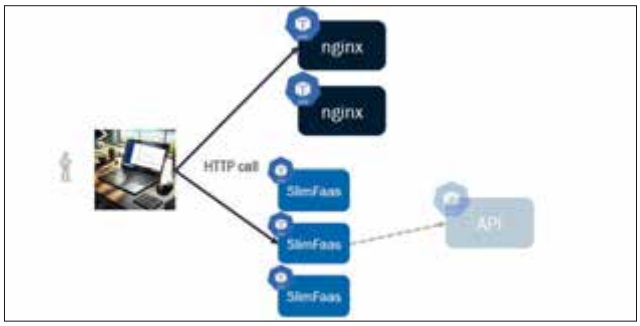
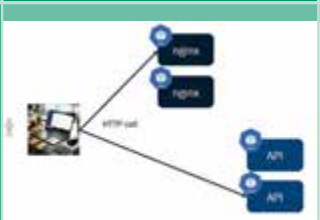
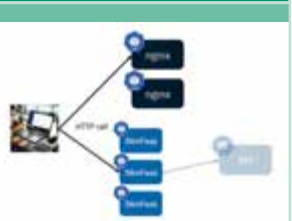


Figure 6 : *Architecture Avec SlimFaas*

Figure 7 : *SlimFaas/Schedule*

TABLEAU RÉCAPITULATIF POUR NOTRE EXEMPLE :

	Avant SlimFaas	Avec SlimFaas
Architecture		
Résilience	Physique : 2 replicas allumés en permanence.	UX : via @axa-fr/SlimFaasPlanetSaver
Temps de latence moyen *(estimé via benchmark 33 request/seconds) sur route http GET)*	7 ms	13 ms (+ 6 ms)
Coût	9 168 € / an	1 834 € /an soit 80% d'économie

CLOUD NATIVE COMPUTING FOUNDATION

Si vous voulez en savoir plus sur SlimFaas ou bien même contribuer, rendez-vous sur la page GitHub :
https://github.com/SlimPlanet/SlimFaas
Soyez les prochains à témoigner de votre perte de poids dans votre cluster Kubernetes ! Un grand merci à Fares Ahmed et Antoine Lernould pour leur relecture !

Qu'est-ce que le Dev "Green" Ops ?

Depuis une quinzaine d'années, les entreprises basculent progressivement d'une solution hébergée dans des datacenters internes, avec gestion des réplications dans des salles blanches, à des solutions cloud.

Matthieu Vincent
Tech'Advocate @ Sopra Steria
https://me.yodamad.fr
Co-Fondateur du meetup GreenTechAuvergne
https://green-tech-auvergne.fr

Sylvain Gougouzian
https://gouz.dev
Co-Fondateur du meetup GreenTechAuvergne
https://green-tech-auvergne.fr

D'un point de vue financier, ce passage au cloud est très intéressant. On minimise les coûts d'infrastructures avec moins d'électricité, pas de gestion de salles blanches, moins de personnels, moins de gestion de sécurité, de maintenance… Aussi, le serveur installé dans le cloud est bien plus optimisé par l'utilisation.

Pour rappel, une machine consomme presque autant d'électricité utilisée ou non. De nombreux progrès ont été également faits au niveau des infrastructures des bâtiments également nécessitant moins de climatisation, et la technologie est également améliorée au fur et à mesure des années.

Certes, le prochain enjeu pour les cloud providers est d'être le plus efficient possible avec l'arrivée de l'IA qui induit des fortes hausses sur la facture énergétique en comparaison à une consommation classique.

La quête d'un numérique plus responsable reste essentielle, elle appelle à éviter d'ajouter de nouvelles ressources (serveurs) qui consomment des terres rares de plus en plus en tension (disponibilité, géopolitique) et donc à plus de sobriété de nos usages.

La consommation des datacenters reste une partie importante de la consommation de l'IT ; si l'on exclut la partie "terminaux" (smartphones, tablettes, smart TV…) qui reste beaucoup plus gourmande aussi bien dans leur consommation que dans leur coût énergétique de fabrication. Cependant, il ne faut pas minimiser l'impact de nos pratiques de développements et DevOps.

Introduction au DevGreenOps

En reprenant le fameux cycle infini du DevOps, on identifie des axes de travail permettant d'optimiser l'impact environnemental : **figure 1**

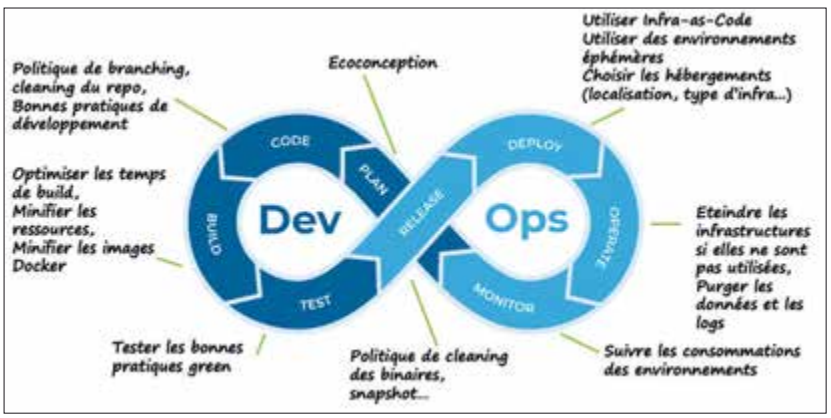
Prenons le temps d'expliquer chaque étape et commençons par l'étape de **conception**. Le lien se fait naturellement avec la pratique d'**écoconception** : comment concevoir une application et ses fonctionnalités en intégrant les notions de frugalité technologique. Avec l'approche d'écoconception, il faut prendre en compte à la fois les nouvelles fonctionnalités et comment optimiser leur implémentation : besoin systématique de temps réel ? Surcharge d'informations ? ; mais il ne faut pas oublier la maintenance des fonctionnalités existantes : supprimer une fonctionnalité inutilisée ? Revoir/simplifier le processus métier existant ?

Une fois la conception aboutie, on démarre alors la phase de **développement**. Sur cette étape, 2 axes sont à prendre en compte : le développement en tant que tel et la gestion du cycle de vie du code source. Dans le cadre du développement, plusieurs initiatives proposent un cadre et des bonnes pratiques : pour le frontend, les 115 bonnes pratiques du web représentent un premier catalogue de règles très complet ; côté API des règles sont proposées par le collectif API Thinking. Bien que ces pratiques soient de plus en plus prises en compte, on oublie trop souvent la gestion du cycle de vie du code source. Même si le code source en tant que tel n'est pas impactant directement, cela reste de la donnée nécessitant du stockage. Maintenir une base saine de code au-delà de la lisibilité de l'historique et de la facilité de maintenance, réduit l'impact environnemental du projet. Pensez donc à faire du ménage dans vos branches et vos tags, ne commiter pas des éléments inutiles ou qui n'ont pas lieu d'être dans un gestionnaire de sources.

En aval du développement, il est devenu indispensable de mettre en place une chaîne d'intégration continue, dites CI/CD, afin d'automatiser tout un ensemble de contrôles et tâches récurrentes.

La mise en place de Github Actions, ou son équivalent chez Gitlab, est devenu assez aisée. Il est donc possible d'ajouter nos propres outils pour valider nos développements. On trouve sur "l'étagère" des solutions comme SonarQube qui permette de diminuer la dette technique, par exemple et que l'on peut agrémenter du plug-in Eco-code.

Figure 1



Côté Front et Back

Si on fait un focus sur l'automatisation de projet "front", côté GreenIT, il existe un outil développé par la communauté GreenIT.fr : le *Green-Analysis CLI*. Son but est de calculer un score pour un site web entre A et G par rapport à son grade d'éco-conception. Ce calcul est basé sur 3 données : la taille du DOM, la complexité d'une page web, le nombre de requêtes et le poids de la page et des "assets".

Une fois intégré dans notre chaîne CI, comme il est possible de bloquer une (Merge|Pull) Request avec un taux de couverture de tests unitaires inférieur à un certain pourcentage avec SonarQube, il est envisageable de bloquer une CICD si le grade d'une application est inférieur à C ou D. Sur des projets de types Node, Deno ou Bun, il est également possible d'informer de l'impact de l'utilisation d'une

bibliothèque JS avec des outils comme *BundlePhobia* ou *PackagePhobia*. Il est important d'alerter si une bibliothèque externe est "Tree-Shakable", c'est-à-dire si elle peut être partiellement découpée lors de la génération du "bundle". Ainsi, on peut minimiser l'impact du poids final de notre application. Souvent cette partie n'est pas visible, car on utilise des bibliothèques connues sans se poser de question, et très souvent pour une fonctionnalité qui représente peu du package total. **Figures 2 et 3**

De plus en plus de bibliothèques proposent également des sous-divisions d'elles-mêmes afin de minimiser leur impact. Par exemple, il est possible de récupérer la lib' "@inquirer/prompts" qui dispose de différents composants pour interagir avec un utilisateur dans un terminal. Celle-ci pèse 24 ko et comprend une "checkbox", une liste de choix, un prompt,… Si l'on utilise seulement "confirm", il est préférable d'utiliser "@inquirer/confirm" qui pèse seulement 9,87 ko.

Ces outils nous permettent également de connaître leur "profondeur", c'est-à-dire le nombre de dépendances qu'elles ont. Nous voyons deux utilités à ce genre d'outils :

- une meilleure maîtrise du poids de notre application
- une meilleure vision de la sécurisation de notre application

En effet, moins de dépendances nous avons, moins de failles potentielles nous obtenons.

Il existe des équivalents pour des projets "back" tel que l'outil *DepClean* qui permet d'analyser les dépendances d'un projet Maven et d'identifier celles qui seraient inutiles. Ainsi on peut facilement réduire l'empreinte de son application en configurant de manière avancée les dépendances (et les dépendances indirectes) qui sont utilisées par nos projets.

Côté conteneur / infrastructure

Dans un monde conteneurisé, il est important également de garder sous contrôle la taille de nos images, car même si nous optimisons nos binaires applicatifs, l'image que l'on utilise pour les déployer comprend souvent des mega-octets de data inutile. Des outils existent donc pour analyser et optimiser la taille d'une image Docker tels que *Dive* ou *Slim* (que l'on retrouve facilement sur GitHub). Avec ces outils, vous pourrez facilement automatiser le contrôle et nettoyer les éléments inutiles. Cela aura la vertu supplémentaire de limiter la potentielle surface d'attaque de votre conteneur une fois déployé.

Nos applications sont prêtes et optimisées pour être déployées. Mais avant cela, une étape primordiale est à avoir en tête : **l'hébergement** et le **dimensionnement**. Concernant l'hébergement, si vous optez pour un hébergement Cloud, les principaux cloud providers fournissent leur propre tableau de bord "Green", les mettant souvent en lumière en comparaison de leurs concurrents… Le collectif Boavizta met à disposition un outil, *Datavizta*, permettant d'avoir une visualisation moins biaisée du coût environnemental de votre hébergement cible. Ainsi, grâce à cet outil, vous pourrez ajouter une dimension "Green" à votre matrice de décision lorsque vous devrez faire un choix. Ayez toujours en tête que la production d'électricité présente des vrais écarts d'impact environnemental en fonction du pays dans lequel elle est réalisée. Pour visualiser ces écarts, le site *ElectricyMaps*

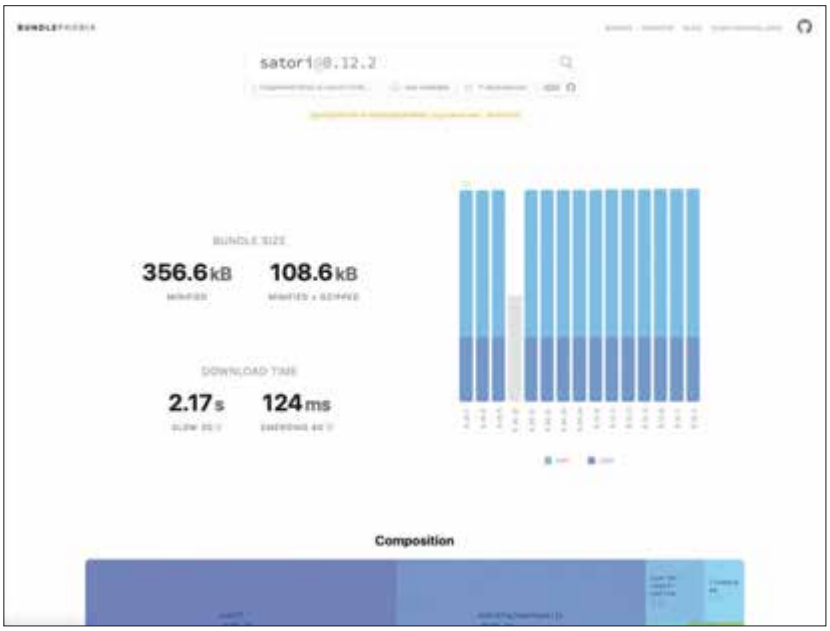


Figure 2 : *Exemple avec bundlephobia.com*

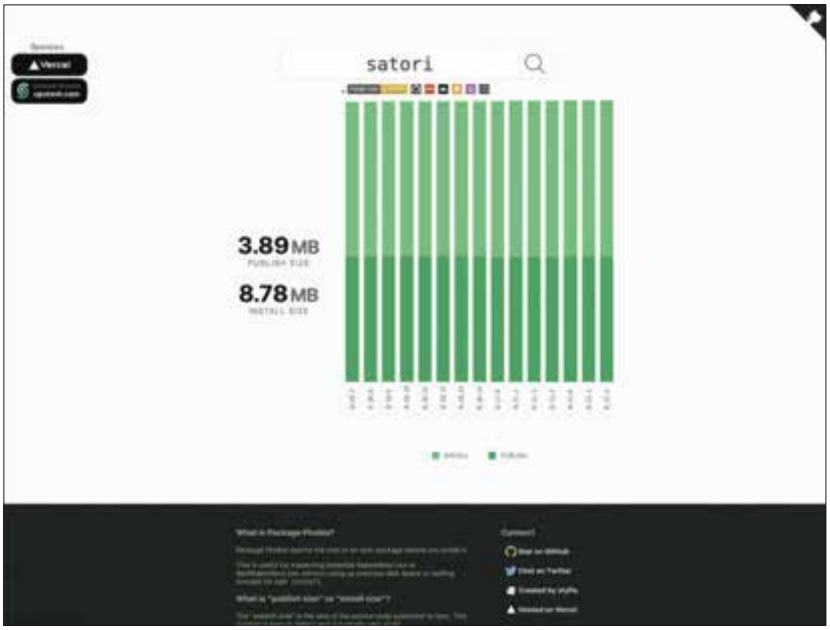


Figure 3 : *Exemple de packagephobia.com*

permet de comprendre les différences en fonction des différents pays. **Figure 4**

Concernant le dimensionnement, la tendance des architectures modernes, dites micro-services, apporte une dérive de surcharge de ressources. Mais est-ce que tous les services ont besoin d'être répliqués 4, 5 fois voire plus alors que vous avez une centaine d'utilisateurs ? De même que lors de la conception fonctionnelle de notre application, il est important de définir notre cible (nombre d'utilisateurs, localisation des utilisateurs, plage horaire d'utilisation, criticité du système…) pour choisir les architectures logicielles et systèmes adéquats et dimensionner correctement les ressources qui seront nécessaires pour l'atteindre.

Une fois notre infrastructure définie, il est temps d'installer et de déployer nos composants et là encore, il est crucial de se poser la question du niveau de service nécessaire pour chacun des environnements : un environnement de dev ou de test n'a probablement pas besoin du même niveau de

Figure 4

Figure 5

service que la production. Aussi, il est probable, en tout cas on va leur souhaiter, que les équipes de dev et de tests ne travaillent pas 24/7. On peut donc optimiser la consommation de ressources en éteignant les services et serveurs qui n'ont pas besoin d'être démarrés en continu. 2 approches sont à prendre en compte pour accompagner ces optimisations : l'IaC ou Infrastructure-as-Code et une politique d'autoredémarrage.

Grâce à l'IaC, on va pouvoir simplement et rapidement détruire et reconstruire des environnements à la demande, permettant ainsi d'avoir des versions déployées automatiquement et le plus à jour possible tout en ne supprimant pas les composants qui ne sont plus utilisés et/ou plus à jour. Ainsi, plus besoin de conserver "au cas où" un environnement dans une version donnée, qui plus est un environnement "allumé".

En complément, une politique de gestion des arrêts / démarrages en automatique permet de simplement optimiser les ressources. Dans un environnement conteneurisé dans Kubernetes, des outils tels que *Kube-green* ou *kube-downscaler* permettent via un simple paramétrage d'arrêter tout ou partie des composants démarrés dans notre cluster. Idéalement, il faut pouvoir faire du "scale 0" afin d'aussi déprovisionner les nœuds, mais en arrêtant une bonne partie des workloads, on peut déjà réduire significativement leurs nombres.

Pour des ressources plus classiques type VM ou des services managés sur les plateformes de cloud, il est possible de mettre en place des règles d'auto-shutdown et d'auto-restart sur vos ressources afin d'également les mettre à disposition uniquement quand cela est nécessaire. Bonne nouvelle en bonus, cela allégera aussi votre facture ! **Figure 5**

Côté Data

Une fois nos applications en production, celles-ci génèrent de la donnée, beaucoup de données, en continu. Mais est-ce nécessaire de conserver toutes ces données ?! De quelles données parle-t-on d'ailleurs ?

Il y a évidemment les données métiers stockées dans les bases de données. Hormis certains cadres légaux qui imposent de les conserver sur plusieurs années voire dizaine d'années, dans les autres cas, il est fort possible de pouvoir purger des données anciennes qui ne seront plus utilisées. On en revient à notre étape initiale de conception où il est important de ne pas considérer que les nouveautés, mais également de prendre en compte le cycle de vie des fonctionnalités (et de leurs données) existantes.

Les autres données, moins visibles, mais tout autant énergivores, sont les logs de nos applications et des services utilisés par nos applications. Des logs qui contiennent certes quelques informations pertinentes, mais souvent une grande partie qui ne serait utile qu'à un instant T en cas de problème sur le système. À quoi bon conserver ces données si aucun problème n'a été à déplorer sur une période. La mise en place d'un nettoyage régulier des logs permet de réduire les ressources nécessaires pour leur stockage, de faciliter la recherche d'informations pertinentes.

Adopter une stratégie de gestion raisonnée des données, intégrant des cycles de suppression ou d'archivage adaptés, contribue non seulement à réduire l'impact environnemental des infrastructures, mais aussi à optimiser leur performance. Cette démarche, en alliant conformité réglementaire et pratiques responsables, s'inscrit pleinement dans une logique de sobriété numérique.

En conclusion

En conclusion, même si toutes les étapes et actions présentées n'ont pas le même impact et le même coût de mise en œuvre, chacune d'entre elles peut avoir un rôle à jouer pour améliorer notre prise en compte des problématiques Green. De plus avec des architectures logicielles qui se complexifient, le nombre de choix possibles de service et la facilité d'accès à des services Cloud et donc à du stockage et du compute, actionner toute ou partie de ces axes d'optimisation apportera un gain si nous le passons à l'échelle. Pour illustrer cela, on peut prendre le cas simple d'une requête sur ChatGPT : à l'échelle d'une personne, la requête ne représente pas un gros "coût" environnemental, mais si on multiplie cette même requête à l'échelle de millions d'utilisateurs qui utilisent le service quotidiennement, l'impact n'est pas le même ! C'est que nous propose de comprendre l'outil proposé par HuggingFace : *ecologits-calculator*.

Comme le disait Confucius : "Celui qui a déplacé la montagne, c'est celui qui a commencé par enlever les petites pierres".

Pour une version illustrée de cet article, vous pouvez retrouver un replay sur la chaîne YouTube de France DevOps.

Boutique ÉtÉ Boutique ÉtÉ Boutiq

Les anciens numéros de programmez!

Le magazine des dév - CTO - Tech Lead

260

264

266

267

268

HS18

269

Complétez votre collection.

Tous les numéros sont disponibles en PDF

Tarif unitaire 8 € (6,99€ + 1,01€) (frais postaux inclus)

☐ *260 uniquement en pdf*
☐ *264 uniquement en pdf*
☐ *267 uniquement en pdf*
☐ *266* : |__| ex
☐ *268* : |__| ex
☐ *HS18* : |__| ex
☐ *269* : |__| ex

Commande à envoyer à :
Programmez!
57 rue de Gisors
95300 Pontoise

soit |__|__| exemplaires x 8 € = |__|__|__|__| € .....

soit au **TOTAL** = |__|__|__|__|__| €

☐ M. ☐ Mme Entreprise : |__|__|__|__|__|__|__|__| Fonction : |__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|

Prénom : |__|__|__|__|__|__|__|__|__|__|__|__| Nom : |__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|

Adresse : |__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|

Code postal : |__|__|__|__|__| Ville : |__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|

Règlement par chèque à l'ordre de Programmez ! I Disponible sur www.programmez.com

Offres pouvant s'arrêter à tout moment, sans préavis **PROG HS19**

Un générateur de sites statiques en PHP, une bonne idée ?

Yoan De Macedo

Je suis développeur web. Passionné par le web depuis mes débuts sur la toile avec mon modem 33.6k, j'en ai fait mon métier. J'ai l'occasion de travailler sur des projets très divers. Mon langage de prédilection est le PHP, depuis longtemps. Très sensible aux problèmes environnementaux, je suis conscient que mon métier doit évoluer. Le numérique doit se réinventer et changer de cap.

https://yoandemacedo.com

L'empreinte environnementale du numérique est conséquente. Le domaine est en pleine croissance. Nous ne pouvons pas continuer à utiliser des ressources comme si elles étaient inépuisables. Nos ordinateurs, nos smartphones, les datacenters mobilisent des quantités incroyables de métaux divers et contribuent à l'épuisement des ressources planétaires, au changement climatique. En développant des outils plus légers, mieux pensés, nous pouvons limiter l'usage du matériel et lui donner une vie bien plus longue. Dans cette optique, le site statique est intéressant. Mais, soyons honnêtes, nous ne sauverons pas la planète avec un site statique. Toutefois, ce n'est pas une raison pour ne pas y réfléchir.

Un site statique, qu'est-ce c'est ?

Aux premières lueurs du web, les sites étaient développés "à la main" via un éditeur de texte en utilisant le langage de présentation HTML. Puis, CSS pour la mise en forme et des langages de scripts comme JavaScript ont permis d'obtenir une certaine interactivité. Nous sommes ici typiquement sur un site statique.

Des logiciels sont ensuite apparus pour ne plus écrire directement en HTML, mais d'utiliser un éditeur graphique pour générer le code. Derrière, c'est toujours du code HTML brut en sortie qui était transféré sur un serveur web. Nous sommes toujours sur du statique.

Très vite, on a voulu faire des sites web avec des formulaires de contact, pouvoir afficher des informations disponibles dans des bases de données, discuter avec des services distants. Diverses technologies se sont succédées pour y parvenir.

Progressivement, nous en sommes arrivés à exploiter des CMS (Content management system) pour gérer les sites.

Une interface d'administration (backoffice), installée directement sur le serveur web et accessible via un simple navigateur permet de gérer son contenu. Celui-ci est généralement stocké dans une base de données. Puis, le moteur de l'application génère à la volée les pages HTML visibles par le client en utilisant un thème pour définir l'aspect graphique final. Cette manière de fonctionner est aujourd'hui relativement classique sur la toile et le CMS le plus célèbre est probablement WordPress.

Une base de données n'est pas toujours nécessaire. Pour utiliser moins de ressources et gagner en vélocité, des CMS "flat file" ont fait leur apparition. Les données sont stockées dans de simples fichiers. Je pense à Grav, à Kirby CMS.

Un autre mouvement s'accélère. Le retour au site statique. En effet, de très nombreux sites (vitrines, blogs) sont finalement mis à jour assez peu souvent. Le calcul systématique des pages n'est donc pas toujours nécessaire. Bien sûr, un CMS peut tirer profit du cache, mais l'ensemble restera plus gourmand que l'approche statique directe. Revenir au site statique à un moment où on recherche la sobriété est une idée naturelle.

Il est tout à fait possible d'écrire à nouveau les pages HTML à la main, mais d'autres méthodes ont aujourd'hui le vent en poupe (pour de bonnes raisons). Je pense notamment aux générateurs de sites statiques dont nous parlerons un peu plus loin.

Les avantages d'un site statique

Je parlais de l'impact environnemental du web. Un site statique est naturellement intéressant pour diminuer l'empreinte d'un site web. Mais, ce n'est pas le seul intérêt :

- Sobriété numérique : Moins gourmand signifie moins de ressources matérielles à employer. C'est bon pour l'environnement.
- Performance : C'est logique. De simples pages HTML, sans code à exécuter, c'est rapide à afficher.
- Sécurité : Pas de trou de sécurité dans le moteur applicatif. Peut-être à la rigueur dans le générateur, mais il n'est pas accessible « à l'extérieur ». Une fois le site en ligne, ce sont de simples pages HTML. Pas de backoffice à pirater non plus. Pas de mise à jour de CMS à faire.
- Facilité d'hébergement : De simples pages HTML, ça s'héberge partout.
- Sauvegarde : Vous n'avez qu'à sauvegarder des fichiers. Pas de base de données à sauvegarder et surtout à oublier de sauvegarder. C'est malheureux, mais ce n'est pas rare encore aujourd'hui (malgré tous les outils de sauvegarde existant et les hébergeurs qui proposent des services pour le faire) d'entendre des histoires de perte de base de données.

Bien entendu, un site web ayant besoin de mises à jour très fréquentes n'est probablement pas un bon candidat pour le statique. La majorité des sites web ne sont pas souvent mis à jour.

Réaliser un site statique

1 – À la main

Un simple éditeur de texte suffit pour créer une page HTML et par extension un site web.

Cette pratique n'est pas ringarde du tout et convient tout à fait lorsqu'on a besoin d'une seule page ou pour un site simple qui ne sera pratiquement jamais mis à jour. Toutefois, dès qu'on a besoin d'un peu plus, le travail peut être rapidement chronophage. C'est à ce moment-là qu'un outil est très utile.

2 - Avec un générateur de sites statiques

Un générateur de sites statiques va tout simplement nous permettre de générer les pages HTML nécessaires au bon fonctionnement du site web, mais en nous facilitant la vie. Je vais prendre l'exemple de mon blog.

Lorsque j'ajoute un nouvel article, j'ai besoin de réaliser les étapes suivantes :

- modifier la page d'accueil du blog
- déplacer le dernier article sur la page d'archive
- ajouter le nouvel article sur la page qui liste les billets (je la maintiens à 30 articles)
- mettre à jour le flux RSS
- mettre à jour le fichier sitemap

Si je devais effectuer ce travail à la main à chaque fois que j'ajoute un nouvel article de blog, clairement, ce serait pénible. Le risque d'erreur, d'oubli, est important et franchement ce n'est pas très intéressant. De plus, cela représente un temps non négligeable.

C'est le travail du générateur de sites statiques. Schématiquement, on a le contenu d'un côté, le thème (les templates) de l'autre qui contiennent la structure HTML, les règles, le CSS (etc.) et le générateur de sites statiques fusionne le tout pour en faire les pages web HTML finales.

Il existe de nombreux générateurs écrits dans différents langages. Je pense à Hugo, Pelican, Jekyll ...

Lorsque j'ai décidé de repasser au statique, j'ai décidé d'écrire le mien en PHP au lieu de me servir d'un outil existant.

Pourquoi développer son propre générateur ?

De nombreux générateurs sont disponibles et fonctionnent très bien. Toutefois, je souhaitais un outil qui corresponde à ma vision des choses et en PHP. Je n'ai pas trouvé l'outil idéal. C'était donc une occasion (et ça compte aussi pour beaucoup) de m'amuser un peu.

Fruga est un petit générateur de sites statiques avec les fonctionnalités de base dont j'ai besoin. Comme le logiciel libre compte beaucoup pour moi, j'en ai profité pour le diffuser sous licence GPL.

L'outil est fonctionnel, je génère régulièrement mon site avec. Il mérite encore d'être peaufiné. Je sais notamment que je dois notamment améliorer la gestion des erreurs. Peut-être aurez-vous aussi une bonne raison pour concevoir votre générateur maison.

Un générateur en PHP, pourquoi ?

J'utilise PHP pour le web, mais aussi pour des programmes CLI, des scripts système.

Clairement, au départ, PHP a été créé pour le web et dans un but "pratique". Si vous avez déjà pratiqué CGI (Common Gateway Interface), vous comprenez facilement le changement que PHP a apporté.

Lorsqu'on lit quelques interviews de Rasmus Lerdorf (le créateur de PHP), on voit vite qu'il l'a développé pour résoudre des problématiques liées au web et pour pouvoir y répondre rapidement. On le sent animé par ce côté "pratique".

C'est pour cette raison, je crois, qu'en plus d'être un langage, on pourrait presque considérer PHP nativement comme un framework pour le web.

Par exemple, c'est déjà un langage permettant le templating. Il intègre de nombreuses fonctionnalités nativement là où il faudrait empiler les dépendances avec d'autres langages (ou développer de grosses fonctionnalités soi-même). Sa conception pensée pour le web est idéale à mon goût pour développer rapidement et efficacement des outils dédiés à la toile.

PHP est-il adapté à la frugalité ?

PHP est un langage interprété, et donc, plus lent qu'un langage compilé. Il demande aussi davantage "d'énergie" pour tourner. Une étude a beaucoup circulé en fin d'année dernière. Elle comparait de nombreux langages à travers l'exécution d'un même algorithme. Finalement, plusieurs critères étaient pris en compte : énergie, temps d'exécution, mémoire utilisée. Évidemment, les langages interprétés n'étaient pas en tête et d'autres langages interprétés se trouvaient derrière PHP. Côté mémoire, il se situait même devant certains langages compilés. C'est une étude très intéressante, mais on ne peut clairement pas s'arrêter à la simple performance des langages. Sinon, on devrait utiliser exclusivement C, Rust et C++ pour tous les développements (et pourquoi pas java). Fin du débat.

A-t-on besoin massivement de bases de données, des API distantes ? L'infrastructure est-elle relativement simple ou faut-il de nombreuses machines ? J'oublie probablement d'autres externalités qu'il faudrait prendre aussi en compte.

Dans le cas d'un générateur de sites statiques, il tourne sur la machine de développement. Mon site web se génère en 0,25 seconde au moment où j'écris ces lignes. Ce serait probablement plus rapide en Rust, mais clairement, ce n'est pas là que l'économie sera la plus intéressante.

Créer un site statique avec Fruga

Pour le côté pratique de cet article, je vais utiliser mon générateur pour illustrer la création d'un site statique. L'approche sera plus ou moins similaire avec un autre générateur. Bien entendu, chacun aura des spécificités. Avec Fruga, j'ai décidé d'être au plus proche de PHP et d'éviter les surcouches. Je n'aurais donc pas besoin d'expliquer beaucoup de concepts par la suite.

PHP 8.x en ligne de commande est nécessaire pour faire tourner l'outil.

Ouvrez votre terminal et installez git si vous ne le possédez pas (vous pouvez aussi télécharger le code directement via le site de GitHub) :

```
git clone https://github.com/yoandm/fruga.git
cd fruga
php fruga.php generate example
```

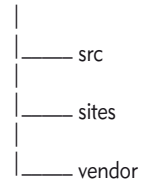
Vous avez généré votre premier site avec Fruga.

Le résultat est visible dans le répertoire sites/example/output.

Architecture générale

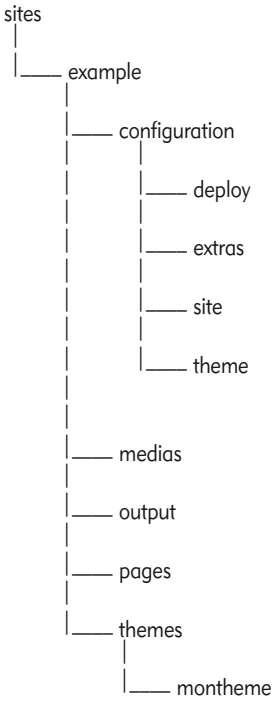
Fruga n'emploie pas de framework généraliste. On est sur une approche PHP "Vanilla". Pour l'instant, une seule bibliothèque externe est utilisée pour « parser » le format markdown. Il n'est pas impossible que je me serve de quelques bibliothèques plus tard pour du traitement d'images notamment.

```
fruga
|____ fruga.php
```



Le fichier fruga.php est le script permettant de lancer le générateur.
Le répertoire src contient le code source de l'application.
Le répertoire sites contient tous les sites à gérer.
Le répertoire vendor contient les bibliothèques externes.

Structure d'un site



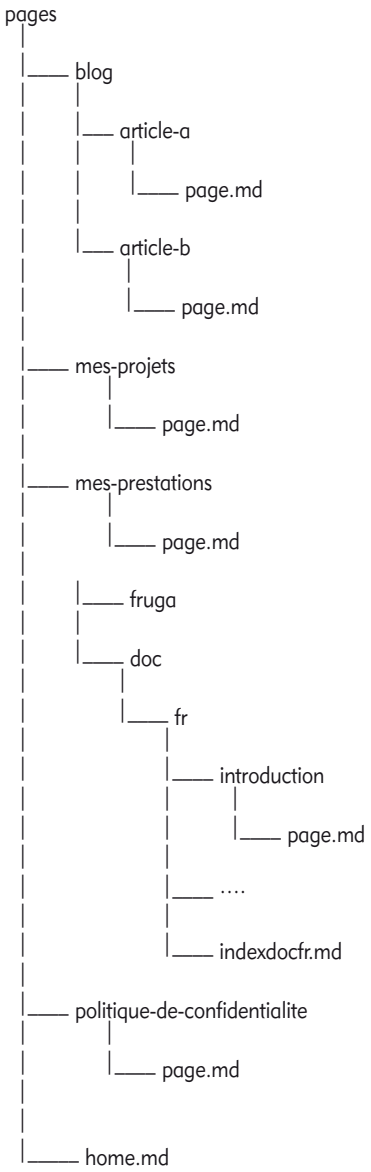
Le répertoire "sites" peut contenir plusieurs éléments. Fruga peut en effet gérer plusieurs sites dans une même installation. Dans l'exemple ci-dessus, on détaille le site "example".
On trouve à l'intérieur du répertoire "example" plusieurs sous-répertoires principaux :

Configuration
4 sous-répertoires se trouvent à l'intérieur :
- deploy permet de configurer le déploiement.
- extras permet de configurer les outils supplémentaires (flux RSS, sitemap ...). site permet de configurer le site web.
- theme permet de configurer le thème du site.
- Le formation json est utilisée pour les fichiers de configuration.

Medias : les éléments présents dans ce répertoire seront copiés dans la structure du site final généré.
On peut y stocker notamment des images, des vidéos si nécessaires (attention à la sobriété).
Output : Le site généré viendra prendre place dans ce répertoire.
Pages : Le contenu du site sera là.
Themes : On trouvera ici le thème du site. On peut avoir plusieurs thèmes pour un même site même si ça devrait être plutôt rare.

Gérer le contenu

Le contenu se trouve dans le répertoire sites/monsite/pages (vu précédemment).
C'est une collection de répertoires et de pages au format markdown.
Le contenu de la page d'accueil se situe à la racine du répertoire pages. On le nomme généralement home.md
Ensuite, on peut créer autant de répertoires et de sous-répertoires que l'on souhaite.
Je vous invite à étudier le contenu du site "example" (sites/example/pages).
Voici un morceau de la structure de mon site (un peu simplifiée) :



Les répertoires : Ce sont les répertoires qui vont permettre de créer la structure du site.
Par défaut, ils vont même définir automatiquement l'URL d'accès à la ressource (nous verrons qu'il est possible de surcharger ce comportement). Par exemple, l'accès à la ressource "commencer" au bout de la branche "fruga" se fera par l'URL contenant /fruga/doc/fr/commencer/

Il est aussi possible de les précéder d'un compteur. Par exemple "01.introduction", "02.commencer". Ce compteur n'apparaîtra pas dans l'URL, mais permettra de réaliser un classement manuel sur une page de listing par exemple.
Les fichiers markdown : Ces fichiers doivent comporter l'extension .md et se situent au sein des répertoires de contenus. On ne peut aujourd'hui n'avoir qu'un seul fichier .md par répertoire.
Vous avez probablement remarqué qu'ils n'ont pas toujours le même nom. On peut voir dans mon exemple home.md, page.md, indexdocfr.md.
Ce nom permettra de faire un lien avec le thème. En effet, il sera possible d'avoir des modèles de pages différents.
Par exemple, toutes les pages de mon blog, les pages d'informations de mon site, utilisent le format "page". La page affichant les différentes sections de la documentation fruga utilise le format "indexdocfr".
La page d'accueil utilise le format "home".
Passons maintenant au contenu de la page markdown.
La page d'accueil de mon site, home.md, commence comme ceci :

```
---
title: 'Yoan De Macedo'
---
```

La première section, délimitée par 3 tirets puis suivie d'un saut de ligne, c'est l'en-tête du fichier.
La seconde section, c'est le contenu de la page. Ce contenu pourra embarquer du markdown.
Différents mots clés peuvent être intégrés dans la première section. Ils pourront ensuite être exploités dans le thème.
Une liste préfinie existe (elle est amenée à évoluer) :

- title : titre de la page
- date : date de publication
- slug : "section" d'url
- published : true ou false (true si paramètre absent). true => publié, false => non publié

Nous verrons dans la section thème comment s'en servir.
Arrêtons-nous sur le mot clé "slug". Je vous expliquais précédemment comment l'URL d'accès à une page est générée via le chemin menant à celle-ci (à l'aide des noms de répertoires).
Il est possible de surcharger celle-ci et de ne pas forcément se baser sur le nom du répertoire.
Si dans la section en-tête de la page indexdocfr.md, j'ajoute :

```
slug: 'frenchy'
```

alors la page ne sera plus accessible via l'URL contenant /fruga/doc/fr/commencer/, mais /fruga/doc/frenchy/commencer/
Vous pouvez aussi ajouter d'autres mots clés comme vous le souhaitez et les exploiter dans le thème.

Les thèmes

Le thème permet de faire le lien entre le contenu et la page HTML générée (le résultat qui sera finalement affiché). Il est possible de préparer plusieurs thèmes pour un site puis de choisir celui qui sera utilisé.
C'est probablement la partie la plus technique dans la maniement de Fruga. Si vous décidez de réaliser votre propre thème alors vous avez besoin de maîtriser le HTML / CSS et d'avoir quelques notions de PHP.
Aucun "méta langage" comme smarty ou twig n'est utilisé. Les thèmes exploitent PHP pour le templating. Je sais que certains n'aiment pas cette approche.
Structure générale d'un thème :

```
montheme
   |
   |____ css
   |
   |____ templates
         |
         |_____ home.php
         |
         |_____ page.php
```

Le répertoire css contiendra vos feuilles de style. Le répertoire template contiendra tous les modèles de pages : si vous avez un fichier de contenu page.md quelque part alors il faudra un page.php, si vous avez un home.md alors on aura besoin d'un home.php. Ce template php définira comment s'affichera le contenu des pages du type correspondant. Vous pouvez tout à fait découper le header, le footer, faire des inclusions d'autres fichiers PHP, etc. Vous êtes libre.
Il est possible aussi d'ajouter d'autres répertoires à la racine du thème (un dossier js par exemple pour votre code JavaScript).

Configuration :
Pour choisir le thème qui sera utilisé par le site, il faudra éditer le fichier sites/monsite/configuration/theme/theme.json

```
{
  "name":"montheme"
}
```

Ce fichier de configuration s'étoffera peut-être dans le futur en fonction des thèmes.
Pour l'instant, on se sert simplement la propriété name qui définit le nom du thème mis en place.
Le système ira chercher ici le thème dans sites/monsite/themes/montheme
Conception du thème : Comme vous l'avez vu précédemment, il est possible de créer plusieurs modèles de pages. Rappelez-vous home.md, page.md, etc.
Imaginons que vous vous rendiez sur la page /blog/article-a et que le dossier sites/monsite/pages/blog/article-a contienne le fichier markdown page.md.
C'est le script page.php du thème qui sera chargé au moment de la génération de cette page.
Celui-ci recevra notamment la structure et le contenu du fichier markdown. Il pourra alors l'exploiter pour afficher les informations aux endroits souhaités.
Je ne vais pas entrer ici dans le HTML, mais il est nécessaire, bien sûr, de maîtriser la structure d'une page HTML pour créer un thème. Au sein de votre page HTML, vous allez pouvoir ajouter du code PHP permettant de récupérer les informations nécessaires auprès de Fruga.

=> Pour récupérer et afficher les informations d'en-tête d'un contenu :

```
<?php echo $page->get('nom_du_header'); ?>
```

Il suffit de remplacer nom_du_header. Par exemple title ou un header que vous auriez ajouté.
=> Pour récupérer et afficher la partie contenu :

```
<?php echo $page->get('content'); ?>
```

=> Pour récupérer et utiliser le lien d'une page :
Imaginons que dans votre menu, vous souhaitiez faire un lien vers la page /mes-prestations.

```
<a href="<?php echo $page->find('/mes-prestations')->get('link'); ?>">Mes prestations</a>
```

=> Lister des pages :
Vous pourriez (pour un blog par exemple) vouloir lister les différents articles avec un lien pour les visualiser. Je le fais aussi sur la page qui liste les différentes sections de la documentation Fruga.

```
$articles = $page->find('/fruga/doc/fr')->children->sort('pos', 'asc');
```

Ici, on récupère toutes les pages qui sont "enfants" de /fruga/doc/fr (Introduction, Commencer avec Fruga, ...).
On les classe via un ordre manuel (le petit numéro dont je vous ai déjà parlé devant le nom du répertoire). On aura donc un classement du plus petit au plus grand. Si je veux faire l'inverse, je me sers de desc à la place de asc.
Ensuite je n'ai plus qu'à "boucler".

```
<ul id="archiveList">
  <?php
    foreach($articles as $article){
  ?>
  <li><a href="<?php echo $article->get('link'); ?>"><?php echo $article->get('title'); ?></a> <a href="<?php echo $article->get('link'); ?>">[Lire]</a></li>
  <?php } ?>
</ul>
```

Générer un site

1 Configuration

Il est possible de créer plusieurs profils de génération. Pour cela, il suffit de créer un fichier .json dans le répertoire sites/monsite/configuration/site/
Vous pouvez créer le profil site.json et site_test.json par exemple. Il peut être pratique d'avoir un profil local pour des tests par exemple et un profil de production.
Voici un exemple de fichier :

```
{
  "name":"Yoan De Macedo",
  "url":"https://yoandemacedo.com",
  "relativeLinks":0,
  "cache":1,
  "extras":["rss","htaccess","sitemap"]
}
```

- name : c'est tout simplement le nom du site
- url : l'url de base du site
- relativeLinks : 0 ou 1 => 0, les liens auront la forme https://yoandemacedo.com/ sinon on aura des liens sous forme relative. Par exemple : ../lapage/
- cache : 0 ou 1 => utilisation du système de cache lors de la génération pour plus de performance. Couper le cache peut-être utile pour du debug
- extras : un tableau permettant d'activer des modules supplémentaires. Ici, rss, htaccess et sitemap sont activés.

2 Génération du site

Pour générer un site, il suffit de lancer la commande suivante :

```
php fruga.php generate monsite profil
```

Par exemple, pour générer le site "monsite" avec le profil "site_test" :

```
php fruga.php generate monsite site_test
```

Si aucun profil n'est spécifié alors Fruga utilisera le profil "site.json". Le site généré sera visible dans le répertoire sites/monsite/ output

3 Modules complémentaires

Fruga propose quelques modules complémentaires pour aller plus loin dans la génération d'un site web. La liste viendra probablement s'étoffer et les modules existants vont sûrement évoluer.
J'ai bien envie aussi d'ajouter un mécanisme de plugins pour que n'importe quel développeur puisse facilement créer son propre module complémentaire pour un besoin spécifique.
Pour activer un module complémentaire, il faudra renseigner l'information dans le profil de génération du site.
Par exemple dans le fichier sites/monsite/configuration/site/site.json

```
{
  "name":"Yoan De Macedo",
  "url":"https://yoandemacedo.com",
  "relativeLinks":0,
  "cache":1,
  "extras":["rss","htaccess","sitemap"]
}
```

Le tableau de la section extras liste les modules complémentaires à activer.
Les fichiers de configurations des modules complémentaires se situent dans sites/monsite/configuration/extras/

RSS

Si vous souhaitez un ou plusieurs fils RSS, le module RSS est fait pour vous.
Il faudra ajouter un rss.json dans sites/monsite/configuration/extras/

```
[
  {

    "pagePath":"/blog",
    "outputFilename":"rss.xml",
    "outputDir":"/blog",
    "title":"Le blog de Yoan De Macedo",
    "description":"Les articles du blog de Yoan De Macedo",
    "creator":"Yoan De Macedo"
```

```
    }
]
```

C'est un tableau de blocs de configuration puisque vous pouvez avoir plusieurs flux si vous le souhaitez :

- pagePath : le chemin pour lequel vous voulez générer un flux. Ici ce sera simplement pour la partie blog
- outputFilename : le nom du fichier pour le fux. On aura donc ici un /blog/rss.xml
- title : le titre du flux
- description : la description du flux
- creator : le créateur du flux

Sitemap

Ce module va générer un sitemap.xml à la racine de l'hébergement.

Le fichier de configuration sera sites/monsite/configuration/extras/sitemap.json

```
{

}
```

Pour l'instant, il est vide. Le module viendra peut-être s'étoffer dans le futur.

Htaccess

Si vous voulez générer un fichier .htaccess avec des directives, c'est possible avec ce module. Il faudra ajouter un htaccess.json dans sites/monsite/configuration/extras/

```
{
  "removeIndexhtml":1,
  "extraLines": [
    "RewriteCond %{HTTP:X-Forwarded-Proto} !https",
    "RewriteRule ^ https://%{HTTP_HOST}%{REQUEST_URI}
[L,R=301]"
  ]
}
```

removeIndexhtml : 1 si vous souhaitez empêcher qu'on accède au contenu via /blog/article-a/index.html par exemple

extraLines : un tableau pour permettant d'ajouter toutes les directives htaccess que vous souhaitez

Déployer un site

Il est possible de créer plusieurs profils de déploiement. On peut imaginer un déploiement local et un déploiement distant par exemple.

Aujourd'hui, 3 types de connecteurs de déploiement sont disponibles : local (système de fichier local), FTP, SFTP.

Pour cela, il suffit de créer un fichier .json dans le répertoire sites/monsite/configuration/deploy/

Voici un exemple de fichier local.json :

```
{
  "type" : "local",
  "config":{
    "path" : "/Users/yoan/Documents/htdocs/frugatest"
  }

}
```

Puis un exemple de fichier de déploiement distant en FTP monhebergeur.json :

```
{
  "type" : "ftp",
  "config":{
    "host" : "ftphost.com",
    "port" : 21,
    "ssl" : 0,
    "login" : "login",
    "password" : "",
    "path" : "",
    "passive" : 1
  }
}
```

On remarque ici le type "ftp" puis la configuration permettant de se connecter :

- host : l'url de connexion
- port : le port de connexion
- ssl : 1 si chiffré, 0 sinon
- login : le login de connexion
- password : le mot de passe de connexion. Si vous préférez le laisser vide pour qu'il ne figure pas en clair dans un fichier, vous pouvez. Il vous sera demandé lors de déploiement
- path : chemin spécifique si vous ne publiez pas le site à la racine
- passive : 1 si mode passif, 0 sinon

Pour du SFTP, on aurait :

```
{
  "type" : "sftp",
  "config":{
    "host" : "sftphost.com",
    "port" : 22,
    "login" : "login",
    "password" : "",
    "path" : "",
  }
}
```

Pour déployer un site, il suffit de lancer la commande suivante :

```
php fruga.php deploy monsite profil
```

Par exemple, pour déployer le site "monsite" avec le profil "monhebergeur" :

```
php fruga.php deploy monsite monhebergeur
```

Conclusion

Générer un site statique est relativement simple. L'idée étant aussi de reprendre la main sur ce qu'on produit et la façon dont on le produit, créer son propre générateur est une approche intéressante. Bien entendu, rien n'empêche de préférer un générateur existant et même d'y contribuer si on le souhaite. J'ai opté ici pour une approche minimaliste en gérant le contenu directement via les fichiers dédiés. Il est tout à fait possible, si on le souhaite, de brancher un CMS en mode "headless" pour gérer les contenus à l'aide d'un outil plus graphique.

Comment choisir son hébergeur en fonction de son impact environnemental ?

On le sait, le cloud a révolutionné le déploiement d'applications et d'infrastructures : en quelques clics, on provisionne un serveur, un cluster Kubernetes ou un service managé. Cette facilité masque pourtant un aspect préoccupant : l'impact environnemental réel de nos choix d'hébergement.

Nicolas Leseignoux

CTO et cofondateur de Green4Cloud un cloud souverain qui a la volonté de devenir le 1er cloud des territoires à énergie positive. Diplômé d'un Master en Sécurité / sûreté de l'information des systèmes informatiques, il a administré un parc réseaux et sécurité de 300 bâtiments de l'Université de Bordeaux avant de cofonder Green4Cloud pour répondre aux enjeux environnementaux du numérique.

Entre la profusion d'options (stockage, compute, services gérés) et l'illusion de ressources infinies, il devient facile d'ignorer la consommation énergétique et l'empreinte carbone que ces solutions impliquent. Pourtant ces cloud se basent sur des datacenters consommant toujours plus, ils représentent aujourd'hui 3,37% de la consommation électrique nationale. Et cette part pourrait doubler d'ici 2035, atteignant 6,74%, avec une puissance IT installée estimée à 2,5 MW. Aujourd'hui, le choix d'un hébergeur ne se résume plus à comparer coûts et performances : il faut désormais prendre en compte :

- L'efficacité énergétique notamment via le PUE (Power Usage Effectiveness)
- La provenance de l'électricité,
- La stratégie de mutualisation des ressources
- La transparence des engagements écologiques (et leur différence avec le greenwashing).

Et même avec tout cela, il reste difficile de s'y retrouver, c'est ce que nous allons explorer dans cet article.

Solutions d'hébergement

Tour d'horizon des solutions d'hébergement

Avant d'entrer dans les critères environnementaux, il convient de rappeler rapidement les principales offres d'hébergement disponibles :

- Le bare metal (serveur dédié): un serveur physique entier, pour un contrôle total (virtualisation, containerisation, choix des OS, etc.)
- Le VPS (Virtual Private Server): une machine virtuelle avec accès administrateur, hébergée sur un hyperviseur partagé.
- Les solutions de containerisation et d'orchestration (PaaS) : comme Kubernetes managé, facilitant le déploiement et la scalabilité d'applications en containers.
- L'hébergement web mutualisé : solution plutôt d'entrée de gamme pour des sites à faible trafic, avec peu de personnalisation possible.
- Serverless: modèle à la demande (FaaS), où des fonctions sont déclenchées par des événements (appels API, cron, interactions utilisateurs…).

Quelle solution est la plus efficiente au niveau énergétique ?

Si l'on considère uniquement le critère de la sobriété énergétique, les services mutualisés ou managés sont les plus vertueux. L'hébergeur y a tout intérêt : plus il mutualise ses infrastructures, plus il les rentabilise économiquement, tout en optimisant leur usage énergétique.

A l'inverse, les solutions VPS présentent un moindre niveau de mutualisation avec une surcharge liée à la virtualisation (overhead). Le bare-metal est en bas de classement car il mobilise des ressources pour un seul client mais bien sûr cela dépend de la charge sur le serveur.

Le PUE : un indicateur clé pour comparer les hébergeurs ? (spoiler : pas vraiment)

Comprendre le PUE

Le PUE (Power Usage Effectiveness) est un indicateur couramment utilisé pour mesurer l'efficacité énergétique d'un datacenter. Il se calcule comme suit :

$$PUE = \frac{Energie\ totale\ consommée\ par\ \leq\ datacenter}{Energie\ utilisée\ par\ les\ équipements\ \_x0000\_}$$

Un PUE de 1,0 signifierait qu'aucune énergie n'est perdue en refroidissement, conversion électrique, éclairage, etc. Ce scénario est évidemment idéal et impossible dans la réalité.

Exemple simple :

- Votre charge IT consomme 90kWh
- L'onduleur a un rendement de 90%, donc il faut 100KWh pour fournir 90KWh utile (100 / 0,9)
- Le refroidissement consomme 20KWh et les autres systèmes (éclairage, sécurité, ...) 5KWh.
- Vous consommez donc 125KWh pour 90KWh d'utile, soit un PUE de 1,38 (125/90)

Quelle est la moyenne du PUE du secteur ?

Selon le Global Data Center Survey 2024 de l'Uptime Institute, la moyenne mondiale du PUE est actuellement de 1,56, avec près de 50 % des datacenters ayant plus de 11 ans.

Un PUE ≤ 1,35 est déjà une bonne performance, souvent atteinte via des techniques comme le free cooling (refroidissement par air extérieur). Les meilleurs PUE (≤ 1,2) sont généralement obtenus grâce à des technologies de refroidissement avancées :

- DLC (Direct Liquid Cooling) : refroidissement liquide ciblé sur les composants (ex : CPU)
- Immersion cooling : les serveurs sont immergés dans un fluide diélectrique pour dissiper la chaleur plus efficacement

Le PUE, un indice imparfait pour comparer les hébergeurs

Le PUE a une vraie valeur en interne pour un hébergeur : il permet de suivre ses progrès (ex : amélioration du rendement d'un onduleur), mais il est moins pertinent pour un client final cherchant à comparer plusieurs fournisseurs.

Il peut varier fortement:

- Avec la localisation géographique : un datacenter en Norvège aura naturellement un meilleur PUE qu'un autre en Espagne à cause d'un refroidissement passif facilité
- Avec le taux d'occupation : un site à 20 % d'occupation aura

un PUE artificiellement élevé (le système de refroidissement fonctionne peu importe l'usage réel),

- Avec la redondance et le niveau de service : un datacenter Tier IV, très résilient, utilise plus d'équipements en doublon, ce qui augmente mécaniquement le PUE,
- Ancienneté du datacenter : les nouveaux datacenters intègrent des systèmes plus performants (refroidissement, distribution électrique…), ce qui tire le PUE vers le bas.

Où trouver le PUE de mon hébergeur ?

Le PUE de votre hébergeur n'est pas disponible dans votre interface d'administration. Il faut aller chercher cette information dans les rapports périodiques RSE (Responsabilité Sociétale des Entreprises), les pages "engagements environnementaux" ou "Politique Environnementale", ou parfois dans les livres blancs techniques de l'hébergeur. Il s'agit souvent d'une moyenne annuelle, non d'un suivi en temps réel, et les données ne sont pas toujours vérifiables ou comparables entre fournisseurs.

En conclusion

Le PUE est un indicateur utile mais à relativiser. Il renseigne davantage sur la gestion interne du datacenter que sur l'impact réel de votre propre consommation cloud. En revanche un hébergeur qui communique régulièrement sur son PUE, explique ses marges de progression et détaille ses choix de refroidissement et d'énergie démontre une maturité sur son bilan environnemental et permet de bâtir une relation de confiance.

Les énergies renouvelables au service des data-centers

"C'est bien gentil ton PUE mais si mon hébergeur utilise de l'énergie renouvelable ça suffit non ?"

Eh bien… pas si vite. Comme souvent, la réponse est plus complexe. Quand un hébergeur affirme utiliser de l'électricité verte, il peut recourir à plusieurs méthodes très différentes en termes d'impact réel.

Trois approches pour l'électricité bas carbone

- Les Garanties d'Origine (GO) : c'est le cas le plus courant : pour chaque 1MWh consommé, 1MWh d'énergie renouvelable a été injecté quelque part dans le réseau européen. C'est un mécanisme comptable, pas physique : le datacenter consomme le mix du pays, mais compense symboliquement. Cela veut donc dire que le 1MWh injecté est potentiellement injecté en Norvège sans que cela ait un effet concret sur la production locale.
- Le contrat PPA (Power Purchase Agreement): l'hébergeur finance la production d'électricité renouvelable via un contrat à long terme avec un producteur (et non pas un fournisseur d'énergie comme les GO). Même sans lien physique entre les deux, ce contrat garantit un prix fixe du MWh sur plusieurs années et favorise l'ajout de capacité renouvelable au niveau du réseau.
- La production d'énergie bas carbone sur site : par exemple en installant des panneaux solaires ou des éoliennes, sur site ou ailleurs, pour alimenter le datacenter. Cette autoconsommation est partielle, mais reste rare et techniquement complexe, surtout en zone urbaine.

Pourquoi l'énergie verte est importante pour les hébergeurs

Le "100% énergie renouvelable" est souvent présenté comme un argument écologique pour les clients, mais en réalité, les bénéfices de ces démarches concernent d'abord l'hébergeur lui-même car l'achat de GO ou la signature d'un PPA permet à l'hébergeur de réduire son propre scope 2 dans son bilan carbone réglementaire. Cela n'aura d'impact pour vous que si l'hébergeur vous fournit un bilan carbone précis pour la consommation de vos services que vous pourrez ensuite utiliser pour votre propre bilan carbone.

Le bilan carbone : un indicateur de plus, mais pas un outil de comparaison fiable

Le bilan carbone est une méthode de calcul des émissions de gaz à effet de serre (GES). Il permet à une organisation de comptabiliser l'ensemble des émissions de gaz comme le CO2, le méthane, ou encore le protoxyde d'azote, générées par ses activités. Il s'agit avant tout d'un outil de pilotage interne : il permet de suivre l'évolution des émissions, de fixer des objectifs de réduction, et d'identifier les postes les plus émetteurs.

Les trois "scopes" du bilan carbone

Le bilan carbone est structuré en trois périmètres (ou scopes) définis par le GHG Protocol (Greenhouse Gas Protocol), largement utilisé en entreprise :

- Le Scope 1 : qui correspond aux émissions directes émises par l'organisation, par exemple les émissions des groupes électrogènes des datacenters, les fuites de gaz frigorigènes des systèmes de refroidissement ou les véhicules de l'hébergeur,
- Le Scope 2 qui correspond aux émissions indirectes liées à l'énergie : c'est souvent le plus gros poste de dépense en termes de bilan carbone pour un hébergeur
- Le Scope 3 : qui correspond aux autres émissions indirectes, c'est souvent le scope le plus vaste et donc le plus négligé car il peut être difficile d'avoir accès à toutes les informations (fabrication et transport d'équipements IT, construction du datacenter, fin de vie des équipements, trajets domicile travail employés, ...). Pour un service de type cloud managé ou IaaS, le scope 3 va inclure la fabrication, le transport et la fin de vie des serveurs, ce qui en fait le poste principal d'émissions, en revanche, dans une offre de housing ou colocation, le client fournit ses propres machines : ce poste sort alors du périmètre de l'hébergeur.

Peut-on comparer le bilan carbone des hébergeurs ?

La réponse est non car même si deux hébergeurs publient un "bilan carbone", leurs périmètres peuvent être très différents (certains n'incluent pas le scope 3), les hypothèses de calcul varient : durée de vie des serveurs, facteurs d'émissions utilisés, etc. Et certains incluent la compensation carbone, d'autres non.

En résumé, nous avons donc un autre facteur qui sert en interne à l'hébergeur pour qu'il puisse réaliser des améliorations mais cela ne peut pas nous servir pour comparer 2 hébergeurs différents.

Et les calculettes des fournisseurs ?

Certains fournisseurs (AWS, Azure, Google Cloud…) proposent des outils de mesure de l'empreinte carbone de vos propres services, ce sont des outils intéressants, mais ils ne sont pas standardisés entre fournisseurs, ne permettent pas de comparer deux clouds entre eux de manière fiable, et ne couvrent pas toujours les scopes 1 et 3.

Pour aller plus loin, l'association Boavizta propose des outils gratuits, ouverts et indépendants pour évaluer, piloter et réduire l'impact environnemental du numérique.

Parmi eux Datavizta permet :

- D'estimer les émissions GES liées à un service cloud (AWS,GCP,Azure),
- De modéliser l'impact de la fabrication et de l'utilisation d'un serveur suivant plusieurs critères (GES, Energie primaire, ressources abiotiques),
- D'être précis sur les caractéristiques du serveur (CPU, RAM, SSD)
- Et même d'utiliser une API pour intégrer ces résultats dans vos propres outils d'audit ou de pilotage interne.

Durée de vie du matériel

Lorsqu'on parle d'impact environnemental d'un datacenter ou d'un service cloud, on pense spontanément à la consommation d'électricité. Pourtant, un poste majeur d'émissions reste souvent hors radar : la fabrication du matériel IT (serveurs, baies, stockage…). Et ce poste dépend directement d'un paramètre : la durée de vie du matériel.

Pourquoi la fabrication pèse autant dans le bilan carbone

Produire un serveur, c'est extraire, transformer et transporter des matériaux rares et énergivores (aluminium, cuivre, silicium, terres rares…), consommer beaucoup d'énergie en phase d'assemblage, et générer des déchets électroniques en fin de vie. Suivant l'ADEME, la phase de fabrication représente jusqu'à 50 % (voire plus) de l'empreinte carbone totale d'un serveur sur l'ensemble de son cycle de vie. Autrement dit : le plus gros impact carbone d'un serveur peut se produire… avant même qu'il soit allumé.

Le problème de l'obsolescence

Les hébergeurs remplacent souvent leurs serveurs au bout de 3 à 5 ans :

- Pour garantir des performances optimales,
- Pour profiter de gains énergétiques liés à des composants plus récents,
- Ou parce que leur modèle économique repose sur un renouvellement rapide du parc.

Mais d'un point de vue environnemental, le "gagnant" n'est pas si évident car dans certains cas, la fabrication aura plus d'impact que l'utilisation elle-même ce qui en devient une aberration.

Comment agir ?

De plus en plus d'hébergeurs proposent des offres vertes basées sur du matériel reconditionné ou allongé en durée de vie, cela permet à nous, clients, de pouvoir héberger nos workloads peu gourmands (ex : tests, CI, jobs batch, services peu sollicités) sur ces infrastructures si l'on a vraiment besoin de services hébergés en bare-metal. Cela permet de plus de confirmer l'intérêt des clients pour ce genre de gammes et donc que l'hébergeur prenne en compte ce besoins et achète moins souvent du matériel neuf.

Checklist

Cette checklist se veut une aide à la décision, et convient plutôt aux acteurs déjà établis plutôt que pour des petits hébergeurs locaux. Encore une fois la notion de relation de confiance est importante et les hébergeurs locaux peuvent aussi mettre en place des actions sans qu'elles soient publiées sur leur site, il est donc intéressant de les questionner sur le sujet pour vérifier leur maturité et éclairer votre choix.

Infrastructure et énergie

- Le datacenter est situé dans un pays à mix énergétique bas carbone (ex : France, Norvège, Suède…)
- L'hébergeur publie des informations sur l'électricité utilisée et précise s'il utilise des Garanties d'Origine (GO), des contrat PPA, ou de la production d'énergie sur site
- Il communique un PUE (Power Usage Effectiveness) clair et mis à jour régulièrement

Mutualisation & Services

- Des offres sur du matériel reconditionné ou "low impact" sont proposées
- Il est possible d'opter pour des services managés (containers, serverless…) à empreinte plus faible
- Le datacenter ou cloud provider propose une localisation par région (choix éco-responsable possible)

Transparence & Mesures

- L'hébergeur publie régulièrement un bilan carbone clair et précise la durée de vie en moyenne de leur matériel IT
- L'hébergeur publie le taux de matériel reconditionné utilisé dans ses datacenters
- Il communique sur le recyclage, réemploi ou reconditionnement de ses serveurs
- Il met à disposition un outil de suivi de votre empreinte carbone (ou à défaut, fournit des métriques brutes)

Sources

https://alliancegreenit.org/le-pue-definition-et-mesure
https://datacenter.uptimeinstitute.com/rs/711-RIA-145/images/2024.GlobalDataCenterSurvey.Report.pdf?version=0
https://www.carbone4.com/analyse-empreinte-carbone-du-cloud#footnote-20
https://www.carbone4.com/files/Publication_Electricite_verte.pdf
https://blog-isige.minesparis.psl.eu/2024/07/05/corporate-power-purchase-agreement-ppa-in-europe/
https://www.maxduboisconsultant.fr/wp-content/uploads/2024/10/Etude-Efficacite-energetique-dans-les-Datacenters-29102024.pdf
https://datacenters.lbl.gov/sites/default/files/WP49-PUE%20A%20Comprehensive%20Examination%20of%20the%20Metric_v6.pdf
https://boavizta.org/media/pages/blog/evaluation-empreinte-environnementale-cloud-public/173154e3cf-1714645915/boavizta-evaluation-de-l-empreinte-environnementale-du-cloud-20240502.pdf
https://www.sia-partners.com/system/files/document_download/file/2024-02/L%20Essentiel%20des%20PPA%20-%20Sia%20Partners.pdf
https://www.maxduboisconsultant.fr/wp-content/uploads/2024/10/Etude-Efficacite-energetique-dans-les-Datacenters-29102024.pdf

L'hébergement de votre site écoconçu

Une fois votre site web écoconçu en réfléchissant aux besoins de vos utilisateurs, optimisé vos assets, paramétré le cache… il reste un dernier choix important : celui de l'hébergeur ! D'un fournisseur à un autre, l'empreinte environnemental peut varier pour un même service numérique. Le référentiel général d'écoconception de services numérique (RGESN) dans sa dernière version possède une thématique à part entière avec 3 critères prioritaires, 6 critères recommandés et 1 critère modéré. Voici un tour d'horizon des critères à prendre en compte pour réussir la dernière étape de l'écoconception de votre site web avant la phase d'utilisation.

Hervé Boisgontier

Formateur en développement informatique au sein d'ENI École Informatique à Nantes depuis 15 ans.

Spécialisé en Numérique Responsable (Écoconception de services numériques, développement accessible conforme au RGAA et audit d'accessibilité numérique).

Auteurs de livres aux éditions ENI (Green IT et Accessibilité, SQL Server 2022, Algorithmique et WordPress, green IT et accessibilité).

Le critère 8.1 du RGESN est le plus général : il vise à vérifier les engagements de l'hébergeur en matière de diminution de son empreinte environnemental (émission de gaz à effet de serre, consommation en énergie, en eau et en ressources non renouvelables).

L'efficience énergétique

L'un des premiers éléments évoqués pour un hébergeur vert est son efficience énergétique. Celle-ci est mesurée par le **PUE** (Power Usage Effectiveness ou Indicateur d'efficience énergétique en français). Le PUE calcule le rapport entre la consommation totale du centre de données par rapport à celle des équipements informatiques.

$$PUE = \frac{Consommation\ totale\ du\ centre\ de\ données}{Consommation\ des\ équipements\ informatiques}$$

L'idéal est d'avoir un PUE valant 1,0. Plus sa valeur est grande, moins bonne est l'efficacité énergétique du centre de données. Cet indicateur permet de mettre en évidence l'énergie utilisée par la climatisation, les onduleurs, la filtration de l'air, la conversion du courant… À l'heure actuelle, la valeur moyenne des PUE des centres de données est d'environ 1,6. Les hébergeurs ne publient pas systématiquement leur PUE. Généralement, s'ils ne le font pas, ce n'est pas bon signe. Cela veut souvent dire qu'ils utilisent encore beaucoup d'énergie en utilisant la climatisation, très énergivore, plutôt que d'utiliser une source naturelle de rafraîchissement, *free-cooling* à air ou à eau.

Voici quelques exemples d'hébergeurs publiant leur PUE :

Hébergeur	PUE
Neutral Hosting	1,06
Infomaniak	1,06
Ex2	1,09 à 1,29
DRI	1,15
PlanetHoster	1,1 à 1,2
OVHcloud	1,29
Hostinger	1,2 à 1,7
Scaleway	1,37

Ces valeurs peuvent être obtenue de deux manières : soit en effectuant un calcul théorique lors de la conception du datacenter (by design) soit à l'aide de valeurs réelles mesurées. Pour valider le critère 8.3 du RGESN, il est nécessaire que le PUE de votre hébergeur soit inférieur à 1,5 avec des valeurs réelles ou inférieur à 1,3 by design si le datacenter a moins de deux ans et donc pas encore de données réelles. Il existe la certification ISO 50 001 permettant d'attester de la performance énergétique. Par exemple, le français **OVHcloud** et le suisse **Infomaniak** possèdent cette certification.

L'efficience en eau

Certains systèmes de refroidissement peuvent être très gourmand en eau. Le WUE (Water-Use Efficiency ou Indicateur d'efficience de l'usage de l'eau en français) permet de comparer la quantité d'eau nécessaire par rapport à la puissance électrique.

$$WUE = \frac{Consommation\ en\ eau\ (L)}{Consommation\ électrique\ (kWh)}$$

Le critère 8.4 du RGESN impose d'avoir un WUE inférieur à 0,4L/kWh sur des données réelles mesurées selon un standard international tel que l'ISO/IEC 30 134. Il est tout fois précisé qu'un WUE élevé dans un pays ne connaissant pas de stress hydrique est moins problématique que dans un pays ayant une tension sur l'eau potable. Malheureusement, à l'heure actuelle, les hébergeurs ne calculent pas ou ne communiquent pas leur WUE.

Les sources d'approvisionnement électrique

La production d'électricité a une empreinte environnementale forte. Mais celle-ci varie très fortement suivant son origine. Même parmi les énergies renouvelables, les impacts sont différents. **Figure 1**

Un certain nombre d'hébergeurs indiquent qu'ils utilisent des énergies vertes. À nouveau, les informations sont plus ou moins précises. Par exemple, le canadien **Planet Hoster** indique utiliser 100% d'électricité renouvelable alors que le français **Neutral Hosting** affiche électricité verte sans plus de précisions. Il est important que ces affirmations soient prouvées à l'aide de certificats.

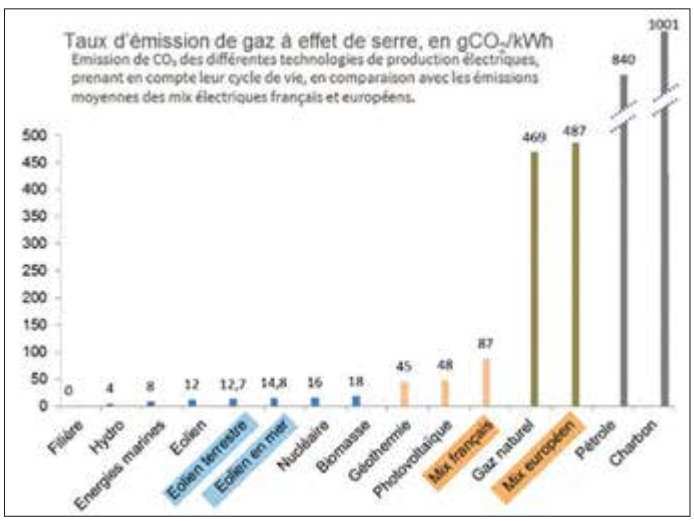


Figure 1
Source : ADEME

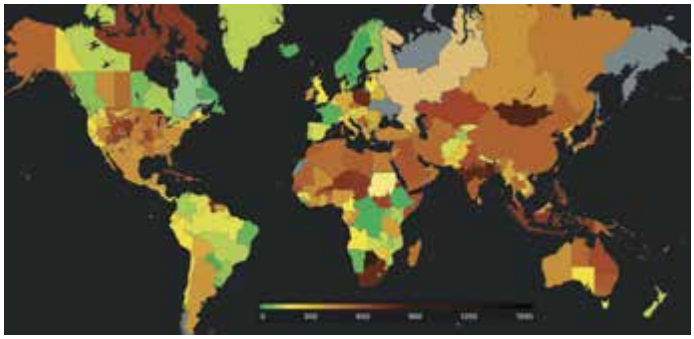


Figure 2
Source :
https://app.electricitymaps.com

Le critère 8.5 du RGESN attend de l'hébergeur une transparence sur son approvisionnement et les justificatifs (PPA ou certificat d'origine de l'électricité).

La localisation

Il est judicieux pour gagner en performances et pour moins solliciter le réseau de choisir son hébergeur proche de la localisation de la majorité de ses visiteurs. Par exemple, un site internet d'une association locale nantaise est principalement visité par des internautes de la région de Nantes. Il est donc plus judicieux d'héberger son site dans un centre de données nantais. Au contraire, pour une entreprise ayant des visiteurs venant de l'ensemble de l'Europe, il est plus judicieux de choisir un hébergeur ayant son centre de données au cœur de l'Europe. Mais il est aussi intéressant de s'intéresser au mix énergétique du pays dans lequel se trouve le datacenter. En effet, en France le mix énergétique est peu carboné mais ce n'est pas le cas de tous les pays. **Figure 2**

Pour valider le critère 8.6 du RGESN, il est nécessaire d'héberger ses données dans un pays ayant une empreinte carbone inférieure à 100 gCO eq/kWh. Pour la France cela est validé avec 33 gCO eq/kWh pour l'année 2024. C'est également le cas pour la Suisse, la Norvège, la Suède, la Finlande et l'Islande pour les pays européens.

La récupération de la chaleur dégagée

Les serveurs produisent de la chaleur. Un certain nombre de centres de données valorisent cette chaleur. Par exemple, **Neutral Hosting** s'en sert pour chauffer l'eau de bâtiments résidentiels. Les serveurs sont plongés dans un bain huile et via un échangeur thermique la chaleur générée par les serveurs est récupérée et transférée dans un ballon d'eau chaude.

Le critère 8.7 du RGESN a pour objectif de minimiser la chaleur perdue. Une première manière de satisfaire à ce critère est mettre en place une stratégie de récupération de la chaleur pour qu'elle soit valorisée. Une autre manière est d'avoir peu de chaleur générée avec un PUE < 1,3.

La politique d'achat et de renouvellement du matériel

Le choix du matériel peut avoir des impacts importants. Certains serveurs supportent des températures plus élevées et par conséquent requièrent un refroidissement moins important, donc moins énergivore. Le choix de processeurs de faible voltage permet d'économiser de l'énergie. Cela peut sembler paradoxal, mais, mathématiquement, cela dégrade le score du PUE. C'est pourquoi il est important de prendre en compte un ensemble de critères. Beaucoup d'impacts écologiques proviennent de la production et de la destruction du matériel. Il est donc important de connaître les engagements de son hébergeur en la matière. Il est difficile de trouver des engagements écrits en la matière. **Neutral Hosting** indique qu'il utilise la technique d'immersion liquide de ces serveurs afin d'augmenter leur durée de vie, en plus de réduire le bruit généré et les refroidir efficacement. **Infomaniak** indique qu'il s'engage à conserver ses serveurs pour une durée minimale de cinq ans et pouvant aller jusqu'à 15 ans. Il est également important de connaître les engagements de votre hébergeur en matière de gestion de ses Déchets d'Équipements Électriques et Électroniques (**DEEE**).

Le critère 8.2 demande que l'hébergeur ait pris des engagements sur la gestion de ses équipements sans toutefois indiquer une durée minimale à atteindre.

Le respect des données

Suivant le pays hébergeant vos données, la législation en vigueur n'est pas la même. Dans l'Union européenne, c'est le Règlement Général sur la Protection des Données (**RGPD**) qui prévaut et en Suisse, c'est la Loi fédérale sur la Protection des Données (**LPD**). Mais aux États-Unis, ou avec une entreprise américaine (même en dehors du territoire américain), ce sont le **Patriot Act** (Agences gouvernementales pouvant agir en secret) et le **Cloud Act** (Autorités bénéficiant d'un mandat) qui s'appliquent. Les données personnelles sont alors moins bien protégées.

La compensation carbone

Les hébergeurs les plus vertueux compensent leurs émissions de CO_2. D'une part, il faut être conscient que cela ne réduit en rien les **GES** (Gaz à Effet de Serre) émis par le service numérique, mais vient seulement le contrebalancer en finançant la restauration de systèmes de captation naturelle (zones humides ou boisées) ou la réduction d'émissions de CO_2 d'autres acteurs. D'autre part, il faut alors être attentif à l'inclusion de l'ensemble des émissions de l'entreprise (fabrication, acheminement et recyclage du matériel, déplacements professionnels des collaborateurs, production d'énergie…). À titre d'exemple, **Infomaniak** compense à 200% ses émissions de CO_2.

Les hébergeurs indépendants

Il n'y a pas que la solution de confier votre site à un hébergeur pour que celui-ci installe votre site dans son centre de données. Il est possible d'héberger son site sur une simple machine ayant une bonne connexion à internet.

Le collectif **CHATONS** (Collectif des Hébergeurs Alternatifs, Transparents, Ouverts, Neutres et Solidaires, https://www.chatons.org/node/1) propose des solutions alternatives aux géants du web. Entre autres, ils proposent des hébergements gratuits pour des sites d'associations, d'initiatives citoyennes, d'activités artistiques, promouvant les solutions libres…

Enfin, certaines personnes ont même un site internet fonctionnant sur un Raspberry PI alimenté par un panneau solaire ! C'est par exemple le cas de **Low Techmagazine** (https://solar.lowtechmagazine.com). Leur site est à Barcelone, il profite donc d'un bon ensoleillement, et, si les batteries alimentées par les panneaux solaires sont épuisées, le site n'est plus disponible ! Mais est-il nécessaire que le site soit accessible à tout moment du jour et de la nuit ?

Eco-concevoir une application web de bout en bout

Benjamin Morali
Développeur web full-stack, membre de la communauté GreenIT, contributeur aux 115 bonnes pratiques d'éco-conception web, formateur en éco-conception

Le numérique génère de nombreux impacts environnementaux : gaz à effets de serre, épuisement des ressources en eau et en métaux. Ces impacts ne cessent d'augmenter depuis les débuts de l'informatique, et en tant que concepteurs de services numériques, notre devoir est d'essayer de les réduire. Pour cela, on peut éco-concevoir à chaque phase du cycle de vie d'un produit. D'abord au moment de la conception, où on aura le plus d'impact, puis dans la partie technique, en s'intéressant au front-end, au back-end, ainsi qu'à l'infrastructure qui héberge le produit numérique.

Le numérique est souvent présenté comme immatériel, mais il n'en est rien. Il repose sur de nombreuses infrastructures, et a un impact environnemental important et grandissant. On peut l'analyser en le divisant en trois tiers [1] : les centres de données (ou data centers), le réseau (fibre optique, antennes 5G, etc.), et les terminaux (smartphones, ordinateurs, objets connectés, etc.). On pourrait penser que la majorité des impacts proviennent des centres de données, mais c'est bien les terminaux qui concentrent l'essentiel de la pollution du numérique. En cause, leur nombre (plusieurs dizaines de milliards), à comparer avec les serveurs de centres de données, qui se comptent en dizaines de millions. Si on regarde de plus près les terminaux, leur fabrication pollue plus que leur utilisation, et de loin (environ 80%). Premier constat : avant même d'avoir écrit la première ligne de code, la plus grande partie de la pollution a déjà été générée. Cependant, si on s'intéresse aux motifs de renouvellement des terminaux, on se rend compte que le logiciel a une influence non négligeable. En effet, une majorité d'utilisateurs remplacent leur smartphone pour des problèmes de lenteurs et d'incompatibilité, sur lesquels les développeurs peuvent – et doivent – agir.

Par ailleurs, les impacts environnementaux (gaz à effets de serre, utilisation d'eau, érosion de la biodiversité, épuisement des ressources abiotiques) sont en augmentation pour plusieurs raisons. D'abord, le nombre d'appareils électroniques produits est en augmentation, on l'estime à environ 30 milliards [2]. Ensuite, le développement récent de l'IA pousse à l'achat de nouvelles machines, ainsi qu'à une multiplication des centres de données. Enfin, depuis ses débuts, l'informatique suit une tendance d'augmentation exponentielle de la puissance, appelée la loi de Moore. Pourtant, en optimisant son code, on pourrait probablement conserver nos machines bien plus longtemps, concept décrit par la loi d'room [3]. L'éco-conception de services numériques va donc consister à réduire la quantité de ressources informatiques utilisées lors du développement d'un produit (conception, front-end, back-end, infrastructure). Ainsi, on tendra à réduire le rythme de renouvellement des terminaux, et l'énergie consommée par le réseau et les centres de données.

Les outils

Avant de mettre les mains dans le code, on peut chercher à avoir une vision d'ensemble avec des référentiels de bonnes pratiques. Le RGESN (Référentiel Général d'Eco-conception de Services Numériques) de l'ARCEP [4] consiste en 78 principes directeurs qui visent à évaluer le niveau de maturité en éco-conception sur l'ensemble de son service numérique. Il permet d'avoir une vision haut-niveau. Pour le compléter, on peut utiliser les 115 bonnes pratiques d'éco-conception web de GreenIT [5]. Celui-ci donne des conseils plus pratiques et ciblés, directement utilisables par des développeurs dans leur travail quotidien. Enfin, on peut compléter ces deux référentiels avec le guide d'éco-conception de services numériques des designers éthiques [6]. Il propose une vision moins technique et plus axée sur le produit, pour se poser les bonnes questions en matière de design, afin de le simplifier dès sa conception. En effet, en limitant le nombre de fonctionnalités d'une application, ainsi que leur complexité, on actionne le levier principal de l'éco-conception, qui intervient avant même de démarrer le développement.

On peut ensuite mesurer certains paramètres d'une application pour savoir si elle est éco-conçue : avec des métriques techniques (temps d'exécution, RAM utilisée, nombre d'éléments de DOM, etc.), on s'intéresse aux performances et donc in fine au ressenti utilisateur.
Par exemple, un site web qui se charge rapidement, qui n'a pas de lenteurs à l'utilisation et qui permet d'aller directement à la section qui intéresse l'utilisateur ne créera pas de sentiment d'obsolescence qui pourra lui faire renouveler son terminal. Une fois qu'on a défini ces métriques, il faut alors se baser sur un ou plusieurs parcours utilisateur types pour mesurer un ensemble cohérent d'actions, et pas juste une page web ou un appel de fonction isolé. Ce parcours type est appelé unité fonctionnelle.

Il existe de nombreux outils de mesure pour l'éco-conception, qu'il faut dans l'idéal combiner pour estimer l'impact environnemental d'un service numérique. D'abord, GreenIT Analysis [7], une extension de navigateur, permet de calculer un éco-index, qu'on peut voir comme un score d'éco-conception. Avec une note de A à G, on peut avoir une idée

de la performance d'un site web en fonction de plusieurs critères qui simulent les trois tiers mentionnés plus tôt : terminal, réseau, centre de données. Il donne par ailleurs une liste de bonnes pratiques de son référentiel en testant si elles sont correctement appliquées sur le site web. Cet outil peut être combiné avec Google Lighthouse [8], un outil qui n'est pas orienté éco-conception initialement, mais qui permet lui aussi d'obtenir un score et des conseils d'amélioration complémentaires pour améliorer les performances de son site. Grâce à ses propres indicateurs (FCP, CLS, NIP, etc.), il permet d'avoir une mesure fine du ressenti utilisateur. Côté back-end, on peut citer Scaphandre [9], un outil open-source qui mesure la consommation électrique de la machine qui exécute le code serveur. Enfin, côté cloud provider, Cloud Carbon Footing [10] tente d'uniformiser les calculs d'émissions de CO2 liées à l'utilisation des centres de données (Google, Amazon et Microsoft ayant des méthodes de calcul différentes pour GCP, AWS et Azure).

Conception/Produit

Avant tout, le travail d'un développeur qui éco-conçoit est un travail de persuasion envers les autres membres de son équipe. En effet, en réduisant en amont le nombre de fonctionnalités à développer ainsi que leur complexité, on s'assure qu'un produit est éco-conçu, car il va droit au but.
Il faut donc convaincre les chefs de projet, PO, designer UX/UI, etc., qu'il faut cibler le besoin essentiel, et supprimer le superflu.

Sur un site de e-commerce, les publicités et suggestions d'articles à acheter alourdissent la page, et détériorent l'expérience utilisateur. Dans la mesure du possible, il faut travailler à une réduction des publicités, et à une simplification des pages pour conserver uniquement les informations les plus pertinentes.

Sur un site de réservation de billets de train, multiplier les propositions de services annexes non sollicités complexifie le parcours, et demande plus de ressources. En effet, multiplier les pages sur lesquelles naviguer détériore l'expérience utilisateur, et augmente le temps d'écran, ce qui résulte en une consommation plus élevée côté client.

Autre exemple, les widgets interactifs de cartographie ou de carrousels demanderont à l'utilisateur de télécharger beaucoup de CSS et de JavaScript, pénalisant ainsi les performances de la page.

Une technique peut alors être de concevoir « mobile-first » en simplifiant l'interface au maximum grâce à la contrainte du mobile. En effet, en affichage mobile, on élimine le contenu superflu en enlevant les sections inutiles, certains widgets interactifs, et on n'affiche que le cœur de l'information à transmettre à l'utilisateur. En appliquant cette même logique au format desktop (ou ordinateur de bureau), on s'assure d'avoir un site simple qui va à l'essentiel, et on élimine d'office une partie de la complexité.

Autre technique évoquée plus haut : réduire le nombre d'étapes d'un parcours. Par exemple, dans un tunnel de paiement, les étapes intermédiaires consistant à la suggestion de produits ajoutent des étapes de parcours et augmentent le temps passé par l'utilisateur sur un site web. En réduisant le nombre d'étapes nécessaires à l'atteinte d'un but, on réduit le temps passé sur un site web et donc la consommation énergétique associée.

Une autre cause de lenteurs des sites web est la grande quantité de ressources nécessaire à son exécution. En tant que développeurs, nous avons un biais à cause de nos machines souvent très puissantes, et avec une très bonne connexion. Ainsi, même du code peu performant s'y exécute en un temps raisonnable, ce qui n'est pas le cas pour une majorité d'utilisateurs. En effet, ceux-ci peuvent avoir du matériel moins récent que le nôtre, ainsi qu'une connexion instable et plus bas débit. Une manière de simuler ces environnements défavorables est de travailler sous contrainte. Pour cela, on peut par exemple simuler une connexion 3G sur sa machine, ou limiter la quantité de RAM allouée à notre application. Ainsi, on se rapprochera de l'environnement d'exécution type des utilisateurs.

Dans la mesure du possible, il faut limiter voire supprimer les pisteurs (cracking) et la publicité. A eux seuls, ils peuvent être responsables de 70 % de la consommation d'un site web [11]. Les pisteurs permettent de mesurer le comportement des utilisateurs, ce qui est une bonne pratique si on s'en sert pour faire la chasse aux fonctionnalités peu ou pas utilisées. Malheureusement, la plupart du temps, ils sont utilisés pour récolter une grande quantité de données sur les utilisateurs de nos applications, sans se soucier de la quantité et de la nature des données récoltées. En plus de potentiellement créer des problèmes de RGPD, ils font de nombreuses requêtes réseau, ce qui pénalise fortement les performances d'une page web. Pour pallier ce problème, on peut choisir de ne surveiller que certains événements très précis au lieu de toutes les actions d'un utilisateur. La publicité est quant à elle souvent imposée à un niveau hiérarchique au-dessus que celui du développeur, mais il faut être conscient de son impact très négatif sur l'éco-conception : téléchargement et exécution de grandes quantités de JavaScript, nombreuses requêtes réseau, affichage parfois très dynamique sur la page, menant à une forte charge CPU.

L'optimisation peut aussi passer par la pagination. Dans le cas d'une grande collection d'objets, on peut créer un système de pages, où chacune contient un petit nombre d'éléments d'une liste (en général autour de 10), ainsi que des pointeurs vers les pages précédentes et suivantes. Ce système est utile à tous les niveaux : taille de la réponse à la requête en base de données, taille de la réponse envoyée au front-end, et CPU nécessaire à l'affichage des éléments.

Enfin, il faut privilégier les pages statiques quand c'est possible. Pour certains usages comme les blogs, on n'est pas obligé d'utiliser un système complexe impliquant une technologie front-end (comme React ou Vue.js), un serveur applica-

tif et une base de données. De simples pages statiques en HTML et CSS stockées sur un serveur suffisent et permettent un chargement quasiment instantané même avec des connexions lentes, ainsi qu'une très bonne rétrocompatibilité. Des outils comme Hugo.io [12] permettent cette génération de sites statiques.

Front-end

Gestion des images

Côté front-end, le gain de performances le plus simple à obtenir (et souvent le plus important) concerne les images. En effet, la tendance générale est d'utiliser des formats non adaptés au web ou non compressés comme png. De plus, les images ont souvent une taille bien trop grande pour le support qui les affichent. Il est en effet courant que les images originales soient en 1080p, voire en 4K, pour être affichées en très petit sur un écran de smartphone. C'est une double peine pour les performances : le navigateur perd du temps à télécharger une image de grande taille pour l'afficher en petit et la redimensionner, ce qui consomme du temps CPU.

Pour pallier ce problème, il faut jouer sur deux paramètres : le format de l'image, et sa taille. Deux formats permettent d'obtenir une très bonne compression tout en conservant la qualité de l'image : webp et avif. Ils sont particulièrement adaptés à l'affichage d'un site web, gèrent la transparence comme png, et permettent souvent d'obtenir des images de très bonne qualité qui font seulement quelques dizaines de ko. Cependant, attention à ne pas pénaliser la rétro-compatibilité. En effet, même s'ils sont très largement supportés par les navigateurs modernes, ces deux formats ne sont pas aussi universels que jpeg par exemple. Il faudra donc prévoir un fallback en jpeg si on utilise un de ces deux formats, afin de garantir la compatibilité avec les plus vieux navigateurs. CanIUse [13] permet d'en savoir plus sur la compatibilité des navigateurs avec diverses technologies.

```
<picture>
 <source type="image/webp" srcset="illustration.webp">
 <source type="image/svg+xml" srcset="illustration.svg">
 <img src="illustration.png" alt="A hand-made illustration">
</picture>
```

Ici, la balise picture permet de préciser plusieurs images dans des balises source. La première sera celle qui est privilégiée, et si le navigateur ne la supporte pas, elle passera à la suivante.

L'autre paramètre à modifier est la taille des images. Il faut d'abord bien connaître ses utilisateurs pour savoir par exemple s'ils utilisent majoritairement des smartphones ou des ordinateurs. On notera que la plupart du temps, les applications web ne sont pas affichées sur des écrans de télévision en 4K, on peut donc se limiter sans risque à du 1080p (1920x1080 pixels). Ensuite, une astuce consiste à stocker chaque image dans plusieurs tailles, afin d'envoyer à l'utilisateur celle qui correspond le mieux à la résolution de son écran. Classiquement, on stocke chaque image dans 4 ou 5 tailles pour couvrir une majorité de résolutions d'écran. Cette technique consomme un peu plus de stockage côté serveur, mais permet de réduire l'utilisation de la bande passante en servant à l'utilisateur une image à la bonne taille, réduisant aussi la charge CPU pour un éventuel redimensionnement.
Voici un exemple de code avec plusieurs images de différentes tailles :

Figure A

```
<picture>
 <source
  srcset="image-small.png 320w, image-medium.png 800w, image-large.png 1200w"
  sizes="(min-width: 60rem) 80vw, (min-width: 40rem) 90vw, 100vw"
 />
 <img src="image-small.png" alt="Image description" />
</picture>
```

Ici, les balises picture et source permettent de gérer 3 formats d'image, chacun s'affichant à une taille d'écran différente. Pour effectuer ces opérations, on pourra utiliser Squoosh [14], un outil gratuit, open-source, et qui s'exécute côté client (pas de problème de RGPD donc) afin de tester plusieurs formats d'image, modifier leur taille ainsi que de nombreux autres paramètres. L'image idéale sera celle qui conserve la meilleure qualité aux yeux de l'utilisateur, tout en ayant une taille réduite. Avec cet outil, on arrive généralement à réduire la taille d'une image mal optimisée d'au moins 80 ou 90 %.

Avec Squoosh, sur cette image, en réduisant les dimensions de l'image par 2 et en passant de jpeg à webp, on passe de 3,99 Mo à 47 Ko sans altérer la qualité de manière visible.
Figure A

Taille du bundle

Après les images, la plus grosse ressource d'un site est souvent son code JavaScript. En effet, depuis une dizaine d'années, la tendance est à l'augmentation de la quantité de code JavaScript exécutée dans le navigateur, ce qui n'est pas anodin sur les performances. Souvent, une grosse partie du code JavaScript du front-end ne vient pas du développeur lui-même, mais de bibliothèques tierces, parfois très lourdes. Bundlephobia [15] permet de vérifier la taille des bibliothèques JavaScript. Trois indicateurs sont à surveiller sur cet outil : le premier est la taille de la bibliothèque. Au-delà de quelques centaines de ko, c'est probablement trop lourd,

sauf peut-être pour une bibliothèque primordiale au fonctionnement de l'application. Sinon, il faudra chercher des alternatives plus légères. Ensuite, il faut regarder l'indicateur du temps de téléchargement en 3G. Comme précisé plus haut, il est primordial de prévoir un site web qui fonctionne avec des mauvaises connexions. Au-delà de quelques secondes de temps de téléchargement en 3G, il est donc là aussi conseillé de chercher une alternative. Enfin, on peut voir sur l'outil la mention – ou non – de « tree-shaking ». Le tree-shaking est une opération effectuée par le bundler (comme webpack par exemple) à la construction de l'application. Elle consiste en un retrait des parties inutilisées de la bibliothèque. Mais cet avantage nécessite une conception modulaire, ce qui n'est pas le cas pour toutes les bibliothèques.

Une autre question qui se pose en éco-conception est le nombre de fichiers CSS et JavaScript d'une application. Vaut-il mieux bien diviser son code avec beaucoup de fichiers ou au contraire tout centraliser en un unique fichier ? La réponse se trouve entre les deux. Si on prend l'exemple du CSS, avec un unique fichier à télécharger, on réduit le nombre de requêtes à une seule, ce qui est intéressant pour la bande passante. Par contre, on embarque potentiellement sur chaque page du CSS inutilisé, destiné à d'autres parties du site web. Autre effet négatif, au moindre changement du code CSS, l'utilisateur devra tout télécharger à nouveau, puisque le cache navigateur ne pourra pas être utilisé. A l'inverse, si on décide de beaucoup compartimenter le code CSS, on bénéficie du cache navigateur en cas de changement, on limite le code téléchargé et non-exécuté, par contre on multiplie le nombre de requêtes au serveur, ce qui in fine pénalise les performances. La bonne approche se trouve donc entre ces deux extrêmes.

Chargements asynchrones

La plupart des sites web font souvent plusieurs dizaines (voire centaines) de requêtes pour afficher une seule page. C'est un critère important à regarder car certaines ressources peuvent bloquer l'affichage tant qu'elles n'ont pas été téléchargées et exécutées dans le navigateur. Pour optimiser le nombre de requêtes faites au premier affichage de la page, on peut utiliser un mécanisme de lazy-loading. Il consiste à ne charger un contenu que s'il arrive dans la zone visible par l'utilisateur. Par exemple, on peut charger des images ou données de bas de page uniquement au défilement. Ainsi, au premier chargement, l'utilisateur ne télécharge que les données nécessaires à l'affichage des éléments de haut de page. Puis, au défilement, plus de données pourront être téléchargées, mais les utilisateurs n'allant pas jusqu'en bas de la page ne téléchargeront pas inutilement des fichiers. Le lazy-loading d'images peut être mis en place très simplement avec l'attribut HTML « loading=lazy ». Pour des cas plus complexes, par exemple une requête au back-end à déclencher au défilement, on peut utiliser l'API navigateur Intersection Observer [16] qui permet d'exécuter du code JavaScript à l'intersection entre la zone visible par l'utilisateur et un élément de DOM.

```
<img src="image.jpg" alt="..." loading="lazy" />
<iframe src="video-player.html" title="..." loading="lazy"></iframe>
```

L'attribut loading=lazy peut être appliqué sur les balises img et iframe, et est géré nativement par le navigateur

Back-end

Pour les cas plus complexes, on peut mettre en place du cache pour réduire la charge de la base de données. Pour éviter d'y faire trop d'appels, parfois coûteux, on met en place un serveur de cache (comme Varnish ou Redis par exemple) qui stocke le résultat des appels les plus fréquents à la base de données. Ensuite, à chaque appel que le serveur applicatif fera pour obtenir des données, le serveur de cache fera un « hit » ou un « miss ». Dans le cas d'un « hit » (qu'on cherche à maximiser), le serveur ira chercher la donnée directement dans sa mémoire et répondra directement au serveur applicatif, évitant ainsi une requête au serveur de base de données. Dans le cas d'un « miss » (qu'on cherche à minimiser), le serveur de cache n'aura pas la donnée et devra faire une requête au serveur de base de données pour récupérer la bonne information, puis la renvoyer au serveur applicatif. Dans ce cas, la nouvelle donnée pourra être stockée dans le serveur de cache pour le prochain appel. Il existe de nombreuses stratégies de cache, chacune adaptée à un besoin précis, mais leur mise en place implique une complexité de développement et de maintenance. C'est seulement au-delà de certaines volumétries qu'il devient approprié.

On peut ensuite s'intéresser à la rétention des données. Par défaut, une application web en stocke de nombreuses : logs applicatifs, logs d'erreur, données utilisateur, mesures d'audience, etc. Au fil du temps, ces données s'accumulent et occupent un grand espace de stockage, alors qu'elles sont peu ou pas utilisées. Pour pallier ce problème, on peut définir une politique de rétention de données : par exemple, pas plus de quelques semaines ou quelques mois pour les logs applicatifs et les logs d'erreur. Quant aux mesures d'audience, on peut les agréger au bout d'un an par exemple. En effet, sur le long terme, on est souvent plus intéressé par le nombre de visites un mois donné qu'au détail de chaque visite (navigateur utilisé, localisation de la connexion, temps de visite, etc.). Ainsi, on peut considérablement réduire le volume de données stockées tout en gardant les informations essentielles.

Enfin, il faut penser au décommissionnement de son application, et des briques qui la constituent. A la création d'une application web, on pense rarement à sa fin de vie, alors que nombre d'entre elles ne durent pas plus de quelques années. Il arrive que certaines applications « lega » continuent à consommer des ressources informatiques et engendrent des coûts à cause d'abonnements ou d'hébergements oubliés. Pour prévenir ce problème, on peut documenter lors du développement les différents services associés à une application afin de les décommissionner le moment venu.

Infrastructure et hébergement

D'une manière générale, la démarche FinOps vise à réduire les coûts de l'infrastructure et fait se poser des questions intéressantes pour adapter l'architecture à la demande tout en réduisant les coûts, et donc, de manière générale, la consommation électrique. Il y a une exception à cette

logique : on peut parfois réduire les coûts en plaçant ses serveurs aux Etats-Unis plutôt qu'en Europe. Cependant, ce pays a un mix électrique bien plus carboné que la plupart des pays européens (notamment la France où l'énergie est bas carbone en raison de l'énergie nucléaire). Il est donc préférable de choisir des centres de données européens et même français.

Pour les échanges faits sur le réseau, il faut passer de HTTP 1 – encore largement utilisé – à HTTP 2 ou HTTP 3. Pour plusieurs appels en HTTP 1, chaque requête reçoit une réponse, sans optimisation sur les en-têtes HTTP. En HTTP 2, les en-têtes sont mutualisés, ce qui économise de la bande passante. De plus, de nombreux navigateurs sont compatibles, pas de raison de ne pas s'y mettre donc. Aujourd'hui, on peut même utiliser HTTP 3, qui est tout de même moins compatible. Cette version utilise UDP plutôt que TCP dans les couches réseau basses, ce qui permet d'améliorer encore un peu les performances des requêtes.

Un autre aspect important est la compression des ressources [17]. Quand un serveur envoie une image, police d'écriture, fichier HTML/CSS/JS, etc., elle peut être compressée avec plusieurs algorithmes. Le plus rétro compatible est Gzip, qui permet d'envoyer une donnée plus petite, économisant ainsi de la bande passante. Le navigateur s'occupera ensuite de décompresser la ressource automatiquement grâce aux en-têtes HTTP. Brotli est un autre algorithme de compression, qui propose généralement une meilleure compression que Gzip, mais avec un temps de compression souvent plus long. Enfin, ZSTD propose un excellent ratio de compression avec des temps de compression très bas, mais il est moins bien supporté par les navigateurs que Gzip et Brotli. Quel que soit l'algorithme choisi, il faut en mettre un en place pour économiser de la bande passante, et ainsi réduire les temps de téléchargement des ressources côté client.

Du côté des machines virtuelles (ou VM) qui hébergent nos applications web, il faut être vigilant à la sur-utilisation de ressources. On a tendance à choisir par défaut des conteneurs avec beaucoup de CPU et de RAM pour exécuter des petites ou très petites applications. Il est donc préférable de commencer les plus petites tailles d'instance, puis de changer dans un second temps, si la charge augmente. De plus, on peut s'appuyer sur des « auto scaling groups » qui créent et détruisent ces conteneurs en fonction de la charge, adaptant ainsi très finement et automatiquement la quantité de ressources allouées au fonctionnement de l'application.

Et si on va au-delà de la technique, on peut se questionner sur la disponibilité du service numérique. Comme pour les VM, on cible parfois des niveaux de haute disponibilité qui ne sont pas vraiment indispensables à l'application qu'on développe. Si on prend la décision de ne pas viser une disponibilité maximale, on peut réduire ses coûts d'infrastructure et simplifier son architecture. Par ailleurs, certains services numériques n'ont peut-être pas besoin de fonctionner en continu (par exemple un intranet d'entreprise). On peut imaginer éteindre les serveurs la nuit et le week-end. Et si c'est trop compliqué à mettre en œuvre, on peut au moins le faire sur les environnements de développement et de pré-production.

Enfin, la sélection de son fournisseur de cloud a de l'importance. Les trois principaux, AWS, GCP et Azure se déclarent très écologiques et parfois même neutres en carbone, mais sont en réalité plutôt mauvais élèves sur de nombreux points : consommation d'eau pour le refroidissement des centres de données, opacité sur les indicateurs environnementaux et leurs modes de calcul, et politique peu ambitieuse de renouvellement des équipements et de recyclage. Les services de cloud français et européens comme OVH et Scaleway sont bien plus transparents sur leurs méthodes de calcul, et attentifs à leurs impacts environnementaux.

En conclusion, le numérique génère une pollution importante et croissante, et les services informatiques que nous développons y sont pour quelque chose. En visant d'abord la sobriété des fonctionnalités, il est possible de réduire le renouvellement des terminaux ainsi que la quantité de ressources informatiques utilisées pour faire fonctionner nos applications. Une fois qu'on s'est concentré sur l'essentiel, on peut alors optimiser à tous les niveaux : produit, front-end, back-end et infrastructure.

Sources

[1] Le GreenIT en 2025 : https://greenly.earth/en-us/blog/industries/everything-you-need-to-know-about-green-it-in-2022

[2] Etude GreenIT : https://greenit.eco/wp-content/uploads/2025/02/greenit-etudemonde2025-20250211.pdf

[3] La loi d'erooM : https://www.lewebvert.fr/blog/2024-06-20-interview-tristan-nitot/

[4] RGESN : https://www.arcep.fr/mes-demarches-et-services/entreprises/fiches-pratiques/referentiel-general-ecoconception-services-numeriques.html

[5] 115 bonnes pratiques d'éco-conception web : https://collectif.greenit.fr/ecoconception-web/115-bonnes-pratiques-eco-conception_web.html

[6] Guide d'éco-conception de services numériques : https://designersethiques.org/fr/thematiques/ecoconception/guide-d-ecoconception

[7] GreenIT Analysis : https://www.greenit.fr/2019/07/02/web-evaluez-lempreinte-dune-page-en-un-clic/

[8] Google Lighthouse : https://developer.chrome.com/docs/lighthouse

[9] Scaphandre : https://github.com/hubblo-org/scaphandre

[10] Cloud Carbon Footprint : https://www.cloudcarbonfootprint.org/

[11] Impact des publicités et trackers : https://marmelab.com/blog/2022/01/17/media-websites-carbon-emissions.html

[12] Hugo.io : https://gohugo.io/

[13] CanIUse : https://caniuse.com/

[14] Squoosh : https://squoosh.app/

[15] BundlePhobia : https://bundlephobia.com/

[16] Intersection Observer : https://developer.mozilla.org/en-US/docs/Web/API/Intersection_Observer_API

[17] Comparaison des compressions : https://aminshamim.medium.com/gzip-deflate-brotli-and-zstd-which-compression-algorithm-should-you-use-for-your-website-033ca5cfa7ca

Jérémy Pastouret
Journaliste pour les e-novateurs, le média du numérique responsable, éthique et inclusif.

Éco-conception et sobriété numérique face à l'explosion de l'IA, comment tenir bon ?

Avant de parler des solutions, posons les bases de ce qui pose problème dans la course au développement de LLM (Large Language Model), ainsi que dans leur intégration de plus en plus forte dans le métier de développeur·se.

Datacenters, eau, électricité : les coulisses de l'IA

Les impacts environnementaux de l'intelligence artificielle s'annoncent massifs. Pour celles et ceux qui en doutent encore, voici quelques chiffres parlants :

- Selon l'Agence internationale de l'énergie (IEA), les systèmes d'IA pourraient consommer d'ici 2030 plus d'électricité que le Japon tout entier, soit 945 TWh. Aux États-Unis, la consommation électrique liée aux centres de données a déjà augmenté de 130 %.
- L'expansion continue : des projets de création de centres de données se multiplient, notamment en Europe via le AI Continent Action Plan, mais aussi aux États-Unis, déjà leader mondial avec 5 381 centres de données recensés en mars 2024. Le projet Stargate, par exemple, finance d'immenses infrastructures pour accélérer le développement des modèles d'IA — les premières images circulent déjà.

Concrètement, cela signifie que l'IA consomme énormément d'électricité, mais aussi des millions de litres d'eau pour refroidir ses serveurs, et des ressources rares pour construire toujours plus de datacenters.

À l'autre bout de la chaîne, une tendance se dessine : de plus en plus de développeur·ses et bidouilleur·ses envisagent de s'équiper de machines plus puissantes, simplement pour faire tourner localement des modèles d'IA open source. L'idée d'une IA sécurisée en local devient ainsi un argument marketing pour justifier le renouvellement du matériel. Une dynamique qui vient alimenter encore davantage la pression sur les ressources et l'obsolescence programmée.

Pendant ce temps, les géants de l'IA peinent toujours à trouver un modèle économique viable. Faut-il s'attendre à ce qu'ils transfèrent progressivement la facture énergétique et matérielle sur les utilisateurs·rices ? C'est une question cruciale. Sur ce sujet, je vous recommande l'article de Louise Pastouret, *Les coûts de l'IA bientôt supportés par les utilisateurs ?*, publié sur le média *Les e-novateurs*.

Face à cette trajectoire insoutenable, il existe heureusement des alternatives plus sobres et prometteuses :

- Alléger les modèles d'IA pour qu'ils puissent fonctionner sur des machines anciennes ou peu puissantes. Cela réduirait les coûts pour les fournisseurs… même si cela ne ferait pas les affaires des fabricants de cartes graphiques.
- Favoriser les petits modèles spécialisés, plus efficaces, plus légers et surtout moins gourmands, afin d'éviter de devoir changer d'ordinateur à chaque évolution technologique.

Maintenant que le constat de l'impact de l'IA est posé, que faire ensuite ?

Réponse : Agir — et tenir bon.

Face à l'ampleur de l'impact écologique de l'IA, l'inaction n'est plus une option. Et en tant que développeur·ses, nous ne sommes pas uniquement spectateurs·rices : nous sommes directement concerné·es, à la fois dans nos outils, nos pratiques, et nos perspectives d'avenir.

Depuis quelques années, de nouveaux outils alimentés par l'IA ont émergé… accompagnés d'une vague de craintes pour la profession. Pour les plus ancien·nes d'entre nous, rien de nouveau sous le soleil :

- « Les AGL (Atelier de génie logiciel) comme Windev vont remplacer les développeurs… »
- « Un nouveau framework devient à la mode, il faut tout réapprendre. »
- « Le no-code va tuer le métier. »
- « L'IA code plus vite que nous : autant changer de carrière. »
- « Si tu n'utilises pas d'IA, tu es déjà dépassé·e. »

Au lieu de céder à la panique ou à la résignation, j'ai choisi de me doter de nouvelles armes :

- Sensibiliser au *slopsquatting* : lorsqu'on code avec une IA, elle peut « halluciner » des bibliothèques qui n'existent pas, mais qui ont été infiltrées par des hackers·ses avec du code malveillant. J'ai détaillé ce risque dans un article pour *Les e-novateurs* : « *IA et slopsquatting : les néo-développeurs confrontés aux enjeux de cybersécurité* ».
- Pointer les effets de bord de la génération de code automatique : failles de sécurité, code inutilement alourdi, perte de lisibilité, incohérences logiques… Résultat ? On passe plus de temps à débuguer qu'à créer. Et on ne profite même plus du plaisir de corriger nos propres bugs.
- Comparer pour mieux décider : pour les plus sceptiques ou enthousiastes, un simple chronomètre suffit. Mesurez le temps réel passé à accomplir une tâche avec ou sans IA. Spoiler : ce n'est pas toujours l'IA qui gagne. **Figure 1**
- Développer son argumentaire : pas évident de faire entendre sa voix quand on est seul·e face à une équipe qui utilise frénétiquement l'IA. Pour nourrir le débat de manière constructive, une bonne approche consiste à appuyer son

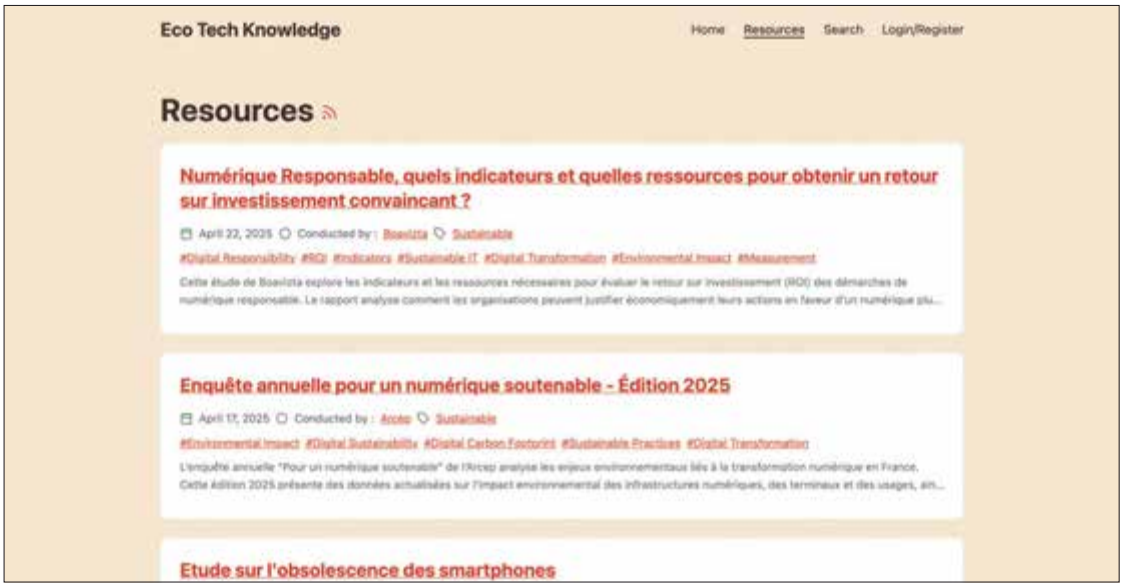


Figure 1

propos sur des chiffres issus de sources fiables : études scientifiques rigoureuses, rapports d'organismes reconnus… Pour cela, plusieurs outils peuvent vous aider. Le média Les e-novateurs compile déjà une veille régulière sur ces sujets. Et l'outil Eco Tech Knowledge — accessible via le lien en ressources — constitue une bibliothèque open source au niveau international, riche en publications consacrées à l'impact du numérique sur l'environnement et la société. Une future mise à jour de la plateforme permettra même de filtrer automatiquement les chiffres et études par thématique, pour gagner du temps dans la construction de vos argumentaires.

Développer un service web éco-conçu avec l'IA ?

Une illusion d'efficacité

Au début de la programmation, les développeurs·ses utilisaient des cartes perforées pour encoder leurs instructions. Les ordinateurs, massifs et coûteux, nécessitaient une planification minutieuse : chaque ligne de code devait être pensée, testée, transcrite. Le temps humain était la ressource principale ; l'exécution était lente, mais très optimisée.

Aujourd'hui, on a inversé la logique : l'IA génère du code à la volée, en réponse à des descriptions en langage naturel. Ce qui prend du temps et de l'énergie n'est plus l'humain, mais la machine : analyser la requête, générer le code, deviner où l'intégrer. Le processus prend du temps tout en consommant une énergie colossale, bien souvent disproportionnée par rapport à l'action réalisée.

Si vous espérez concevoir un service sobre en énergie tout en laissant l'IA écrire votre code, vous risquez de faire exploser votre empreinte carbone avant même la mise en production. Comme toujours, tout dépend de l'usage. Petit tour d'horizon des pratiques actuelles — et de leurs impacts :

Mode Autopilot

Je ne code rien, je décris simplement la fonctionnalité : l'IA s'occupe du reste. J'attends, je clique, je bois un café, je laisse tourner l'IDE dopé à l'IA, et j'espère qu'il trouvera où injecter la bonne ligne de code.

Problème : ce mode est non seulement très énergivore, mais il est aussi intellectuellement épuisant. Plus le projet grossit, plus le modèle doit digérer des volumes massifs de données. Résultat : lenteurs, incohérences, erreurs… et fatigue mentale. Sans compter que les IDE boostés à l'IA ne sont pas encore capables de comprendre correctement les structures complexes. Certains vont jusqu'à interroger l'IA ligne par ligne, générant un déluge de requêtes inutiles. Il serait temps que les éditeurs d'IDE et de plugins IA partagent des mesures précises d'impact et fassent preuve de transparence.

L'alternative ? Reprendre la main en réduisant volontairement les interactions avec l'IA… et se faire confiance.

Mode *Debug*

À chaque message d'erreur surligné en rouge, je demande à l'IA : « répare ce bug ». Encore une fois, soit on indique l'erreur et on donne tout le codebase à l'IA : l'impact est assez important, sans compter le temps d'attente. Soit on identifie bien la source de l'erreur et on précise notre prompt à l'IA, sans lui donner accès à tout le projet : ce qui est beaucoup plus léger.

Pourquoi ne pas simplement effectuer une recherche ciblée, **comprendre l'erreur et s'améliorer** ? En déléguant systématiquement la résolution de bug, on perd en logique, en autonomie. Le cerveau aussi a besoin d'entraînement.

En plus, on n'est même pas certain·es que l'IA fera mieux que nous. En fonction de la taille du projet, elle peut tourner en rond et accroître sa complexité. Au bout d'un moment, vous en avez assez et vous enquêtez vous-même, découvrant que la solution était beaucoup plus simple que toutes les tentatives de l'IA. Encore une fois celle-ci fonctionne bien dans des cas bien définis, dans la gestion d'erreurs assez classiques.

Mode Copier-Coller / hors IDE

Je pose ma question directement dans ChatGPT, à l'extérieur de l'environnement de développement. Bonne nouvelle : c'est plus léger et plus confidentiel. Car si ChatGPT a pu scraper tout le Web sans autorisation, pourquoi pas votre code aussi ?

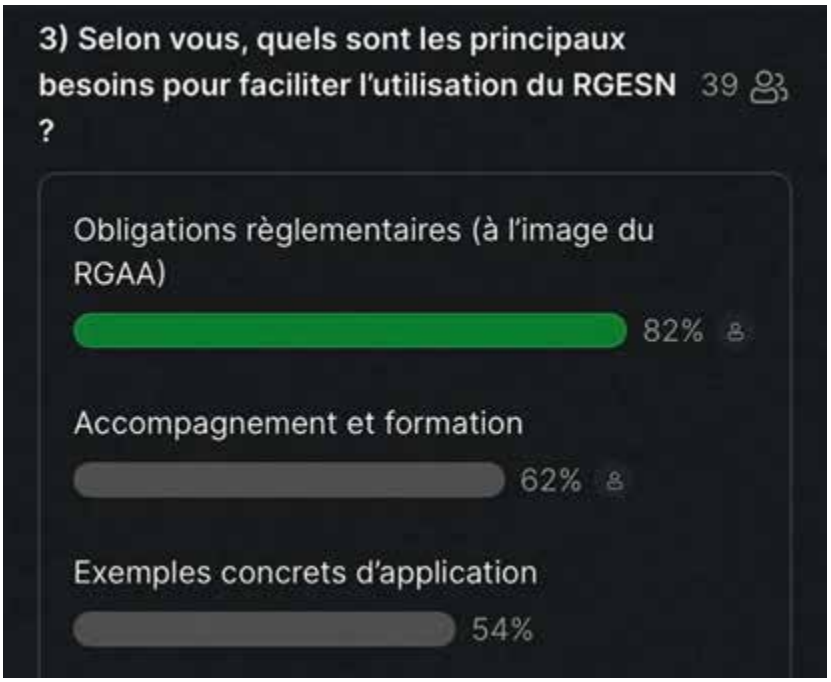


Figure 2

Et si vous posez une question sur comment créer une IA avec… une autre IA, ne soyez pas surpris que cela déplaise fortement à OpenAI (bonjour *DeepSeek*, bonsoir modération).

—> Cette friction — le fait de devoir sortir du flux de travail — limite l'automatisme. Et c'est une bonne chose. Car l'objectif est clair : éviter à tout prix que le recours à l'IA devienne un réflexe. Comme le scroll infini ou la dopamine des likes, les GAFAM savent très bien créer l'addiction.

Mode *Relecteur de PR* (aka le cauchemar)

Laisser une IA relire une pull request ? C'est probablement le pire scénario.

Même entre humain·es, une relecture peut être tendue. Avec une IA incapable de comprendre le contexte, le nom de la branche, la logique métier ou la portée du changement… c'est l'accident assuré.

Résultat : feedbacks erronés, remises en question inutiles, perte de confiance et surtout des erreurs passées en production. Dans l'open source, où les ressources humaines sont limitées, l'IA peut sembler être une solution miracle. Mais en pratique, elle fragilise les dynamiques collectives et renforce la solitude des contributeurs·rices.

Un référentiel pour reprendre la main sur le numérique

Le 17 mai 2024, la France a publié son Référentiel Général d'Écoconception des Services Numériques (RGESN). Attendu depuis longtemps par les acteurs·rices de la conception numérique responsable, ce document arrive un an et demi après l'émergence de ChatGPT auprès du grand public, dans un contexte de fortes attentes autour de l'impact environnemental du numérique.

« Un référentiel unique et robuste porté par la puissance publique », déclarait Sandrine Elmi Hersi (cheffe de l'unité "Internet ouvert" de l'Arcep et pilote de ce référentiel) dans une interview accordée au média Les e-novateurs lors de sa sortie.

Composé de 78 fiches pratiques, le RGESN ne fait pas l'impasse sur l'intelligence artificielle. Une section intitulée « Algorithmie » y est dédiée, avec 7 critères spécifiques à valider. À ce jour, aucun fournisseur de solutions d'IA ne semble avoir publié de déclaration d'écoconception conforme à ces exigences.

C'est d'ailleurs l'une des critiques récurrentes à l'égard du RGESN : bien qu'il ait vocation à s'appliquer à l'ensemble des services numériques, certaines technologies comme la cryptomonnaie ou l'Internet des objets (IoT) ne disposent pas encore de sections dédiées — elles ne sont que brièvement mentionnées. Par ailleurs, le référentiel gagnerait à proposer un système de filtrage ou de personnalisation, permettant d'adapter les critères en fonction des spécificités de chaque projet. Par exemple, ignorer les critères liés à l'intelligence artificielle lorsqu'un service n'en intègre pas .

Pour contribuer à l'amélioration du référentiel, un premier forum organisé par l'ARCEP a eu lieu le 19 mai 2025. L'événement a permis de recueillir les retours d'expérience d'acteurs comme Thales, la SNCF, France Télévisions ou encore Greenspector sur l'usage du RGESN dans leurs structures respectives.

Ce forum a également été l'occasion de réaffirmer un principe clé : le référentiel repose aujourd'hui sur une démarche volontaire de la part des organisations. Toutefois, un sondage réalisé en fin de session a révélé une tendance claire parmi les participant·es : le souhait que le RGESN évolue vers une obligation réglementaire à l'avenir, tout comme le RGAA.

Figure 2

Pour vous tenir au courant des actualités du RGESN, vous pouvez remplir le formulaire (lien en fin d'article).

Un consortium d'expert·es, ouvert à tous·tes, est également actif. Vous pouvez retrouver les informations et échanger à ce sujet sur le Mattermost de l'association des Designers Éthiques.

RGESN vs intelligence artificielle : David contre Goliath

Le Référentiel Général d'Écoconception des Services Numériques (RGESN) apporte une aide précieuse dans la conception des services web. Il invite, par exemple, à tester les sites sur des terminaux anciens (de plus de 5 ans), limiter l'usage de la vidéo, réduire le nombre de polices utilisées… des mesures qui peuvent sembler anecdotiques face à l'empreinte carbone d'un modèle d'intelligence artificielle, mais qui ont toute leur importance.

Pourquoi ? Parce que le Web a une histoire, et surtout un recul suffisant pour identifier clairement les leviers d'allègement. Ces efforts doivent continuer. Tout le monde n'utilise pas l'IA au quotidien, et les problématiques d'optimisation ne sont pas identiques selon les usages.

Le RGESN offre un cadre clair, qui permet non seulement de mieux concevoir mais aussi de communiquer en transparence avec les utilisateurs·rices sur les impacts de notre service : quels compromis ont été faits ? Quels impacts ont-ils ?

Chez Les e-novateurs, nous avons pris le temps de justifier un maximum de critères du RGESN — même ceux qui sont les

plus difficiles à objectiver. A la clé : une déclaration publique, accessible en bas de page du média, accompagnée de démonstrations vidéo montrant par exemple que le site fonctionne sous Windows XP ou d'autres systèmes anciens.
Côté IA, les fournisseurs ne sont pas encore à ce niveau de transparence. La course à la performance prime sur l'éco-conception. Pourtant, les outils existent. Un référentiel d'IA frugale a déjà été publié par l'Afnor et le ministère de la Transition écologique, avec l'aide d'expert·es (dont j'ai eu la chance de faire partie).

Vers plus de transparence sur l'impact des requêtes IA

Il est temps que les fournisseurs d'IA affichent clairement l'impact environnemental de chaque requête. En attendant, du côté des e-novateurs, nous avons lancé le développement d'une extension Chrome / Firefox (également compatible Android via Firefox mobile) qui, grâce aux données d'Ecologits (calculateur carbone spécialisé IA), estime la consommation liée à l'usage d'une IA sur le Web. **Figure 3**
C'est un projet open source, et nous sommes toujours ouvert·es aux contributions. L'outil permet de croiser l'impact des pages Web et celui des requêtes IA, pour rendre visible ce qui ne l'est pas toujours.

Sources :

Les e-novateurs : L'IA fait flamber la demande mondiale d'électricité
https://les-enovateurs.com/flash-news/1#ia-fait-flamber-demande-mondiale-electricite

Les e-novateurs : L'UE mise sur l'IA… au prix d'une explosion de centres de données ?
https://les-enovateurs.com/breves/union-europeenne-mise-sur-ia-explosion-centres-donnees

Figure 3

YouTube - Stargate Data Center being built in Abilene, Texas
https://www.youtube.com/watch?v=fUiI03X6DQc

Les e-novateurs : Les coûts de l'IA bientôt supportés par les utilisateurs ?
https://les-enovateurs.com/couts-ia-bientot-supportes-par-utilisateurs

Les e-novateurs : IA et slopsquatting : les néo-développeurs confrontés aux enjeux de cybersécurité
https://les-enovateurs.com/ia-slopsquatting-neo-developpeurs-confrontes-enjeux-cybersecurite

Déclaration d'eco-conception du média les e-novateurs - https://les-enovateurs.com/eco-conception

Eco-surf – extension impact carbone IA / Open Source
https://github.com/les-enovateurs/estimate-good-website

EcoTechKnowledge – liste de ressources sur l'impact du numérique sur l'environnement et la société - https://knowledge.les-enovateurs.com

Ecologits – calculateur d'impact des llm - https://ecologits.ai/latest/

Arcep - Vous êtes intéressés par les actualités du référentiel d'écoconception des services numériques ? Restons en contact ! https://www.arcep.fr/sondage/forum-referentiel-ecoconception-des-services-numeriques.html

Davide Usai

Avec un BTS en informatique industrielle suivi d'un parcours en ingénierie électronique, j'habite en région nantaise et je participe à l'animation d'un atelier de co-réparation. Je cherche à offrir aux lecteurs une réflexion accessible sur le fonctionnement des objets qui façonnent notre quotidien. Vous pouvez retrouver mes articles dans mon blog davideusai.eu

À vos tartines ! Un exemple concret de réparation

Cet article retrace l'expérience de réparation d'un grille-pain qui ne veut pas rester allumé. Le problème présenté est celui que l'on constate dans la plupart des cas : le levier d'insertion ne reste pas baissé. Il faut donc qu'il le tienne appuyé pour que ses tartines chauffent.
Ceci est également l'occasion d'aborder le fonctionnement de cet objet du quotidien ainsi que d'autres sujets profondément liés comme sa consommation et son indice de réparabilité. À vos tartines et bonne lecture !

Commençons par un peu d'histoire et d'anecdotes

Il faut savoir qu'il existe « Le musée international du grille-pain ». Il se trouve en Allemagne et il a un site web http://www.toastermuseum.com/
Est-ce que le grille-pain a été inventé en Allemagne ? Pas du tout. C'est une invention qui arrive depuis les États-Unis grâce à l'inventeur Frank Shailor qui travaillait pour la General Electric. Son D-12 de 1909 est bien le premier grille-pain qui ressemble tout particulièrement à ceux actuellement dans nos maisons. Le procédé fut possible grâce à l'invention du nichrome 4 ans plus tôt par un autre inventeur américain, Albert Leroy Marsh. Si vous trouvez quelque part une histoire romancée autour de Alan MacMasters à Edimbourg, sachez que c'est une fake new !

Figure 1

Fonctionnement

Les tartines sont insérées et le levier est baissé. Après un certain temps elles remontent avec une coloration dorée et un agréable parfum. Vous venez de soumettre vos tartines à un rayonnement infrarouge.
Effectivement la tartine a été pendant ce temps-là en proximité d'une source de chaleur produite par effet Joule à la suite du passage d'un courant électrique à travers un fil de nichrome. Cette chaleur a transformé l'amidon contenu dans la pâte en glucose ce qui a rendu votre pain plus croustillant. Ah les gourmands !
Vous pouvez déterminer ce temps d'exposition par un minuteur. Ce temps pourrait être automatiquement ajusté dans certains modèles en fonction de certains éléments variables.
Son électronique de contrôle n'est alimentée qu'au même moment que ses éléments contrôlés.
Cette particularité le rend probablement unique parmi l'ensemble d'appareils électroménagers que l'on peut retrouver dans une maison. Je trouve cette trouvaille aussi simple que magique.
Étant donné que le minuteur est toujours piloté par un potentiomètre, donc un composant mécanique et passif, un paramétrage reste tout de même possible avant le début de la cuisson.

Passons au démontage

Mettre à nu l'intérieur en soulevant le châssis et sans rien casser est particulièrement compliqué sur tous les modèles de cette marque. Ils le font très probablement délibérément, mais on y arrive quand même. **Figure 1**
L'on peut facilement voir le levier solidaire à un mécanisme composé principalement de 5 éléments

- Sur la droite un élément en plastique conique qui ferme un interrupteur bipolaire
- Sur la gauche un crochet qui travaille en binôme avec un électro-aimant
- Une tige centrale pour contraindre les mouvements sur l'axe vertical
- Un ressort sur la gauche pour maintenir normalement haut le mécanisme
- Un ressort sur la tige centrale pour amortir le mécanisme lorsqu'il effectue le mouvement du bas vers le haut au moment où le crochet ne sera plus maintenu par l'électro-aimant

Revenons au fonctionnement :

- Le levier se baisse
- L'élément en plastique ferme les contacts de l'interrupteur bipolaire
- Le courant électrique traverse la résistance
- L'effet Joule est en marche et les tartines commencent à griller
- À un endroit bien précis, un fil vient piquer la résistance pour sortir une tension souhaitée. Nous avons donc un pont diviseur

- Cette tension excite l'électro-aimant qui tient le crochet du mécanisme vers le bas. Ce qui nous permet de lâcher le levier qu'il restera donc en position basse
- À partir de ce moment, un compte à rebours commence avant que l'électro-aimant ne se déclenche
- Toute l'électronique sert à calculer ce temps en tenant compte du paramétrage dans le minuteur ainsi que d'autres éléments que nous verrons plus tard
- Le temps passé, l'électro-aimant n'étant plus alimenté va lâcher le crochet
- Le ressort sur la gauche tire vers le haut, ce qui va faire remonter le mécanisme
- Le ressort central amortit ce mouvement
- L'interrupteur bipolaire n'est plus fermé ce qui fait que le courant dans la résistance ne circule plus
- Comme il n'y a plus de courant, la sortie du pont diviseur donne une tension nulle. L'électronique n'est pas donc non plus alimentée.

Dans ce modèle, la tige centrale ne reste pas en place lorsque le châssis n'est pas installé. Cela gêne tout particulièrement les essais pendant la réparation. Je considère ce point comme un défaut de conception dans le design du produit.

L'électronique du grille-pain

Entre la prise électrique et les résistances chauffantes, on retrouve deux cartes :

- Une avec des éléments plutôt de puissance : **Figure 2**
- Et l'autre avec l'électronique de contrôle : **Figure 3**

Le fabricant de ces deux cartes est la Hong Kong Yorkwell Industries Limited : https://www.hkyorkwell.com.hk/index.html

Il s'agit dans les deux cas d'un PCB simple face et avec des composants exclusivement traversants. La sérigraphie ainsi que le soudage me semblent être d'une excellente qualité. La valeur fondamentale est bien lisible pour chaque composant élémentaire.

Le circuit intégré comporte la désignation A0201D. C'est le seul composant pour lequel nous ne trouverons rien, aucune datasheet, aucune information sur le fabricant.

Il semblerait y avoir du monde qui cherche des informations. Une personne a partagé le document TUV concernant le rapport de test d'émission EMC d'un certain nombre de produits utilisant cette puce. Le document est disponible ici : https://elektrotanya.com/toaster_kt600_kt600f_kt700_pt8a2511pe_pt8a2512ne_a0201d_kt600e_a0201f_a0201_kt600g2_kenyerpirito.pdf/download.html

Il s'agit de la société Ningbo Kaibo Group Co., Ltd. Il semblerait que la société ait obtenu la certification pour des cartes qui utilisent deux puces différentes. L'autre est la PT8A251. **Figure 4**

La bonne nouvelle est que pour celle-ci on retrouve la datasheet : https://www.mouser.com/datasheet/2/115/PT8A2511-1350784.pdf

Toujours dans le même document où l'on retrouve un schéma d'usage. **Figure 5**

Il est utile, car il reprend le fonctionnement du pont diviseur dont nous avons parlé plus haut. L'alimentation est à 230 VAC. L'alimentation du PCB est à 12 V. Toujours en AC, car elle est prise directement sur un point du pont diviseur. Elle sera redressée dans un second temps. Le schéma d'application est dans une configuration à trois résistances en série.

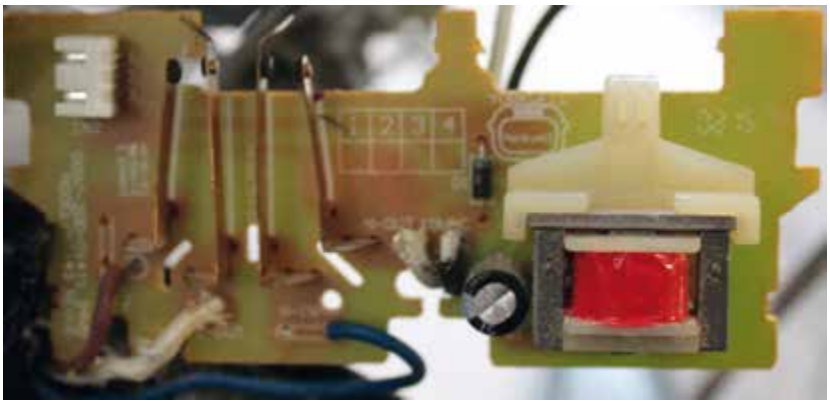


Figure 2

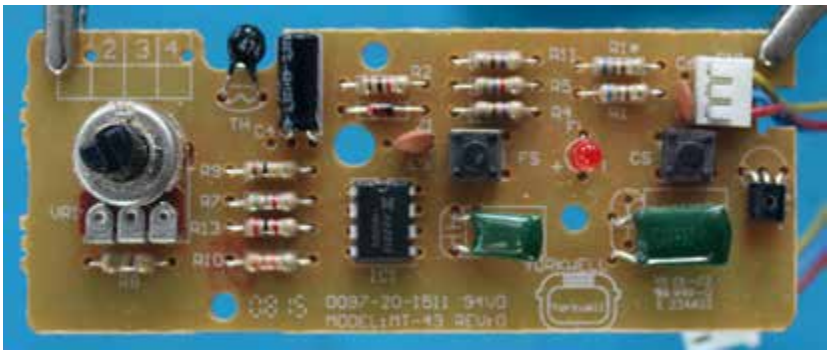


Figure 3

5. Integrate circuit	1) AnteeDesign Microelectronics Co.,Ltd.	A0201D (For KT-600, KT-600F, KT-600G, KT- 600G2, KT-700, KT-600E/4) A0201F (For KT-600E)	Notest with appliance)
	2) Linpo Shengyong Electronics(shenzhen)Ltd.	PT8A2511PE (For KT-600, KT-600F, KT-600G, KT-600G2, KT-700, KT-600E/4) PT8A2512NE (For KT-600E)	Notest with appliance)

Figure 4

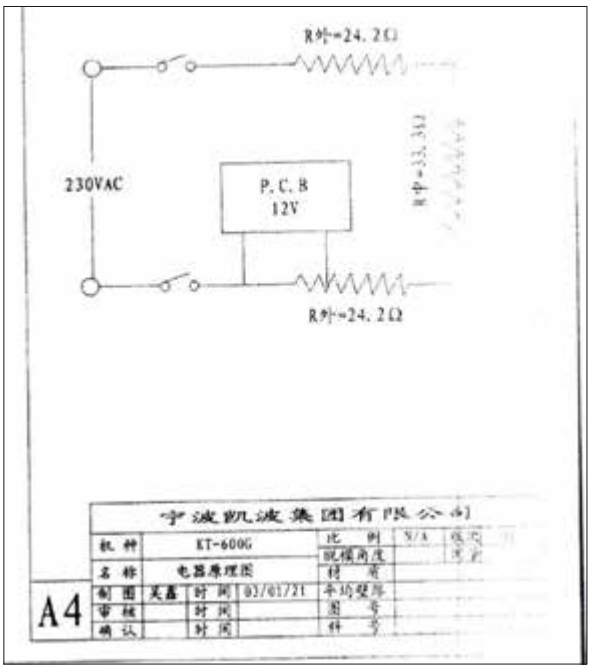
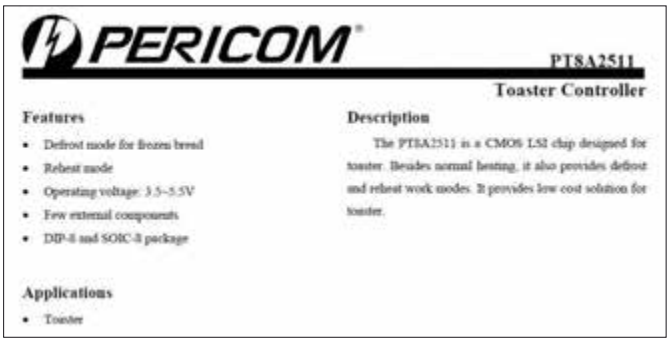


Figure 5

Dans un grille-pain pour deux tartines nous avons bien trois résistances de chauffe. Leurs valeurs de résistance sont également données.

Partons du principe que les puces A0201D et PT8A2511 soient tellement similaires que l'on va faire mener la suite de nos réflexions avec la datasheet de la deuxième. **Figure 6**

Au regard de la description, nous ne devons pas être si loin que cela.



PERICOM

PTSA2511

Toaster Controller

Features

- Defrost mode for frozen bread
- Reheat mode
- Operating voltage: 3.5~5.5V
- Few external components
- DIP-8 and SOIC-8 package

Description

The PTSA2511 is a CMOS LSI chip designed for toaster. Besides normal heating, it also provides defrost and reheat work modes. It provides low cost solution for toaster.

Applications

- Toaster

Figure 6

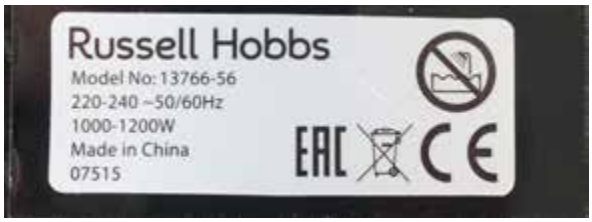


Figure 7

La puissance de notre grille-pain

Si le grille-pain ne s'allume pas du tout, même en tenant le levier baissé, la première chose à faire est de vérifier si la série des trois résistances donne la valeur souhaitée. Si la valeur est infinie, cela voudra dire que le circuit est ouvert donc que le fil de nichrome est coupé quelque part.

Dans notre cas le grille-pain chauffe donc la série n'est pas endommagée. La valeur mesurée directement à la prise est de 44,5 Ohms.

Nous pouvons utiliser Ohm pour trouver l'intensité de courant qui traverse cette série de résistances :

$I = U / R = 230V / 44{,}50\Omega = 5{,}17A$

La puissance électrique étant donnée par $P = U \times I$ nous obtenons $P = 230V \times 5{,}17A = 1188$Watts.

Effectivement sous le socle du grille-pain une étiquette nous confirme cette valeur : **Figure 7**

Les éléments chauffants

Nous avons vu plus haut que des résistances produisent de la chaleur au passage d'un courant.

Dans les grille-pains ces éléments résistifs sont constitués d'un fil de nichrome enrobé autour d'une plaque de mica.

La plaque de mica sert en tant que matière isolante et rigide pour le support du fil.

Le nichrome, un alliage de Nickel et de Chrome, confère au fil chauffant les caractéristiques qui ont rendu possible la réalisation du grille-pain électrique moderne. Il s'agit d'un matériel à très forte résistivité, 60 fois plus grande que le cuivre !

Nous essayons dans la suite de ce paragraphe un calcul qui pourrait nous conduire à des valeurs très approximatives.

Nous avons vu plus haut que la puissance électrique (P) dissipée dans un élément chauffant est donnée par : $P = U^2 / R$ où :

P est la puissance en watts (W).

U est la tension en volts (V)

R est la résistance en ohms (Ω).

La résistance R d'un élément résistif :

- Dépend de la matière, de sa caractéristique appelée résistivité
- Dépend de sa longueur, d'une manière proportionnelle
- Dépend de sa section, inversement proportionnelle

La résistance (*R*) d'un élément est donnée par la formule : $R = (\rho \times L) / A$ où :

R est la résistance en ohms (Ω)

ρ est la résistivité (Ω m)

L est la longueur du fil en mètres (m)

A est la section transversale du fil en mètres carrés (*m*2)

La résistivité ρ est une constante. Elle peut dépendre de la température.

La résistivité du nichrome à 20 °C est de $1{,}08 \times 10^{-6}$ Ω m

https://archive.wikiwix.com/cache/index2.php?url=http%3A%2F%2Fwww.goodfellow.com%2Fcsp%2Factive%2FgfMaterialInfo.csp%3FMATID%3DNI05#federation=archive.wikiwix.com&tab=url

Ce grille-pain se compose de 3 plaques de mica d'une largeur de 16 cm.

La longueur du fil sur une plaque est donc de 32 cm/tour (2 cotés *16 cm)

Le fil effectue 10 tours autour de la plaque en mica. La longueur du fil sur une plaque est de 320 cm (32 cm/tour * 10 tours)

La longueur complète du fil pour les trois plaques et de 960 cm (320 cm/plaque * 3 plaques)

J'ai volontairement isolé la section, car cela était une des deux inconnues, mais la plus compliquée à mesurer. Le fil n'a pas une forme circulaire, il est plat et son épaisseur n'est pas aisée à être mesurée.

Je peux écrire la formule comme suit : $A = (\rho \times L) / R$

$A = (1{,}08 \times 10^{-6}\ \Omega\ m \times 0{,}960\ m) / 44{,}5\ \Omega = 0{,}023 \times 10^{-6}\ m^2 = 0{,}023\ m^2$

Supposons que le fil de nichrome était de forme circulaire. Son rayon serait : $d = 2 \times R = 2 \times 0{,}086\ mm^2 = 0{,}17$ mm.

Mon calcul ne semble pas si mal que cela.

Et pourquoi ils chauffent

La chaleur produite, souvent considérée comme un inconvénient dans les circuits électroniques, est au contraire dans le cas du grille-pain le phénomène recherché. Car ce que l'on veut dans un grille-pain c'est bien produire de la chaleur. La transformation de l'énergie électrique en énergie thermique est le résultat d'un processus que nous ne décrirons pas dans cet article.

La loi qui décrit ce comportement est appelé « Loi de Joule-Lenz », elle affirme :

« La quantité d'énergie thermique produite par un courant électrique est directement proportionnelle au carré de l'intensité du courant lorsqu'il circule dans le conducteur et à la résistance que ce conducteur oppose au passage du courant »

La notation mathématique est la suivante : $Q = R \times I^2 \times t$ où :

Q – quantité de chaleur dégagée [J] ;

I – intensité du courant électrique [A] ;

R – résistance électrique du conducteur [Ω] ;

t – temps du courant électrique [s].

$Q = 44{,}5\ \Omega \times (5{,}17A)^2 \times 90s = 107040$Joule $= 107$ kJ

L'énergie thermique fournie par le grille-pain est donc de 107 kJ

L'électronique de contrôle

Grâce à la datasheet, nous disposons désormais de trois éléments de documentation fondamentale :

- Le brochage de la puce

• Le « Application circuit »
• Le « Block Diagram »

Figure 8

Je mets sous tension le grille-pain et je mesure la tension entre les pins 4 et le 8 donc entre Vcc et le GND. Je mesure 4,39 V donc la puce est alimentée avec une tension qui est conforme aux caractéristiques du composant (Operating Voltage 3,5V ~ 5V).

Toujours dans le même document je trouve l'Application Circuit : **Figure 9**

Le redressement AC/DC est opéré par D2 et C1. La tension est ensuite régulée par la diode Schottky VZ1 et stabilisée par le condensateur C3.

L'électro-aimant qui ne tient pas le crochet est représenté par l'inductance L1. Au regard du schéma il est alimenté à la sortie du bloc de redressement Diode 2 et condensateur C1 donc avec tension continue.

Je mesure une résistance de la bobine de 130 Ohm, ce qui est une valeur d'une bobine en bon état. Pour que L1 soit alimenté, le transistor Q1 doit rester en conduction. Il s'agit d'un NPN, un signal High doit lui arriver à sa Base depuis le pin 3 de la puce.

C'est Q1 qui maintient la bobine alimentée ainsi que le reste de la carte. Lorsque l'utilisateur appuie sur le bouton de relâche (SW3), la Base et l'Emetteur sont mis en court-circuit, Q1 est bloqué et aucun courant ne peut le traverser entre le collecteur et l'émetteur, l'électro-aimant relâche le crochet qui ouvre l'interrupteur bipolaire S1. À ce moment la tension Vcc devient nulle et la carte n'est plus alimentée.

Les interrupteurs SW1 et SW2 ferment le PIN1 et le PIN2 à la masse (GND). Cette action génère une impulsion négative qui active la modalité DEFROST et REHEAT.

En retour de cette sollicitation, PIN1 et PIN2 sont mis à GND. Les diodes LED sont donc alimentées pour indiquer cet état. PIN1 et PIN2 sont bien identifiés comme des I/O dans le brochage.

Jusqu'à quand la carte sera alimentée, la LED en série à R5 sera elle aussi alimentée en indiquant que le grille-pain est en état de fonctionnement.

La carte sera alimentée pendant un certain temps. Ce temps est choisi en effectuant une variation sur R6. La résistance variable R6 correspond effectivement au sélecteur extérieur qui nous permet de choisir pendant combien de temps la tartine va dorer.

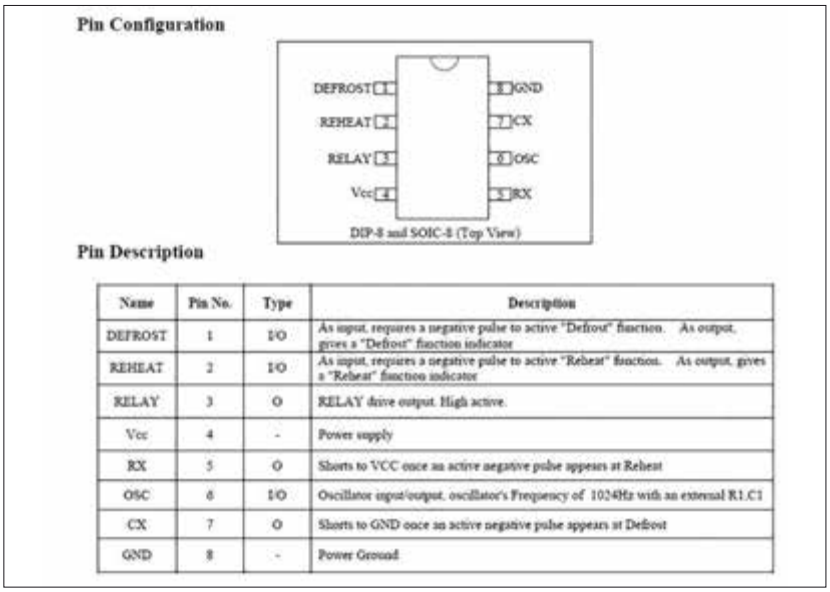
Un grille-pain clever

Ce modèle de grille-pain dispose d'une certaine forme d'intelligence. Pour utiliser un terme d'informaticien, il n'est pas stateless.

On peut remarquer que la résistance variable R6 est en série à une thermistance NTC. La photo de la carte permet de l'identifier, il s'agit du composant en haut à gauche désigné TH 474.

Sa valeur de résistance est de 470 kΩ à 25 °C. Cette valeur baisse à l'augmentation de la température ce qui fait que la résistance globale vue par les PIN 5,6 et 7 variera en fonction de la température. Plus la valeur sera petite et plus le courant qui circule sera important. Ce courant chargera plus rapidement le condensateur. Le cycle charge/décharge sera plus rapide, le temps de cuisson plus court.

Lorsque nous grillons notre première tartine, la température de la thermistance est celle de la pièce. Lorsque nous grillons une deuxième tartine juste après, le grille-pain sera chaud et la thermistance également. La puce réduira donc le temps de cuisson.



Name	Pin No.	Type	Description
DEFROST	1	I/O	As input, requires a negative pulse to active "Defrost" function. As output, gives a "Defrost" function indicator
REHEAT	2	I/O	As input, requires a negative pulse to active "Reheat" function. As output, gives a "Reheat" function indicator
RELAY	3	O	RELAY drive output. High active.
Vcc	4	-	Power supply
RX	5	O	Shorts to VCC once an active negative pulse appears at Reheat
OSC	6	I/O	Oscillator input/output, oscillator's Frequency of 1024Hz with an external R1.C1
CX	7	O	Shorts to GND once an active negative pulse appears at Defrost
GND	8	-	Power Ground

Figure 8

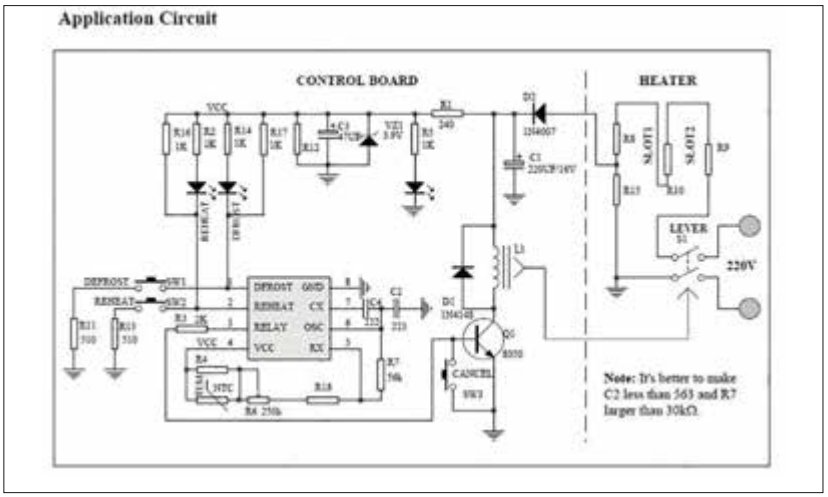


Figure 9

Le temps de cuisson

L'électronique dans ce modèle de grille-pain est physiquement implémentée sur deux PCB distincts. Celui avec la puce est en basse tension. Avec un connecteur à trois broches, j'identifie la Vcc, la GND qui arrivent depuis l'autre carte un la sortie vers l'électro-aimant. Ceci dit, l'on pourrait alimenter de manière complètement autonome cette carte.

Le fil rouge est la GND et le fil jaune est la Vcc.

Lorsque je mets le grille-pain sous tension, je mesure une tension de 10 V entre ces deux broches.

Je peux fournir à la carte cette tension en complète autonomie par une alimentation de laboratoire.

Je monte petit à petit et je m'arrête à 7 V, car elle semble déjà fonctionner.

Ma manipulation ressemble à la : **Figure 10**

Avec le potentiomètre positionné à la valeur la plus petite, sur

PIN3 je mesure 3,42 V pendant 1 minute et 45 secondes.
Avec le potentiomètre positionné à la valeur la plus haute, le PIN3 passe à 0V après 4 minutes et 4 secondes.
À l'aide d'un sèche-cheveux, je chauffe légèrement la thermistance.
J'effectue une deuxième fois la mesure avec le potentiomètre dans la position la plus petite, la tension chute à 0 V en 1 minute et 35 secondes. Le temps est réduit, c'est merveilleux !

L'oscillateur

Le Pin 6 indique l'entrée/sortie d'un oscillateur et la description indique que sa fréquence dépendra des valeurs d'une résistance et d'un condensateur extérieurs.
Avec l'oscilloscope je pique sur le PIN 6 et je trouve un signal en dents de scie.
Avec le potentiomètre dans la position la plus petite j'obtiens : **Figure 11**

Figure 10

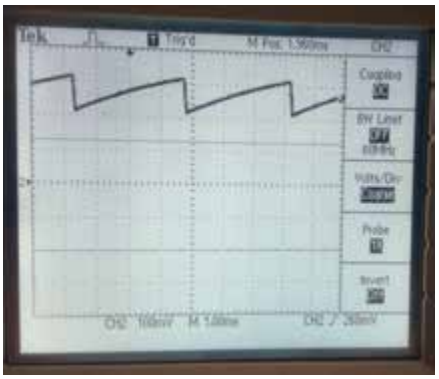


Figure 11

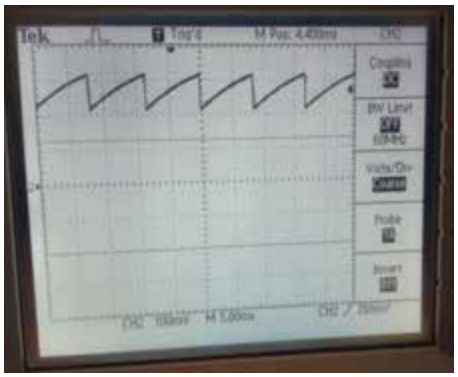


Figure 12

La base de temps est sur 1 ms sec/div.
L'amplitude est de 3,5 divisions.
L'amplitude de la période est donc de 3,5 div * 1 ms/div = 3,5 ms
La fréquence du signal est de 285 Hz (1/(35 * 10^3))
Avec le potentiomètre dans la position la plus haute j'obtiens : **Figure 12**
La base de temps est sur 5 ms sec/div.
L'amplitude est de 1,6 division.
L'amplitude de la période est donc de 1,6 div * 5 ms/div = 8 ms
La fréquence du signal est de 125 Hz.
Cette mesure a permis de vérifier que l'oscillateur modifie la fréquence en fonction de la position du potentiomètre. La fréquence augmente lors de l'augmentation de la valeur de la résistance.
À l'aide d'un tableur, j'insère les mesures effectuées aux extrêmes et j'obtiens : **Figure 13**
En colonne C je calcule le coefficient de cette droite.
La datasheet de la puce PT8A2511 ne contient pas d'indications particulières liées aux usages. En revanche d'autres puces qui servent dans les mêmes situations disposent de datasheets parfois plus complètes. Il y en a une pour laquelle la datasheet est riche, c'est la ga5210ph.
Hélas je ne la trouve qu'en chinois et dans un paragraphe l'on retrouve cette formule : **Figure 14**
À l'aide d'un traducteur en ligne j'obtiens : **Figure 15**
Vous remarquerez la présence d'une constante 24350. Elle est du même ordre de grandeur du coefficient de la droite.
Le Block Diagram suivant montre cet oscillateur ainsi qu'un diviseur de fréquence en série. **Figure 16**
Proposition de fonctionnement :

- Le diviseur de fréquence (Frequency Divider dans le Block Diagram) vient compter le nombre d'oscillations.
- Au bout de 24350 oscillations, ce composant va déclencher la commande de repos via la broche de sortie « RELAY ».
- Plus la fréquence du signal est haute plus le nombre d'oscillations est haute dans une période constante. Atteindre les 24350 oscillations sera d'autant réduit.

La réparation

Au regard des vérifications effectuées, le problème ne pouvait être que sur le transistor Q1 ou sur l'électro-aimant. Je

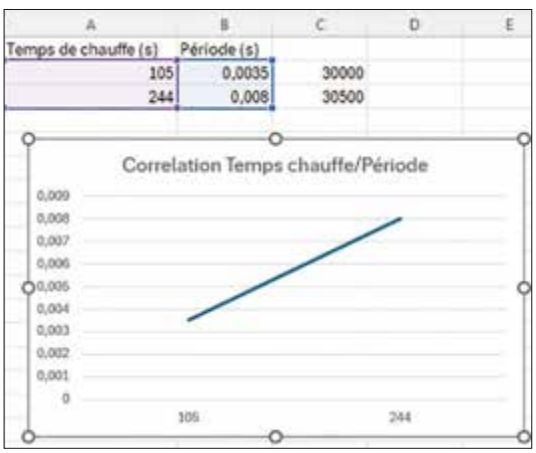


Figure 13

正常定时时间的计算公式为：T=24350×Rtotal×C5，其中 Rtotal=（RT1//R6）+（VR1//R5）+R7+R8
REHEAT 定时时间的计算公司为：T= 24350×R8×C5
DEFROST 定时时间的计算公式为：T=24350×Rtotal×（C5+C4）
更为精确的定时时间计算方法如下：
T=32×1024×（1/fRC）。该公式在任何情况下均有效，定时时间可以根据实际档位对定时时间的要求加以调整。
❖ 高压方案 2

Figure 14

The simple calculation method of timing time is as follows:

The calculation formula for normal timing time is: T = 24350 × Rtotal × C5, where Rtotal = (RT1//R6) + (VR1//R5) + R7 + R8

The calculation formula of REHEAT timing time is: T = 24350 × R8 × C5 The

calculation formula of DEFROST timing time is: T = 24350 × Rtotal × (C5 + C4)

A more accurate timing calculation method is as follows:

T=32×1024×(1/fRC), this formula is valid in any case. The timing time can be adjusted according to the actual gear position.

Figure 15

vérifie les deux jonctions du Q1, le transistor est en bon état. [Les différents retours m'ont fait remarquer je j'aurais pu illustrer comment j'ai effectué le test du transistor à l'aide d'un simple multimètre. Je garde précieusement ce conseil pour les prochains articles.]

La photo suivante met en évidente l'électro-aimant : **Figure 17**

La poussière des tartines se dépose entre la partie fixe et la partie mobile ce qui empêche cette dernière de rester collée lorsque la bobine est alimentée.

J'ai nettoyé très soigneusement l'extérieur de l'électro-aimant à l'aide d'alcool isopropylique. Si vous n'en avez pas, un coup d'air comprimé suffira.

Ensuite il faut ouvrir la gâchette et nettoyer les impuretés qui y seront sûrement déposées. C'est tout ce qu'il fallait faire.

Bien sûr il ne faut pas passer par toutes ces étapes. Dans une situation classique pendant un temps de réparation au Repair Café, la vérification de ceux deux composants est la première chose à faire et souvent la seule, car le nettoyage suffit dans la plupart des cas.

L'électro-aimant

Étant donné que la panne concernait exclusivement l'électro-aimant, j'ai pensé qu'il serait utile de faire un rappel de son fonctionnement.

Comme son nom l'indique, ce n'est rien d'autre qu'un aimant électrique, un aimant piloté. Ce dispositif permet une conversion de l'énergie électrique en énergie mécanique.

Le courant électrique parcourant la bobine produit un champ magnétique qui déterminera une force d'attraction.

Les électro-aimants peuvent être classés par catégorie. Cette catégorie est déterminée principalement par leur géométrie. Les réversibles, les bistables, les rotatifs, etc.

Celui que l'on retrouve dans un grille-pain est de type proportionnel. Proportionnel, car la force magnétique est proportionnelle au courant circulant dans la bobine, même si dans cet usage la force sera constante étant donné que la courant ne varie pas.

Les électro-aimants peuvent avoir deux modes de fonctionnement, le tirant et le poussant. Celui dans un grille-pain est dans une configuration de mode tirant, car le flux magnétique génère une force qui attire l'armature mobile. Lorsque la bobine ne sera plus alimentée, l'armature est relâchée.

Des formules, un peu compliquées je l'avoue, permettent de déterminer cette force d'attraction. L'on peut arriver à calculer la force d'aimantation.

Cette force est due à différents facteurs, dont la finition de la pièce à aimanter, les matériaux la composant.

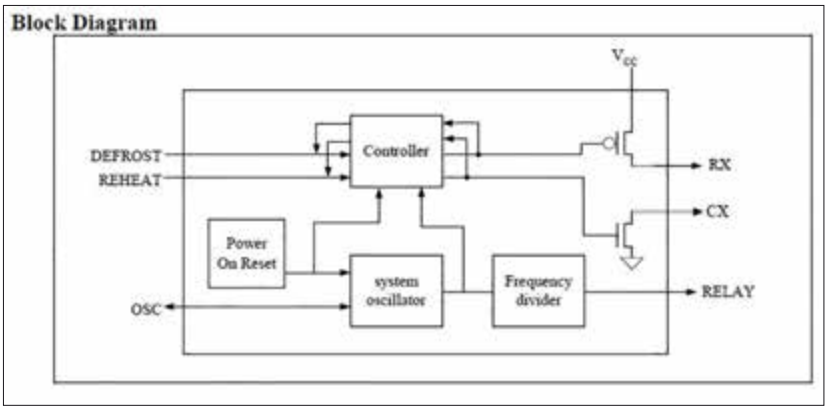
Le dépôt de poussière vient modifier la surface. C'est comme si la finition changeait donc son adhérence. Ce changement réduit la puissance.

À l'aide d'un dynamomètre, il serait possible de mesurer cette force, avant et après nettoyage. Je conserve cette expérience pour la prochaine réparation.

La consommation

Je ne sais pas vous, mais moi j'aime bien lorsque la tartine est brune, mais pas cramée. Pour ma tartine de pain complet, j'obtiens ce résultat au bout de 1 minute et 30 seconds environ. Et j'en grille toujours deux en même temps, car j'aime bien varier la confiture, mais avec bien sûr du beurre salé sur les deux.

Figure 16



Constantes : 1 heure = 60 minutes = 3600 secondes

Si 1 heure / 60 minutes = 90 seconds / x heures

Xheures = 90 s / 3600 s = 0,025 heure

L'énergie électrique est représentée par la formule E = P * t donc l'énergie utilisée pour dorer ma tartine sera de E = 1,188 kW x 0,025 h = 0,0297 kWh.

Avec le contrat dont nous disposons avec le fournisseur, le prix de l'énergie est de 0,2276 €/kWh.

Si je déjeune tous les jours de l'année à la maison, j'aurai un coût global de tartines dorées de 2,5 € (0,068 €/jour x 365 jours) !

Figure 17

Figure 18

Le coût de l'électricité

Nous avons vu que pour dorer mes deux tartines le grille-pain aurait consommé 0,0297 kWh (29,7 Wh) et que l'énergie thermique fournie a été de 107 kJ.

C'est peu, pas beaucoup, beaucoup ? Tout dépend de comment l'on prend les choses.

Pendant que j'ai fait dorer mes tartines j'étais en train de m'occuper avec d'autres activités, à aucun moment j'ai dû m'occuper de comment l'énergie électrique était fournie au grille-pain. C'est mon fournisseur d'électricité qui le faisait à ma place.

Est-ce que j'aurais été capable par mes propres moyens de produire autant d'énergie électrique pour que je puisse griller la tartine ?

À ce propos je vous invite à regarder la vidéo qui raconte l'expérience produite à Stockholm avec le cycliste professionnel Robert Förstemann. La vidéo est disponible ici https://youtu.be/S4O5voOCqAQ?si=W17EVldJcVlh94dv

Comme vous l'aurez compris, la réponse est négative.

Conclusion

La réparation a permis de remettre en état de fonctionnement un grille-pain qui autrement aurait vu la déchetterie comme seule destination possible. J'espère que la lecture de cette fiche vous a plu et qu'elle vous permettra de réparer facilement le grille-pain qui traîne dans un coin de votre grenier.

Un remerciement particulier à Philippe Dessoulles et à Philippe Le Guen qui inexorablement effectuent une lecture de mes articles tout en apportant des remarques pertinentes et constructives. Ainsi qu'à mon ami Jérémy pour avoir fourni le protagoniste de cette réparation, le grille-pain.

Jérôme Cilly
CTO France et l'équipe IroCO2 – Ippon Technologies

IroCO2 : Mesurer pour transformer, agir pour réduire – vers un Cloud plus responsable

Le numérique représente 4 % des émissions de gaz à effet de serre dans le monde et risque de voir son chiffre doubler d'ici la fin de l'année. (source : Arcep, 2025). Plusieurs causes expliquent cela : l'explosion des usages Cloud, la croissance exponentielle des données, la multiplication des objets connectés et leur renouvellement prématuré ainsi que l'essor de l'IA générative.

Voici quelques données :

- Le volume de données numériques double tous les 2 ans *(source : Oracle, qu'est-ce que le big data ?)*
- L'IA générative peut générer jusqu'à 500 tonnes de CO pour l'entraînement d'un seul modèle comme GPT-3 *(source : AIAAIC 2022)*
- Le nombre d'objets connectés dépassera 29 milliards d'ici 2030, avec un impact énergétique conséquent *(source : Statista 2025)*
- À lui seul, le cloud mondial consomme plus d'électricité que l'Allemagne, et son empreinte ne cesse de croître *(source : carbone4 2024)*

Cette croissance est d'autant plus préoccupante quand on regarde de près l'usage réel des ressources cloud :

70 à 90 % des ressources cloud dans les grandes entreprises sont surdimensionnées ou inutilisées. (source : Flexera, 2024)

Face à ce constat, il devient urgent d'agir. Le Green IT ne peut plus être un sujet mis de côté : c'est un vrai levier pour rendre les entreprises plus durables, plus efficaces et plus responsables.

Pour les DSI, c'est une opportunité de jouer un rôle clé dans la transition écologique, tout en optimisant les coûts et en anticipant les futures réglementations.

Chez Ippon Technologies, on pense que cette transition doit passer par des outils simples, concrets et efficaces. C'est pour ça qu'on a créé IroCO2 : une solution pour aider les équipes tech à estimer l'impact carbone de leurs infrastructures cloud, à l'analyser, et surtout à l'améliorer dans la durée.

Parce que faire du numérique responsable, ce n'est pas juste une bonne intention : c'est un chantier qu'on peut piloter, suivre et réussir – ensemble.

IroCO2 : Un outil simple et modulaire pour un Cloud plus vert

IroCO2 est un outil conçu pour évaluer et anticiper l'empreinte carbone des infrastructures cloud AWS composé de plusieurs fonctionnalités.

1 Simuler l'empreinte carbone avant de déployer : la force de la calculatrice IroCO2

La **calculatrice carbone** d'IroCO2 est un simulateur pré-déploiement pensé pour les architectes et développeurs cloud. Elle permet de modéliser une infrastructure théorique et d'en estimer l'impact carbone en quelques clics, avant même la première ligne de provisionnement.

L'outil repose sur un moteur d'estimation multi-paramètres prenant en compte :

- la configuration des ressources (vCPU, RAM, stockage, type de CPU),
- leur usage prévu (durée, fréquence, réplication),
- et surtout la **localisation géographique**, avec prise en compte du **mix énergétique local** (répartition des différentes sources d'énergies consommées dans une zone géographique).

Chaque composant peut être personnalisé et comparé dans plusieurs scénarios : types d'instances, régions alternatives, profils d'usage. À la clé, une **projection claire de l'empreinte carbone mensuelle** (en CO eq), enrichie d'**équivalents concrets** (trajet voiture, avion…) et de **recommandations d'optimisation**, comme la **comparaison automatique entre régions** pour identifier celles à plus faible intensité carbone.

C'est une **approche design-first de la sobriété numérique**, qui permet d'**intégrer l'empreinte carbone dans les choix d'architecture** au même titre que le coût ou la résilience. IroCO2 transforme ainsi la contrainte environnementale en **levier d'optimisation technique**. Elle peut être adoptée très facilement par les architectes cloud de par sa ressemblance avec la simulation de coûts des grands cloud providers. **Figures 1 et 2.**

Ce simulateur constitue une première dans l'écosystème cloud : selon le Head of Sustainability d'AWS France, aucune autre solution ne propose aujourd'hui d'évaluer les émissions carbone avant déploiement des infrastructures Cloud AWS. C'est précisément cette approche préventive, intégrée dès la phase de design d'architecture, qui distingue IroCO2 des autres outils disponibles.

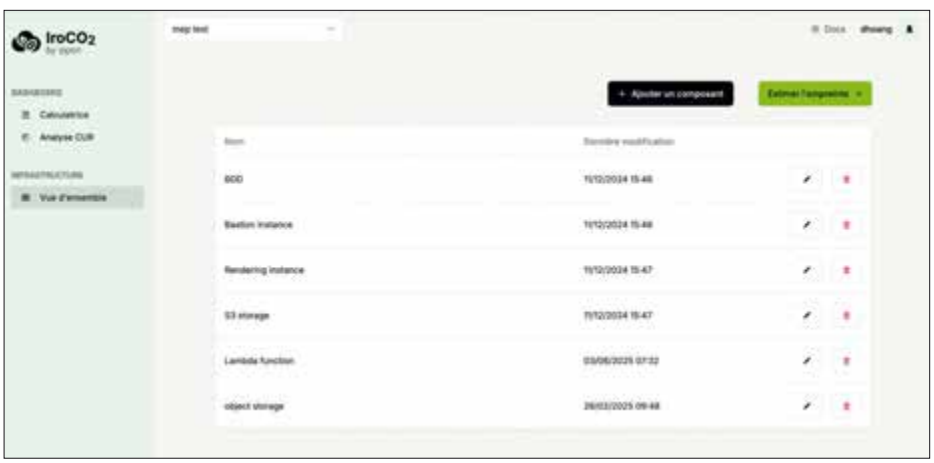


Figure 1 : *Interface de gestions des composant dans la calculatrice carbone IroCO2*

2 Mesurer l'impact carbone réel de son infrastructure : la puissance d'analyse du CUR Analyzer

Le CUR Analyzer d'IroCO2 permet de visualiser concrètement l'empreinte carbone de son infrastructure AWS après déploiement. Il s'appuie sur le rapport de facturation détaillé d'AWS, le Cost and Usage Report (CUR), que l'utilisateur peut facilement importer dans l'outil.

À partir de ce rapport, IroCO2 reconstitue l'impact carbone de chaque ressource utilisée : machines virtuelles, stockage, fonctions serverless, etc. L'analyse prend en compte la localisation des services, leur durée d'usage et leur configuration, pour produire une estimation fidèle des émissions.

Les résultats sont ensuite triés et présentés par service (EC2, S3, Lambda…), par type de ressource, par région, mais aussi selon les balises utilisées dans le compte (projet, équipe, environnement…).

Cette vision détaillée permet d'identifier les postes les plus émetteurs, de responsabiliser les équipes et de guider les efforts de réduction là où ils auront le plus d'impact. Avec le CUR Analyzer, chaque ligne de facture devient une opportunité d'optimisation environnementale.

Figure 2: *Estimation de l'impact carbone prévisionnel d'une infrastructure sur la calculatrice carbone IroCO2*

3 Suivi carbone en continu : le Scanner IroCO2 (Alpha)

Complémentaire du CUR Analyzer, le **Scanner IroCO2** apporte une nouvelle dimension à l'analyse environnementale des infrastructures cloud : la **collecte automatisée et le suivi en continu des émissions**.

Là où une analyse classique repose sur l'import manuel de rapports, le Scanner s'installe directement sur un compte cloud pour suivre l'évolution des ressources et de leur impact carbone, de manière autonome. Il exécute trois étapes clés :

1. **Collecte des données d'usage** pertinentes (type de ressource, durée d'exécution, volumes mobilisés),
2. **Transmission sécurisée** des informations essentielles,
3. **Analyse et visualisation** des résultats dans un tableau de bord centralisé, avec un suivi temporel des émissions.

Cette approche permet d'**identifier rapidement les écarts** (ressources inactives, surdimensionnement, pics liés à des déploiements) et de **mesurer l'effet concret des actions correctives** (extinction automatique, migration, redimensionnement).

Cette brique s'inscrit dans une stratégie d'observabilité continue, indispensable à l'industrialisation d'une démarche GreenOps.

Une méthodologie transparente, fondée sur des sources ouvertes

IroCO2 repose sur un modèle hybride d'estimation, combinant plusieurs sources de données reconnues :

- Les coefficients d'émission par service cloud (issus de projets comme Cloud Jewels d'Etsy)
- Les données régionales de mix énergétique (fournies par des sources fiables telles que l'ADEME ou Ember Climate)
- Les facteurs d'efficacité énergétique (PUE - Power Usage Effectiveness) des datacenters
- Les caractéristiques techniques des ressources cloud (nombre de vCPU, mémoire, type de CPU, TDP)

L'ensemble des formules, sources et hypothèses est **documenté publiquement** sur https://docs.greensuite.fr. Ce choix garantit l'**auditabilité** du modèle, sa **reproductibilité** dans d'autres contextes, et surtout son **évolutivité** : chacun peut proposer des améliorations, contribuer à l'enrichissement du référentiel ou adapter la logique à ses propres besoins.

Exemple d'usage : 50 % d'économie, 90 % d'émissions évitées

Lors d'un projet interne, IroCO2 a permis de diagnostiquer une surconsommation significative :

- Environnements de test actifs 24/7 sur des instances r5.4xlarge (EC2)
- Déploiement en Irlande (400-500 g CO2/kWh)
- Résultat : 200 kg CO e/mois, 11 000 € de coût

Actions correctives menées :

- Migration en région Paris (20 g CO /kWh)
- Redimensionnement des instances vers Graviton2 (ARM)
- Automatisation de l'extinction des environnements non critiques

Bilan : une baisse de la facture de 50 % et une division par 30 des émissions, sans impact fonctionnel.

Bien plus qu'un outil RSE

IroCO2 n'est pas un outil de reporting RSE, mais un **levier d'optimisation opérationnelle**. Son intégration dans les workflows DevOps, FinOps, ou GreenOps permet :

- En conception : arbitrer architectures, types d'instances, régions
- En exécution : monitorer les émissions en continu
- En pilotage : définir et suivre des objectifs d'impact par projet ou produit

L'objectif est de rendre les arbitrages environnementaux aussi naturels que ceux liés au TCO (Total Cost of Ownership), à la performance ou à la résilience.

Open source, souveraineté et collaboration : une ambition ouverte pour un numérique plus durable

Nous sommes convaincus que les enjeux environnementaux liés au cloud exigent des réponses collectives, transparentes et accessibles. C'est pourquoi nous nous engageons à rendre **IroCO2 open source à moyen terme**, dans une logique d'amélioration continue portée par la communauté.

IroCO2 est un outil toujours en cours de développement et cette démarche ne se limite pas à une simple publication de code : c'est une volonté stratégique de **favoriser la collaboration**, **accélérer l'innovation** et **bâtir un socle commun** pour le calcul d'impact environnemental dans le cloud.

En rendant IroCO2 ouvert :

- Nous permettons à chacun de **vérifier, adapter et enrichir** les modèles de calcul d'empreinte carbone.
- Nous offrons une base technique pour **intégrer IroCO2 dans des outils tiers** (CI/CD, FinOps, observabilité, ESG...).
- Nous contribuons à l'émergence d'un **standard européen ouvert** de calcul d'impact Cloud, au croisement du Green IT, de la souveraineté numérique et de la responsabilité collective.

La documentation et les démonstrateurs sont déjà accessibles sur https://greensuite.fr.

IroCO2, c'est une brique technologique que nous voulons **utile, fiable et partageable**, pour que chaque équipe tech puisse s'en emparer et agir.

Conclusion : mettre les mains dans le CO2

Avec IroCO2, Ippon apporte une réponse opérationnelle aux défis du Green IT et transforme la contrainte carbone en **opportunité d'optimisation**, en **levier d'innovation** et en **critère de pilotage** à part entière. Pensé pour les techs, par des techs, l'outil permet de remettre la **performance environnementale au centre des pratiques d'ingénierie logicielle et cloud**.

Mais au-delà de l'outil, c'est une **nouvelle posture** que nous défendons : celle de techniciens et techniciennes qui prennent leur part dans la transition écologique, sans attendre.

Car mesurer, c'est comprendre. Comprendre, c'est pouvoir agir.

Et agir, c'est ce que nous proposons de faire — ensemble.

Mesurer. Comprendre. Agir. Et surtout, ouvrir la voie.

Kévin ANSARD
consultant Diggers en développement backend.

Eco-conception : le langage au cœur du problème

Le numérique et son utilisation son invisible à nos yeux, il ne fume pas, ne rouille pas et ne dégrade pas nos plages, pourtant le numérique pollue et beaucoup. Si nous parlons de chiffres étant des ingénieurs nous les aimons tout particulièrement, le numérique représente environ 4% des émissions des gaz à effet de serre, ce chiffre vous paraît sûrement petit. Je vous propose donc de le comparer à plusieurs indicateurs.

- Si le numérique était un pays, nous serions le 5e plus gros pollueurs derrière la Chine, les États-Unis, l'Inde et la Russie.
- Il émet plus que l'ensemble de l'aviation civile ou bien correspondrait à la totalité des émissions de gaz des voitures d'un pays comme l'Allemagne

Mais le pire ici c'est que son empreinte croit chaque année de 6% à 9% depuis 2013. Vous me direz en quoi cela nous concerne ? En tant que développeurs nous écrivons le code et que celui joue un rôle central dans cette pollution. Chaque ligne de code, chaque architecture mal pensée, chaque appel non optimisé consomme de l'énergie, alors oui à l'échelle d'une api cela ne représente rien, mais à l'échelle mondiale cet impact est énorme.

Je vous rassure, je n'écris pas cet article pour vous culpabiliser ou pour vous faire une leçon. Je l'écris uniquement pour vous montrer qu'avec des choix simples et d'autres, un peu plus complexes, nous pourrions énormément réduire l'émission du numérique. En gros je vais vous parler d'Eco-conception.

Mais c'est quoi l'éco-conception ? L'idée est de mieux penser nos applications, chacune des couches de notre application doit être pensée pour être optimisée et consommer le moins de ressources possible. Le but en somme est de moins polluer.

GreenDev : au cœur d'un code responsable

Nous commettons souvent l'erreur de penser que la pollution incombe uniquement à l'infrastructure et à sa consommation, évidemment si tous les data-centers du monde fonctionnaient avec une énergie verte, le problème serait résolu et cet article n'aurait aucun sens.

Mais ce qui est important à comprendre ici c'est que l'utilisation énergétique des infrastructures de nos applications est impactée directement par notre code, le développeur est donc un acteur clé dans le processus d'écoconception.

L'objectif majeur pour nous est de produire du code mieux pensé, plus clair et forcément plus optimisé. Pour cela nous devons respecter trois étapes :

1. La sobriété numérique (écrire moins, mieux sans fausses notes)
2. Efficacité énergétique (optimisé l'architecture et le code de nos applications)
3. La longévité de nos logicielles (moins de dette technique)

Nous devons nous poser les bonnes questions. La première ? Quel langage de programmation et quel framework sont utilisés ?

Langages, Framework et consommation

Dans les faits je pourrais vous dire de prendre un langage connu pour avoir de hautes performances, ce qui impacterait directement la dépense énergétique de vos applications, mais je vous propose ici d'aller un peu plus loin dans l'analyse et de découvrir comment choisir le bon langage.

Langage compilé ou interprété :

- **Les langages compilés** (ex. Rust, Go, C) sont **transformés directement en code machine**. Ils s'exécutent rapidement, sans surcouche inutile.
- **Les langages interprétés** (Python, Ruby, PHP…) ont besoin d'un interpréteur qui **traduit le code ligne par ligne à chaque exécution**, ce qui **consomme de la CPU, de la RAM, etc.**

En gros, moins de cycles CPU = moins de consommation énergétique. Jusque-là, rien de très surprenant. Mais au-delà du type de langage, il faut aussi faire attention à la mémoire. Comme vous le savez, beaucoup de langages utilisés dans le développement web aujourd'hui ont un garbage collector intégré. C'est pratique, on ne va pas se mentir, mais ça a un coût en ressources. Et il existe des alternatives plus sobres, comme Rust, qui permettent de gérer la mémoire sans garbage collector.

Pourquoi c'est mieux ? Parce qu'un langage qui gère mieux la mémoire limite les accès disques (les fameux swap), réduit la charge CPU et stabilise l'ensemble. Encore une fois, sur une seule app, ça peut sembler négligeable, mais à l'échelle de plusieurs millions d'utilisateurs ou de milliers d'instances en production, ça fait une vraie différence. Et si on pousse encore plus loin, à l'échelle de toute une industrie ou d'une planète, l'impact devient clairement visible.

Et ce raisonnement ne s'arrête pas au langage. Il faut aussi se pencher sur les frameworks qu'on utilise. Si vous prenez un framework super lourd, avec plein de dépendances inutiles, comme Express pour Node.js, votre app part déjà avec un handicap. C'est pareil côté frontend. Oui, je sais, c'est plus compliqué parce qu'on veut une UI/UX jolie, fluide, responsive… mais quand on charge plusieurs mégas de JavaScript pour afficher une simple page, c'est qu'il y a un souci.

Alors, quel langage choisir ? La réponse simple serait : Rust, ou C. Mais soyons honnêtes : il faut que ça colle à votre projet. Oui, Rust est top pour l'écoconception, mais il est plus

complexe à apprendre, et ce n'est pas évident de convaincre toute une équipe de faire la bascule. Donc l'idée n'est pas de tout réécrire en Rust, mais plutôt de faire des choix raisonnés : un langage sobre, un framework léger, un code structuré, un front optimisé, et des outils adaptés à la réalité de votre projet. Voilà ce que vous devez garder en tête.

Et au-delà du langage ou du framework, le code en lui-même reste central. Je ne vais pas vous faire la leçon sur les boucles imbriquées, les tris manuels ou les filtres post-requête SQL — ça, c'est la base. Un bon algo bien pensé fera toujours la différence. Mais on peut aussi aller plus loin avec quelques principes simples à intégrer :

- Ajouter de la **pagination** et du **filtrage** sur vos endpoints
- Mettre en place un **cache** pour éviter les recalculs ou les requêtes inutiles

Selon l'app, ce n'est pas toujours si simple à implémenter. Ça demande un peu d'effort, mais au-delà du gain écologique, vos applications seront aussi plus rapides, plus fluides, et tourneront sur des infrastructures plus légères. Donc double bonus.

On sait maintenant quels langages privilégier, quoi ajouter dans notre code, mais la vraie question c'est : **comment on mesure l'impact de tout ça ? Comment savoir si ça fonctionne vraiment ?** Et surtout, comment suivre les perfs de nos applis dans le temps ?

Mesurer pour progresser

Je vous propose maintenant un peu de pratique, le but ici est étayer mon propos en comparant 2 API Rust une avec les bonnes pratiques et l'autre non.

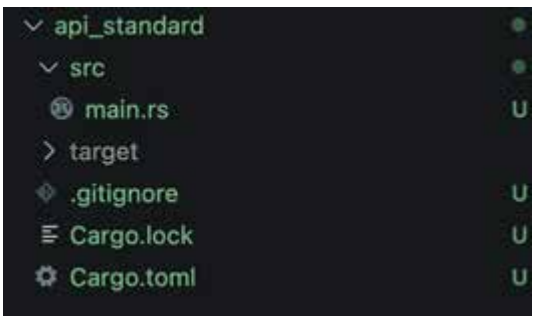
Voici les prérequis pour pouvoir reproduire ce test.

- Rust
- Cargo
- Wrk (outils de benchmarking)
- Hyperfine (outils de benchmarking)

Tout d'abord nous allons créer l'api standard, pour ce faire taper la commande suivante :

```
cargo new api_standard
```

Cette commande va créer un dossier comme l'image ci-dessous :



Une fois cela fait, ajouter le code suivant dans le fichier Cargo.toml, ce fichier nous permet de référencer les librairies utilisées dans notre projet.

```
actix-web = "4"
```

Votre fichier devrait ressembler à cela :

```
api_standard > ⚙ Cargo.toml
1   [package]
2   name = "api_standard"
3   version = "0.1.0"
4   edition = "2024"
5
6   [dependencies]
7   actix-web = "4"
```

Enfin remplacer le code dans le fichier main.rs du dossier src par celui-ci :

```
use actix_web::{web, App, HttpServer, HttpResponse, Responder};

async fn get_users() -> impl Responder {
  let users = (0..10_000)
    .map(|i| format!("User {}", i))
    .collect::<Vec<_>>();
  HttpResponse::Ok().json(users)
}

#[actix_web::main]
async fn main() -> std::io::Result<()> {
  HttpServer::new(|| App::new().route("/users", web::get().to(get_users)))
    .bind("127.0.0.1:8080")?
    .run()
    .await
}
```

Pour faire simple, la première ligne permet d'importer **d'Actix web** : nécessaires pour construire un serveur HTTP, gérer les routes, les réponses, etc.

La fonction get_users() génère 10000 utilisateurs et renvoie la liste au format JSON et la fonction main permet de lancer le serveur HTTP sur le port 8080.

Pour finir, il vous suffit d'exécuter la commande suivante dans votre terminal pour lancer l'api standard.

```
cd api_standard
cargo run
```

Vous devriez voir cela s'afficher dans votre terminal : **figure 1**

Passons maintenant à l'api green ici notre but va d'être de prendre la même base, mais de rajouter les principes vus précédemment en ajoutant les points suivants :

- Compression automatique
- Structure de données sérialisables
- Cache en mémoire
- Thread-Safe

Idem que tout à l'heure taper la commande qui suit pour crée le dossier de l'api :

```
cargo new api_green
```

Figure 1

```
fisk@Host-003 api_standard % cargo run
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.08s
     Running `target/debug/api_standard`
```

Ajouter le code suivant dans le fichier Cargo.toml pour rajouter les dépendances du projet :

```
[dependencies]
actix-web = "4"
serde = { version = "1", features = ["derive"] }
once_cell = "1.18"
tokio = { version = "1", features = ["rt-multi-thread", "macros"] }
```

Enfin ajouter le code suivant dans le fichier main.rs nous verrons ensuite les différentes parties du code :

```
use actix_web::{web, App, HttpServer, HttpResponse, Responder, middleware::Compress}; use serde::{Serialize, Deserialize}; use std::collections::HashMap; use once_cell::sync::Lazy; use tokio::sync::RwLock;
#[derive(Serialize, Clone)] struct User { id: u32, name: String, }
#[derive(Deserialize)] struct Pagination { page: Option, per_page: Option, }
static USER_CACHE: Lazy<RwLock<HashMap<String, Vec>>> = Lazy::new(|| RwLock::new(HashMap::new()));
async fn get_users(query: web::Query) -> impl Responder { let page = query.page.unwrap_or(1); let per_page = query.per_page.unwrap_or(20); let key = format!("{}:{}", page, per_page);
if let Some(result) = USER_CACHE.read().await.get(&key) {
  return HttpResponse::Ok().json(result.clone());
}

let start = (page - 1) * per_page;
let users = (start..start + per_page)
  .map(|i| User { id: i, name: format!("User {}", i) })
  .collect::<Vec<_>>();

USER_CACHE.write().await.insert(key, users.clone());

HttpResponse::Ok().json(users)

}
#[actix_web::main] async fn main() -> std::io::Result<()> { HttpServer::new(|| { App::new() .wrap(Compress::default()) .route("/users", web::get().to(get_users)) }) .bind("127.0.0.1:8081")? .run() .await }
```

Dans ce code nous avons plusieurs parties :

```
#[derive(Serialize, Clone)]
struct User {
  id: u32,
  name: String,
}

#[derive(Deserialize)]
struct Pagination {
  page: Option<u32>,
  per_page: Option<u32>,
}
```

Celle-ci nous permet de créer une structure pour nos utilisateurs et de crée notre structure de Pagination.

```
static USER_CACHE: Lazy<RwLock<HashMap<String, Vec<User>>>> = Lazy::new(|| RwLock::new(HashMap::new()));
```

Notre variable ici permet de crée le cache sans DB bien entendu. Notre cache sera de la forme suivante :
clé : "1:20"
valeur : Vec<User> pour la page 1 avec 20 éléments.
Enfin la fonction get_user() a été modifiée pour faire la pagination ainsi que la gestion du cache et la main pour autoriser les compressions.
De même que pour la première api vous devez taper les commandes suivantes pour lancer votre application :

```
cd api_green
cargo run
```

Comparaisons

Comparons les API avec les outils de benchmark le but est de voir nos performances évoluées et à quel point elle évolue.
Pour commencer, nous allons utiliser l'outil "wrk" :
Taper dans votre terminal la commande qui suit pour voir les performances de l'api standard :

```
wrk -t4 -c100 -d10s http://127.0.0.1:8080/users
```

Cette commande devrait vous donner un résultat similaire à celui-ci : **Figure 2**

Je vous ferais un tableau de comparaison à la fin, mais les informations qui nous intéressent ici sont les suivantes :

- Requêtes par seconde
- Latence moyenne
- Volume de transfert

Figure 2

```
fisk@Host-003 GreenIT % wrk -t4 -c100 -d10s http://127.0.0.1:8080/users
Running 10s test @ http://127.0.0.1:8080/users
  4 threads and 100 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    29.64ms    7.47ms  63.40ms   68.56%
    Req/Sec   845.71     35.22     1.00k    73.93%
  33860 requests in 10.08s, 3.75GB read
Requests/sec:   3358.58
Transfer/sec:    381.17MB
```

```
fisk@Host-003 GreenIT % wrk -t4 -c100 -d10s "http://127.0.0.1:8081/users?page=1&per_page=20"
Running 10s test @ http://127.0.0.1:8081/users?page=1&per_page=20
  4 threads and 100 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency     0.98ms    4.00ms 125.84ms   98.40%
    Req/Sec    40.06k     5.68k  116.47k    95.76%
  1600363 requests in 10.10s, 0.94GB read
Requests/sec: 158449.82
Transfer/sec:     95.20MB
```

Figure 3

- Total de requête exécuter dans le temps imparti
- Données totales lues

Avant de décortiquer nos performances je vous propose d'abord de voir en détail notre commande :

- -t4 => nombres de thread de connexion
- -c100 => nombres de connexions en simultané
- -d10s => le temps du test

Taper la même commande pour notre seconde api :

```
wrk -t4 -c100 -d10s http://127.0.0.1:8081/users?page=1&per_page=20
```

Figure 3

Comparons maintenant les résultats :

Critères	API standard	API Green	Résultats
Requêtes/s	3 358	158 449 req/s	+4618.55 %
Latence moyenne	29,64 ms	0,98 ms	−96 %
Volume transféré/s	381,17 MB/s	95,20 MB/s	−75 %
Total de requêtes	33 860	1 600 363	
Données totales lues	3,75 GB	0,94 Go	

Analyses et interprétations

1 Performance explosive pour l'API éco-conçue

- L'API green a permis de traiter 47 fois plus de requêtes par seconde, grâce à la pagination, compression, et cache.
- Elle tient mieux la charge, ce qui signifie moins de serveurs nécessaires pour le même trafic.

2 Latence réduite de 96 %

- L'API standard renvoie un énorme JSON (10 000 utilisateurs), ce qui ralentit tout : sérialisation, envoi réseau, parsing côté client.
- L'API éco-conçue renvoie seulement 20 utilisateurs paginés, compressés donc réponse quasi instantanée.

3 Consommation de bande passante drastiquement réduite

- L'API standard envoie ~3,75 Go en 10 secondes, soit ~381 Mo/s de charge réseau.
- L'API green n'envoie que 95 Mo/s, pour beaucoup plus de requêtes, grâce à :
  - JSON allégé
  - Compression Gzip/Brotli
  - Requêtes ciblées

4 Sobriété énergétique implicite

- Moins de calculs, moins de transferts, moins de charge mémoire et réseau cela signifie moins d'énergie consommée par le serveur et le client.
- C'est l'objectif fondamental de l'éco-conception.

L'API éco-conçue est plus rapide, plus scalable, plus sobre, plus propre – sans compromis fonctionnel. Le même benchmark aurait pu être fait avec l'outil Hyperfine, les résultats auraient été les mêmes.

Donc si nous reprenons en ajoutant simplement quelques points d'attention à notre code, nous avons une api bien plus rapide et qui consomme beaucoup moins.

La dernière partie de cet article est plus personnel, elle vise à convaincre des entreprises ou des CTO d'opter pour l'éco conception.

Comment convaincre ?

Ici, je vous propose de revoir les points qui vous permettront de convaincre votre entreprise d'adopter l'écoconception, en espérant déjà vous avoir un peu convaincus. Cette fois, on va aborder les choses sous un angle plus concret et surtout plus financier.

Tout d'abord, parlons du plus évident : la réduction des coûts d'infrastructure. Un code plus optimisé, moins vorace en RAM et en CPU, permettrait de faire de grosses économies, surtout pour celles et ceux qui utilisent des solutions de cloud public. Moins de ressources consommées, c'est tout simplement moins de serveurs, donc moins de factures. Et ces économies peuvent être significatives à l'échelle d'un SI complet.

Mais ce n'est pas tout. Un code éco-conçu peut aussi vous faire gagner de l'argent. Pourquoi ? Parce qu'il augmente l'attractivité de votre entreprise. Que ce soit dans les appels d'offres — qui sont de plus en plus nombreux à inclure des critères liés au Green IT — ou vis-à-vis des collaborateurs. Les consultants, les devs, les techs en général sont de plus en plus sensibles à l'impact environnemental de l'entreprise dans laquelle ils travaillent. Intégrer l'écoconception dans vos pratiques, c'est donc aussi un moyen de séduire les talents et de valoriser votre image.

Enfin, on ne peut pas parler d'écoconception sans évoquer l'anticipation des réglementations. Des lois comme la REEN en France sont déjà en place, et ce n'est que le début. De nouvelles obligations vont arriver, que ce soit au niveau national ou européen. Prendre les devants maintenant, c'est éviter de devoir courir demain pour se mettre en conformité, tout en se positionnant comme un acteur responsable et prévoyant.

Conclusion

L'écoconception (et au-delà la sobriété du code), c'est juste du bon sens. Coder proprement, penser à l'impact de ce qu'on développe, faire mieux avec moins —, c'est dans notre intérêt, pour aujourd'hui comme pour demain. Pas besoin de tout changer du jour au lendemain, mais si chacun commence à faire un peu mieux, le résultat à l'échelle est énorme. Alors si on peut coder plus sobre, plus propre, et que ça profite à tout le monde, pourquoi s'en priver ?

Développeurs, agissons concrètement pour la planète avec l'outil Creedengo

En tant que professionnel du numérique, lorsque l'on souhaite agir de manière concrète et positive sur l'impact environnemental de nos applications, on se heurte souvent à une réalité : la majorité des outils disponibles (dans 95 % des cas) sont payants et propriétaires, les rendant « obscurs » dans leur méthodologie et divers calculs proposés.

David DE CARVALHO

Actuellement Architecte Logiciel et Solution chez Capgemini, je cumule plus de 20 ans d'expérience dans les métiers techniques, d'abord en tant que développeur, puis leader technique, avant d'évoluer vers l'architecture logicielle. Le partage de connaissances et l'amélioration continue sont, selon moi, des piliers essentiels de notre profession et sont très importants pour moi. Depuis 2022, je m'investis activement dans des initiatives open-source, avec la volonté d'agir concrètement pour la planète en tant que professionnel du numérique.

« Creedengo », anciennement connu sous le nom de « ecoCode », se distingue en tant qu'outil open source, gratuit et libre d'utilisation dans les entreprises. Porté par une communauté engagée et en pleine croissance, il vise à contribuer activement à la réduction de l'impact environnemental du numérique.

L'objectif de cet article est de montrer concrètement comment chacun peut s'impliquer et contribuer à cette démarche, en s'appuyant sur des outils open source comme Creedengo.

Contexte développeur

En tant que développeur, notre rôle est de concevoir des applications qui répondent efficacement aux besoins des utilisateurs, tout en garantissant des performances optimales. Cela implique également de veiller à produire une implémentation aussi « propre » que possible, afin de favoriser la maintenabilité et l'évolutivité du code applicatif sur le long terme. On attend des développeurs qu'ils produisent du code « propre » — communément appelé « clean code » — afin de garantir une maintenabilité et une évolutivité optimales des applications. Cependant, cet objectif est loin d'être trivial, que l'on parte d'une application à construire complètement ou d'un système existant (« legacy ») à faire évoluer.

Pour répondre à cet enjeu, de nombreuses entreprises s'appuient sur des outils d'analyse statique de code tels que « PMD », « CheckStyle » ou « FindBugs », qui permettent de contrôler la qualité du code selon des règles définies. Ces dernières années, un outil s'est imposé comme une référence dans ce domaine, couvrant de nombreux langages de programmation : SonarQube, développé par l'entreprise SonarSource.

Issu du monde open source, SonarQube propose une gamme complète de fonctionnalités permettant d'évaluer la qualité du code à différentes étapes du cycle de développement d'une application :

- Lors de la phase de développement, SonarQube met à disposition un plugin appelé SonarLint, compatible avec plusieurs environnements de développement (IDE). Ce plugin permet aux développeurs de détecter et corriger les problèmes de code en amont, avant même que celui-ci ne soit sauvegardé (« commité ») dans le dépôt partagé.
  SonarLint se connecte au serveur SonarQube pour récupérer les règles de qualité sur le poste du développeur, puis les applique localement lors de l'analyse du code. Le développeur peut ainsi lancer une analyse sur son poste et corriger les problèmes remontés.
- Lors de la phase de build dans une chaîne d'intégration continue, l'analyse SonarQube est généralement intégrée très tôt dans le processus, et devient très souvent une étape obligatoire et bloquante de ce processus. Une fois le code compilé et les tests unitaires exécutés, une analyse est lancée automatiquement. Celle-ci est évaluée dans SonarQube au travers d'un « Quality Gate », préalablement configuré selon des critères précis (taux de couverture des tests, nombre de bugs critiques, etc.) définis par l'entreprise. Le résultat de cette évaluation détermine si le code passe le contrôle qualité (« Quality Gate vert ») ou s'il nécessite des corrections (« Quality Gate rouge »).

SonarQube s'est imposé comme un outil incontournable dans le monde du développement, largement adopté et parfaitement intégré aux processus de contrôle qualité. Il figure aujourd'hui parmi les leaders du secteur en matière d'analyse et de supervision de la qualité du code.

En tant que professionnel du numérique, il nous revient de tirer pleinement parti de ce type d'outil pour améliorer la qualité de nos applications. Cela permet non seulement de réduire la « dette technique » applicative, mais aussi notre « dette environnementale », en produisant un code plus sobre, plus durable, finalement plus frugal, et donc plus respectueux de la planète.

Contexte Environnemental

L'ADEME (Agence de l'Environnement et de la Maîtrise de l'Énergie) est un organisme public entièrement financé par l'État. Elle joue un rôle clé dans la conduite de nombreux chantiers et études liés à l'environnement.

Parmi ses travaux, plusieurs études mettent en lumière l'impact croissant du numérique sur l'environnement. Dans une publication de janvier 2022, fondée sur des données de 2020, l'ADEME estime que le secteur numérique représente 2,5 % des émissions carbone en France. Bien que précieuse, cette étude présente certaines limites, qu'il convient de garder à l'esprit pour interpréter les résultats avec justesse :

- L'absence de prise en compte des data centers situés à l'étranger ;
- Un manque de précision concernant l'impact environnemental de certaines composantes du numérique, comme les réseaux de télécommunication ou les usages audiovisuels.

Cette étude révélait néanmoins un point clé : 80 % des émis-

sions de carbone du numérique provenaient de la fabrication des terminaux, contre seulement 20 % pour leur usage.
En janvier 2025, une mise à jour de cette étude, basée sur les données de 2022, est venue affiner ces résultats. Elle corrige les limites de la version précédente et met en évidence une réalité encore plus préoccupante : le numérique représente désormais 4,4 % des émissions de carbone en France, soit un niveau équivalent à celui du transport routier de marchandises.
Enfin, la répartition entre fabrication et usage a également été revue : 60 % des émissions sont liées à la fabrication, mais l'usage représente désormais 40 %, soit une part bien plus importante que ce que l'on estimait auparavant.
C'est précisément sur cette part liée à l'usage que nous, professionnels du numérique, avons un levier d'action direct — et une responsabilité.
À mesure que le numérique s'impose dans tous les aspects de nos vies, il devient impératif de concevoir des applications plus sobres, moins gourmandes en ressources et en données. L'open-source représente une voie concrète pour contribuer à cette transition.

Référence étude 2022 : https://www.arcep.fr/uploads/tx_gspublication/etude-numerique-environnement-ademe-arcep-volet01_janv2022.pdf
Visuel résumé de l'étude 2022 : https://infos.ademe.fr/magazine-avril-2022/faits-et-chiffres/numerique-quel-impact-environnemental/

Référence étude 2025 :
https://ecoresponsable.numerique.gouv.fr/docs/2024/etude-ademe-impacts-environnementaux-numerique.pdf
Visuel résumé de l'étude 2025 : https://infos.ademe.fr/magazine-janvier-2025/numerique-quel-impact-environnemental-en-2022/

Le monde de l'Open Source

Nous intégrons souvent des dizaines, voire des centaines de briques techniques (frameworks) open-source dans nos applications. Il semble donc naturel, voire essentiel, de contribuer en retour : soit directement aux composants que nous utilisons, soit à des projets qui résonnent avec nos valeurs, comme ceux liés à la réduction de l'impact environnemental du numérique.
Contribuer à l'open source, c'est aussi investir dans la pérennité des frameworks que nous utilisons au quotidien. Cela permet de garantir leur évolution, leur maintenance, et leur adaptation aux nouveaux enjeux.
Mais au-delà de l'aspect technique, contribuer à l'open-source, c'est aussi faire partie d'une communauté permettant à chacun de s'améliorer dans plusieurs domaines, notamment le travail d'équipe à distance et la communication.
La participation à une communauté possède d'autres avantages non négligeables :

- Monter en compétences techniques ;
- Valoriser son expertise ;
- Participer à des événements techniques (hackathons, conférences, etc.) ;

Prenons l'exemple du collectif « Green Code Initiative », qui œuvre depuis 2019 pour un numérique plus responsable. Devenu une association loi 1901 en novembre 2024, ce collectif met à disposition plusieurs outils open-source dédiés à la mesure et à la réduction de l'impact environnemental du code :

- Creedengo (anciennement « ecoCode ») : le premier outil du collectif, lancé en 2019, permet de détecter les mauvaises pratiques de développement ayant un impact environnemental négatif. Il a été renommé en décembre 2024.
- EcoSonar : un outil de mesure d'impact environnemental, reversé en open-source par la société Accenture fin 2024.
- GreenSight : une solution de visualisation (dashboarding) de l'impact environnemental, reversé par la société Capgemini à mi-2024.

L'open-source devient alors un levier puissant pour mutualiser les efforts, partager les bonnes pratiques et construire ensemble des solutions plus durables.

CREEDENGO : utilité et objectifs

Creedengo est conçu pour détecter les mauvaises pratiques de développement ayant un impact environnemental négatif, et proposer une alternative plus vertueuse pour chaque problème détecté.
Cet outil repose sur SonarQube, une solution d'analyse statique de code largement utilisée dans l'industrie. SonarQube permet d'identifier les mauvaises pratiques dans le code source et de suggérer des améliorations, en s'appuyant sur un ensemble de règles configurables.
Creedengo s'intègre à SonarQube sous forme de plugins, disponibles notamment via le « Marketplace » officiel. Une fois installés et configurés, ces plugins rajoutent des règles spécifiques de « green coding », permettant d'évaluer le code sous l'angle de l'impact environnemental. Chaque plugin Creedengo est dédié à un langage ou une plateforme spécifique. À ce jour, les plugins disponibles couvrent :

- Le monde mobile : Android (Java), iOS (Swift)
- Le monde non mobile : Java, PHP, Python, JavaScript/TypeScript, C#
- En cours de développement : Rust, HTML/CSS, Angular, Flutter, ainsi que des technologies d'infrastructure comme Docker, Kubernetes et Ansible

CREEDENGO : utilisation des plugins SonarQube

L'intégration des plugins Creedengo dans SonarQube se fait principalement via le « Marketplace » officiel, accessible une fois connecté avec des droits administrateurs. Il est possible d'y installer autant de plugins que nécessaire, selon les langages ou plateformes ciblés.
La majorité des plugins Creedengo — notamment pour Java, PHP, Python, JavaScript/TypeScript, C#, Android Java — sont déjà disponibles directement depuis ce Marketplace.
D'autres, comme le plugin iOS Swift, sont encore en cours de validation par les équipes de SonarQube. En attendant leur publication officielle, ils doivent être installés manuellement, en copiant le fichier JAR versionné du plugin dans le répertoire des extensions de SonarQube.
Une fois le plugin installé, l'ensemble des règles de « green coding » qu'il contient devient accessible dans l'interface de SonarQube. Pour qu'elles soient prises en compte lors des analyses, il est nécessaire de les activer dans un profil SonarQube (existant ou nouvellement créé).

Les règles proposées par les plugins Creedengo s'appuient sur des référentiels reconnus, tels que :

- « Les 115 bonnes pratiques » de GreenIT.fr : https://github .com/cnumr/best-practices
- « GR491 » – Guide de Référence de l'écoconception de services numériques : https://gr491.isit-europe.org/
- « RGESN » – Référentiel Général d'Écoconception des Services Numériques : https://ecoresponsable.numerique.gouv.fr/publications/referentiel-general-ecoconception/
- « Mobile Ecodesign Best Practices » : https://github.com/cnumr /best-practices-mobile

La **figure 1** ci-dessous propose un exemple de mauvaise pratique identifiée.

CREEDENGO : contribuer en créant une règle

Cas d'utilisation sur le plugin Creedengo-java

Prenons un exemple concret : nous souhaitons détecter, dans du code Java, toutes les requêtes SQL de la forme « SELECT * FROM » comme montré dans la figure 1 ci-dessus.

Cette pratique est considérée comme problématique, car elle récupère l'ensemble des colonnes d'une table, sans distinction. Or, dans la majorité des cas, seules quelques-unes de ces colonnes sont réellement utilisées dans le code. Cela entraîne un transfert de données inutile entre la base de données et l'application, voire vers d'autres systèmes en aval.

De plus, si une nouvelle colonne est ajoutée à la table, elle sera automatiquement incluse dans les résultats de la requête, même si elle n'est pas pertinente pour le traitement en cours. Cela peut générer une surcharge de données, une consommation réseau accrue, et donc un impact environnemental plus important.

La bonne pratique consiste donc à ne pas utiliser les requêtes de type « SELECT * FROM », et à privilégier des requêtes listant explicitement les colonnes réellement utilisées dans le code de l'application.

Figure 1 : *exemple mauvaise pratique*

Rapide explication des repository Github de Creedengo

Sur GitHub, sous l'organisation « Green Code Initiative », plusieurs repository sont disponibles pour contribuer au projet Creedengo. Voici les principaux :

- « creedengo-common » : contient la documentation générale et quelques petits outils pratiques pour les développeurs.
  —> Le fichier « starter-pack.md » est le point d'entrée recommandé : il explique la méthodologie à suivre et comment bien démarrer une contribution.
- « creedengo-<LANGAGE> » : chaque repository correspond à un plugin Creedengo pour un langage donné (par exemple : « creedengo-java », « creedengo-python », etc.).
- « creedengo-rules-specifications » : regroupe le référentiel technique de l'ensemble des règles utilisées par tous les plugins (cf listing exhaustif dans le fichier « RULES.md »).

ÉTAPE 1 : Création de la classe implémentant la règle

Étape 1.1 : Initialisation de la classe

Nous créons une classe contenant l'implémentation de la règle "AvoidFullSQLRequest" au sein du package « org.greencodeinitiative.creedengo.java.checks », qui regroupe l'ensemble des classes d'implémentation des règles Creedengo pour le langage Java.

Cette classe doit hériter de la classe « IssuableSubscriptionVisitor » fournie par l'API de SonarQube, qui permet ainsi l'analyse syntaxique (AST - Abstract Syntax Tree) des fichiers source Java analysés.

Elle doit également être annotée avec « @Rule », annotation qui permet d'associer cette implémentation à l'identifiant unique de la règle qu'elle implémente.

```
package org.greencodeinitiative.creedengo.java.checks;

import org.sonar.check.Rule;
import org.sonar.plugins.java.api.IssuableSubscriptionVisitor;

// Annotation de déclaration du numéro de la règle
@Rule(key = "GCI74")
// Héritage de la classe IssuableSubscriptionVisitor de l'API SonarQube
// permettant de déclarer la classe actuelle comme pouvant analyser des noeuds de l'AST
public class AvoidFullSQLRequest extends IssuableSubscriptionVisitor {
```

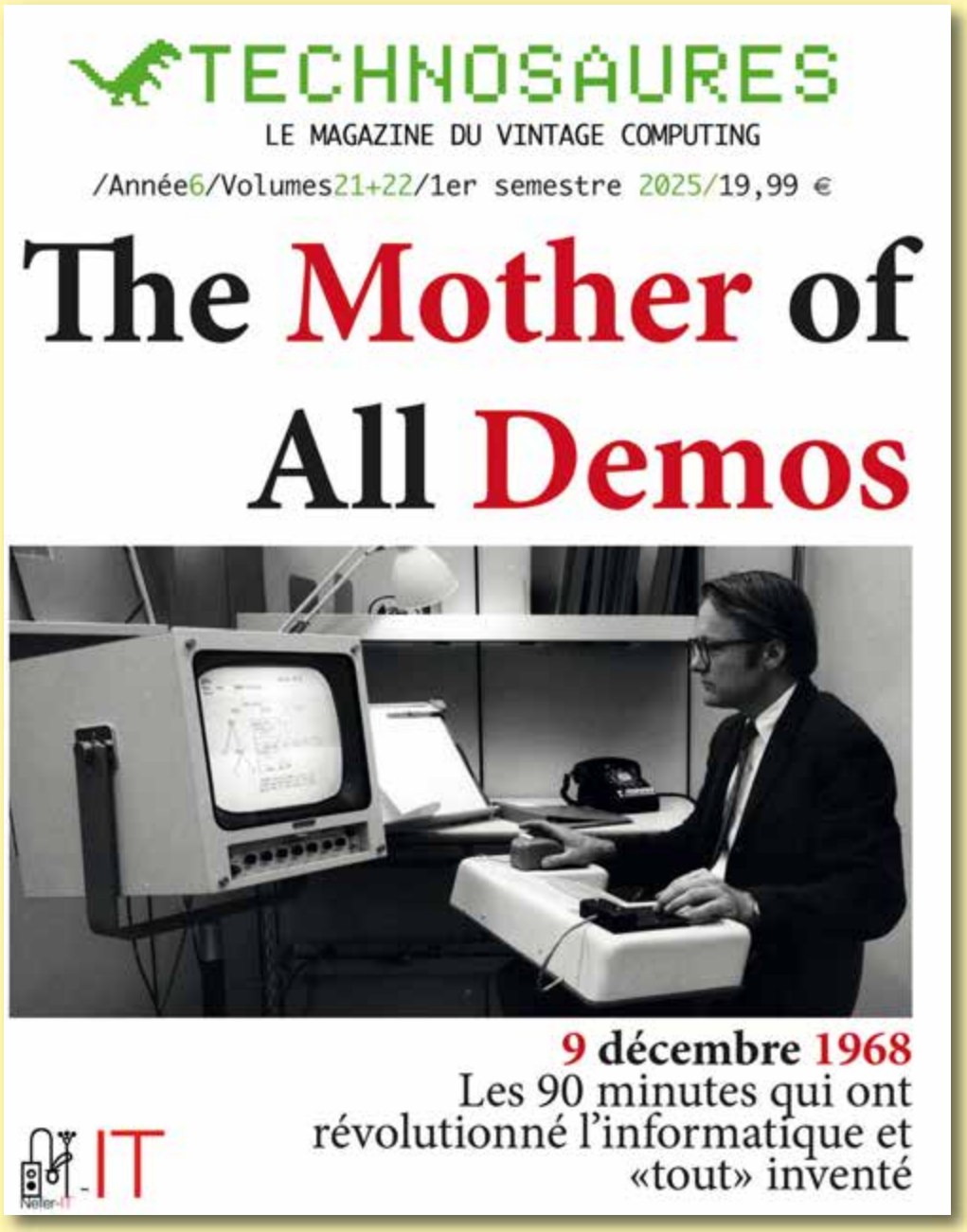
Étape 1.2 : surcharge de la méthode de déclenchement de l'analyse

Nous implémentons la méthode « nodesToVisit », issue de la classe « IssuableSubscriptionVisitor » de l'API SonarQube. Cette méthode permet de spécifier les types de nœuds de l'arbre syntaxique (AST) que l'on souhaite analyser lors du parcours du code source.

Dans notre cas, l'objectif est d'inspecter toutes les chaînes de caractères présentes dans le code Java, afin d'y détecter les occurrences de requêtes SQL contenant le motif « SELECT * FROM ».

```
// Détermination des types de noeud dans l'AST à visiter
  @Override
  public List<Kind> nodesToVisit() {
    return singletonList(Tree.Kind.STRING_LITERAL);
  }
```

Le nouveau numéro de Technosaures est disponible !



Au sommaire :

- The Mother of All Demos
- Il y a 50 ans, l'Altair 8800
- GeOS tente de concurrencer Windows
- Falcon 030 ne sauve pas Atari

Disponible sur programmez.com et amazon.fr

Étape 1.3 : ajout de la méthode d'analyse

Nous définissons une variable contenant une expression régulière destinée à identifier, dans une chaîne de caractères, les requêtes SQL de la forme « SELECT * FROM » indépendamment de la casse.

```
// Expression régulière pour détecter la requête SELECT * FROM
  private static final Predicate<String> SELECT_FROM_REGEXP =
      compile("select\\s*\\*\\s*from", CASE_INSENSITIVE).asPredicate();
```

Nous implémentons ensuite la méthode d'analyse principale, héritée de la classe « IssuableSubscriptionVisitor » de l'API SonarQube. Le paramètre d'entrée « tree » correspond à la chaine de caractère détectée. Celle-ci est analysée à l'aide de l'expression régulière définie précédemment, afin de détecter la présence du motif SQL ciblé.

Si une correspondance est trouvée, une alerte est générée via l'appel à la méthode « reportIssue ». Cette alerte sera ensuite affichée dans l'interface de SonarQube, précisément à la ligne où la chaîne problématique a été détectée dans le code (comme montré dans la figure 1 ci-dessus).

```
// Traitement effectué sur chaque noeud de l'AST visité
  @Override
  public void visitNode(Tree tree) {
    String value = ((LiteralTree) tree).value();
    if (SELECT_FROM_REGEXP.test(value)) {
      reportIssue(tree, "Don't use the query SELECT * FROM");
    }
  }
```

Voici la classe complète d'implémentation de la règle :

```
package org.greencodeinitiative.creedengo.java.checks;

import static java.util.Collections.singletonList;
import static java.util.regex.Pattern.CASE_INSENSITIVE;
import static java.util.regex.Pattern.compile;
import java.util.List;
import java.util.function.Predicate;
import org.sonar.check.Rule;
import org.sonar.plugins.java.api.IssuableSubscriptionVisitor;
import org.sonar.plugins.java.api.tree.LiteralTree;
import org.sonar.plugins.java.api.tree.Tree;
import org.sonar.plugins.java.api.tree.Tree.Kind;

// Annotation de déclaration du numéro de la règle
@Rule(key = "GCI74")
// Héritage de la classe IssuableSubscriptionVisitor de l'API SonarQube
// permettant de déclarer la classe actuelle comme pouvant analyser des noeuds
de l'AST
public class AvoidFullSQLRequest extends IssuableSubscriptionVisitor {

  // Expression régulière pour détecter la requête SELECT * FROM
  private static final Predicate<String> SELECT_FROM_REGEXP =
      compile("select\\s*\\*\\s*from", CASE_INSENSITIVE).asPredicate();

  // Détermination des types de noeud dans l'AST à visiter
  @Override
  public List<Kind> nodesToVisit() {
    return singletonList(Tree.Kind.STRING_LITERAL);
  }

  // Traitement effectué sur chaque noeud de l'AST visité
  @Override
  public void visitNode(Tree tree) {
    String value = ((LiteralTree) tree).value();
    if (SELECT_FROM_REGEXP.test(value)) {
      reportIssue(tree, "Don't use the query SELECT * FROM");
    }
  }
}
```

Étape 1.4 : déclaration de la nouvelle classe afin d'être reconnue par l'API Sonarqube

Il est désormais nécessaire d'enregistrer cette nouvelle classe d'implémentation dans la variable statique « ANNOTATED_RULE_CLASSES » de la classe « JavaCheckRegistrar » (au cœur du système d'enregistrement des classes d'implémentation des règles), afin qu'elle soit reconnue par le système comme une règle active et prise en compte lors des futures analyses.

```
package org.greencodeinitiative.creedengo.java;

import java.util.Collections;
import java.util.List;

import org.greencodeinitiative.creedengo.java.checks.*;
import org.sonar.plugins.java.api.CheckRegistrar;
import org.sonar.plugins.java.api.JavaCheck;
import org.sonarsource.api.sonarlint.SonarLintSide;

@SonarLintSide
public class JavaCheckRegistrar implements CheckRegistrar {
  static final List<Class<? extends JavaCheck>> ANNOTATED_RULE_
CLASSES = List.of(
      ArrayCopyCheck.class,
      IncrementCheck.class,
      AvoidUsageOfStaticCollections.class,
      AvoidGettingSizeCollectionInLoop.class,
      AvoidFullSQLRequest.class // Ajout de la nouvelle classe d'implémentation
  );

  /**
   * Register the classes that will be used to instantiate checks during analysis.
   */
  @Override
  public void register(RegistrarContext registrarContext) {
    // Call to registerClassesForRepository to associate the classes with the
correct repository key
    registrarContext.registerClassesForRepository(
      JavaRulesDefinition.REPOSITORY_KEY,
      checkClasses(),
      testCheckClasses()
    );
  }

  /**
   * Lists all the main checks provided by the plugin
```

```
   */
  public static List<Class<? extends JavaCheck>> checkClasses() {
    return ANNOTATED_RULE_CLASSES;
  }

  /**
   * Lists all the test checks provided by the plugin
   */
  public static List<Class<? extends JavaCheck>> testCheckClasses() {
    return Collections.emptyList();
  }
}
```

ÉTAPE 2 : Ajout d'un test unitaire

Étape 2.1 : Création de la classe du test unitaire

La classe de test unitaire « AvoidFullSQLRequestCheckTest » permet de valider le bon fonctionnement de la classe d'implémentation de la règle « AvoidFullSQLRequest ».

Elle s'appuie sur le framework de test de l'API SonarQube, notamment la classe « CheckVerifier ».

La méthode de test repose sur les éléments suivants (cité dans l'ordre du code source ci-dessous) :

- Le fichier ressource contenant le code Java à analyser ;
- La nouvelle classe d'implémentation de la règle, utilisée pour effectuer l'analyse ;
- la méthode de lancement de la vérification (« verifyIssues() », qui s'assure que les problèmes attendus sont bien détectés et signalés à la bonne ligne du code source analysé.

```
package org.greencodeinitiative.creedengo.java.checks;

import org.junit.jupiter.api.Test;
import org.sonar.java.checks.verifier.CheckVerifier;

class AvoidFullSQLRequestCheckTest {

  @Test
  void test() {
    CheckVerifier.newVerifier()
        .onFile("src/test/files/AvoidFullSQLRequestCheck.java")
        .withCheck(new AvoidFullSQLRequest())
        .verifyIssues();
  }

}
```

Étape 2.2 : Création du fichier ressource à analyser

Voici le fichier Java qui sera analysé dans le test unitaire (« src/test/files/AvoidFullSQLRequestCheck.java »).

```
package org.greencodeinitiative.creedengo.java.checks;

import java.util.regex.Pattern;

class AvoidFullSQLRequestCheck {
  AvoidFullSQLRequestCheck(AvoidFullSQLRequestCheck mc) {
  }

  public void literalSQLrequest() {
    dummyCall("  sElEcT * fRoM myTable"); // Noncompliant {{Don't use the
query SELECT * FROM}}
    dummyCall("   sElEcT user fRoM myTable");

    dummyCall("SELECTABLE 2*2 FROMAGE"); //not sql
    dummyCall("SELECT    *FROM table"); // Noncompliant {{Don't use the
query SELECT * FROM}}
  }

  public void variableSQLrequest() {
    String requestNonCompiliant = "  SeLeCt * FrOm myTable"; // Noncompliant
{{Don't use the query SELECT * FROM}}
    String requestCompiliant = "   SeLeCt user FrOm myTable";
    dummyCall(requestNonCompiliant);
    dummyCall(requestCompiliant);

    String noSqlCompiliant = "SELECTABLE 2*2 FROMAGE"; //not sql
    String requestNonCompiliant_nSpace = "SELECT   *FROM table"; // Noncompliant
{{Don't use the query SELECT * FROM}}
  }

  private void dummyCall(String request) {

  }

}
```

Les commentaires en fin de ligne sont obligatoires et jouent un rôle essentiel.

En effet, le framework de test de SonarQube s'appuie sur ces commentaires (notamment le « // Noncompliant ») pour identifier les lignes du code source censées déclencher une alerte. Sans ces commentaires, les tests ne peuvent valider correctement la détection des problèmes.

ÉTAPE 3 : ajout d'un test end-to-end (ou test d'intégration)

Étape 3.1 : création du fichier Java exemple à analyser

Le fichier Java d'exemple à analyser est le même que celui pour le test unitaire et il est également copié dans le répertoire des ressources utilisées pour les tests end-to-end. Les commentaires en fin de ligne ne sont pas pris en compte dans ce contexte de tests end-to-end, car la méthode de vérification utilisée diffère totalement de celle des tests unitaires. Ces commentaires restent tout de même utiles car ils permettent tout de même à un utilisateur de détecter visuellement les lignes sur lesquelles il devrait y avoir un problème remonté par SonarQube.

Étape 3.2 : Ajout d'une méthode de test end-to-end

Pour valider la nouvelle implémentation dans un environnement réel (end-to-end), il est nécessaire d'ajouter une méthode de test dans la classe de test « GCIRuleIT », qui centralise l'ensemble des tests end-to-end.

```
package org.greencodeinitiative.creedengo.java.integration.tests;

import org.junit.jupiter.api.Test;

class GCIRulesIT extends GCIRulesBase {
```

```
  @Test
  void testGCI74() {

    String filePath = "src/main/java/org/greencodeinitiative/creedengo
/java/checks/AvoidFullSQLRequestCheck.java";
    int[] startLines = new int[]{8, 12, 17, 23};
    int[] endLines = new int[]{8, 12, 17, 23};
    String ruleId = "creedengo-java:GCI74";
    String ruleMsg = "Don't use the query SELECT * FROM";

    checkIssuesForFile(filePath, ruleId, ruleMsg, startLines, endLines, SEVERITY,
TYPE, EFFORT_20MIN);

  }

}
```

Cette nouvelle méthode de test fait appel à une méthode commune nommée « checkIssuesForFile », à laquelle sont transmis les éléments suivants :

- le fichier Java à analyser ;
- les lignes sur lesquelles il devrait y avoir une erreur (ligne de début et ligne de fin) ;
- l'identifiant de la règle censée déclencher les erreurs ;
- le message d'erreur attendu.

Cette méthode partagée, utilisée par l'ensemble des tests end-to-end, a pour objectif de vérifier que les problèmes détectés lors de l'analyse correspondent bien aux éléments fournis en entrée. Contrairement aux tests unitaires, les tests end-to-end simulent un environnement d'exécution complet. Ils impliquent plusieurs étapes supplémentaires automatiquement lancées, notamment :

- la compilation et la préparation du plugin Java contenant la nouvelle règle ;
- le démarrage d'une instance locale de SonarQube sur un port disponible ;
- l'installation du plugin Java avec la nouvelle règle implémentée ;
- l'installation du profil SonarQube nécessaire à l'activation de cette règle ;
- l'exécution de l'analyse via le scanner Sonar ;
- l'envoi des résultats d'analyse à l'instance locale de SonarQube ;
- la vérification que les erreurs attendues sont bien remontées aux lignes spécifiées ;
- l'arrêt de l'instance locale de SonarQube

Le test est considéré comme réussi si les erreurs sont bien détectées aux lignes indiquées, avec l'identifiant de règle et le message attendus.

Hackathon de mai 2025 à Paris : plus de 120 participants

Manifestation / Conférences techniques

« Green Code Challenge »

Chaque année, un hackathon gratuit rassemble plus d'une centaine de développeurs pour dynamiser la contribution open-source autour des plugins « Creedengo ». Cet événement de deux jours est une formidable opportunité pour s'impliquer sur un sujet porteur de sens : réduire l'impact environnemental des applications que nous concevons.
Le hackathon : https://green-code-initiative.org/challenge
Devenez contributeur de l'outil et rejoignez notre communauté ! https://green-code-initiative.org/contributeur#joinus

Conférences techniques

BreizhCamp 2023 : Conférence « ecoCode : comment réduire l'empreinte carbone de vos projets en utilisant des outils open-source » https://www.youtube.com/watch?v=VVTxus-70ew
Capitole du Libre 2024 : Conférence « les apps écologiques, c'est pas automatique ? » https://www.youtube.com/watch?v=77R-TSWveOA
Devoxx 2025 : Conférence « le futur sera vert, notre code aussi » https://www.youtube.com/watch?v=0KWhs5DN6oo
BreizhCamp 2025 : Conférence « Développeurs, votre code façonne le monde de demain ! » (sur Youtube courant de l'été 2025)

	Référence / URL
ADEME : Etude janvier 2022	https://www.arcep.fr/uploads/tx_gspublication/etude-numerique-environnement-ademe-arcep-volet01_janv2022.pdf
ADEME : visuel résumé de l'étude 2022	https://infos.ademe.fr/magazine-avril-2022/faits-et-chiffres/numerique-quel-impact-environnemental/
ADEME : Etude janvier 2025	https://ecoresponsable.numerique.gouv.fr/docs/2024/etude-ademe-impacts-environnementaux-numerique.pdf
ADEME : visuel résumé de l'étude 2025	https://infos.ademe.fr/magazine-janvier-2025/numerique-quel-impact-environnemental-en-2022/
EcoCode : réduisez la dette environnementale de vos applications (article également paru sur Programmez en 2023)	https://glalloue.medium.com/ecocode-r%C3%A9duisez-la-dette-environnementale-de-vos-applications-991b199e0bc9
Référentiel « 115 bonnes pratiques » de l'association « GreenIT.fr »	https://github.com/cnumr/best-practices
Référentiel « GR491 »	https://gr491.isit-europe.org/
Référentiel « RGESN »	https://ecoresponsable.numerique.gouv.fr/publications/referentiel-general-ecoconception/
Référentiel « Mobile Ecodesign Best Practices »	https://github.com/cnumr/best-practices-mobile
Github de l'organisation Green-Code-Initiative et du plugin java	https://github.com/green-code-initiative https://github.com/green-code-initiative/creedengo-java
Site web de l'organisation Green-Code-Initiative	https://green-code-initiative.org/

Les partenaires 2025 de

programmez!

Le magazine des dév - CTO & Tech Lead

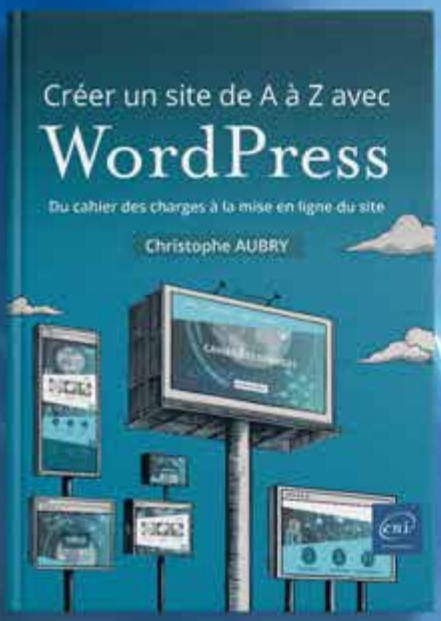
Vous voulez soutenir activement Programmez! ?
Devenir partenaires de nos dossiers en ligne et de nos événements ?

Contactez-nous dès maintenant :

ftonic@programmez.com

APPRENEZ
PROGRESSEZ
MAÎTRISEZ

avec nos nouveaux livres



ENI Editions

FORMATIONS À L'INFORMATIQUE EN LIGNE

www.editions-eni.fr