INGÉNIERIE QUALITÉ & EXPÉRIENCE DÉVELOPPEUR

Rendre simple ce qui est complexe.

Je transforme les besoins produit et techniques en logiciels, plateformes de test et outils que les équipes peuvent utiliser avec confiance.

Mon parcours relie l'analyse métier, le développement logiciel et l'ingénierie qualité. J'aborde un besoin à travers les usages, les processus de l'organisation, les contraintes techniques et les preuves nécessaires pour valider le résultat. Cette lecture d'ensemble m'aide à cadrer le problème, construire une réponse et donner aux équipes les moyens de la faire évoluer.

DES BESOINS AUX SYSTÈMES

- **Relier qualité et enjeux métier**
  Fnac, Manpower, AFNOR, EDF (division nucléaire) / Avant 2014
- **Comprendre le contexte**
  OPCAIM / 2014-2016
- **Construire la réponse**
  BearingPoint et BNP Paribas Cardif / 2017-2021
- **Rendre vérifiable**
  Agorapulse / 2021-2022
- **Éclairer la livraison**
  Deezer / 2022
- **Valider sous contraintes**
  HarfangLab / 2023
- **Outiller les équipes**
  Believe / 2023-2025
- **Formaliser les connaissances**
  R&D indépendante / 2026 - En cours

DANS CE DOSSIER

QaaS 2
Du scénario métier à l'exécution

Produits & DevEx 3
beOne, interfaces et assistant local

Qualité & performance 4
Des signaux utiles pour décider

QA Hub 5
Définir les garanties du système

Agent Assurance 6
Résultats, parcours et preuves

POUR ÉCHANGER

lydia.arad@proton.me

Réalisations chez Believe (2023-2025) et recherche indépendante. Chaque étude complète est accessible depuis les liens du dossier.

01 / PLATEFORME QUALITÉ

QaaS
Le scénario au premier plan.

Conception et développement du socle de test partagé et d'intégrations de QaaS, en collaboration avec les ingénieurs des équipes contributrices. L'objectif : rendre l'écriture, l'exécution et la maintenance des tests plus accessibles aux équipes.

Scénario métier
Fixtures, commandes et factories

→

Accès partagés
Clients API, configuration et identité

→

Exécution
Lanceurs et environnements

→

Restitution
Consoles, rapports et métriques

API / pytest

Les fixtures préparent clients et données ; les assertions communes expriment les vérifications. Interfaces publiques, métadonnées, décorateurs, registres et plugins organisent la déclaration et la sélection des tests.

Web / Cypress

Les commandes métier regroupent la session et les parcours récurrents. Sélecteurs, attentes et correspondances de statuts se maintiennent dans le socle partagé.

Performance / k6

Les factories composent opérations, profils de charge, contrôles et métriques. Les couches HTTP et d'instrumentation harmonisent les appels et la mesure.

2 sem. → 2 h
Prise en main de QaaS

80 → 21
Lignes d'un scénario pytest

13 → 2
Appels Cypress d'un exemple métier

Ce que montrent les chiffres : le gain de prise en main concerne les QA et les développeurs qui démarrent avec QaaS. Les réductions de code illustrent des scénarios individuels ; la préparation et les contrôles récurrents sont mutualisés dans le socle.

Préparer les accès une fois

Les contrats OpenAPI référencés dans Backstage, le catalogue interne de services, alimentent la génération de clients Python. Une couche commune prépare leur configuration et leur authentification ; les intégrations Vault et OIDC préparent les accès partagés.

Des environnements prêts à l'emploi

Docker et Compose fournissent des runtimes versionnés, des profils de développement et de débogage, des caches et des résultats persistants. Les lanceurs préparent les exécutions ; l'outillage k6 génère des Jobs Kubernetes et suit statuts, logs et nettoyage. Les adaptations couvrent macOS, Linux et Windows via WSL2.

Les auteurs assemblent les capacités métier ; les mécanismes récurrents restent maintenables dans un même socle. Une requête commune aux moteurs et un catalogue de scénarios constituent la prochaine direction de la plateforme.

Architecture et exemples Gains illustrés

02 / PRODUITS ET EXPÉRIENCE DÉVELOPPEUR

Construire le produit. Outiller les équipes.

Comprendre le travail des développeurs, des QA et des Product Owners, puis réunir préparation, expression du scénario, lancement et lecture du résultat dans un parcours utilisable.

beOne / de l'idée au MVP

Cadrage produit, UX/UI et développement React/TypeScript d'un portail interne d'onboarding. Contrôles de contribution, versionnement et adaptation des pipelines GitLab accompagnent les prévisualisations et la livraison en intégration via Argo CD sur GCP. Le MVP déployé permet à l'équipe d'utiliser et d'évaluer le produit.

Interface Mapper / automatiser la préparation

Un serveur MCP et Playwright inspectent la page, extraient les éléments et des sélecteurs candidats, puis vérifient leur unicité. La carte sert à préparer des commandes Cypress. Un add-on navigateur distinct analyse le DOM pour générer tests et commandes ; une extension Faker prépare des données de test.

RENDRE LE RÉSULTAT LISIBLE

Sorties pytest structurées avec Loguru, réponses API lisibles et statut/durée par test ; formateur Cypress distinguant les phases d'exécution avec quatre niveaux de détail, captures et vidéos ; API Tester VS Code avec sélection Swagger, formulaires de requête et historique. Les interfaces relient le contexte du test aux diagnostics utiles.

Ève / un accès naturel au savoir de QaaS

J'ai conçu et développé un assistant documentaire RAG local : extraction de documentation, docstrings et commentaires, recherche sémantique et génération avec références aux sources. Il aide les ingénieurs à retrouver une convention, comprendre une commande et suivre l'explication vers le fichier concerné.

Extraire — Passages et chemins sources → **Indexer** — MiniLM et FAISS → **Retrouver** — Contexte lié à la question → **Répondre** — llama.cpp et références

Service Flask, index et modèle intégrés à l'environnement QaaS ; Phi-2 et Mistral sont des options de modèles locaux. L'inférence est exécutée localement après installation.

Évolution : évaluer la qualité des réponses, l'exactitude des références et la fraîcheur de l'index. La préparation et le lancement de campagnes via un orchestrateur commun constituent une piste d'extension.

beOne et Interface Mapper Interfaces et consoles Ève en détail

03 / QUALITÉ ET PERFORMANCE

Relier les résultats aux décisions.

Chez Believe, j'ai relié résultats k6, observations Datadog et risques métier aux recommandations Go/No Go. Les analyses étaient présentées aux leads et VP pour les décisions de déploiement par Business Unit, notamment autour de la migration GCP.

UNE CAMPAGNE DE SYNCHRONISATION ANONYMISÉE

La campagne suivait la soumission, la synchronisation, l'envoi des fichiers, leur transfert, l'encodage et la résorption du travail accumulé. Un envoi réussi était suivi jusqu'à la mise à disposition en aval.

Volume

Objectif de nombre de publications atteint

Délai

Mise à disposition en aval encore trop lente

Action

Traiter les goulots avant d'élargir le déploiement

Observer toute la chaîne

Les délais de transfert et d'encodage continuaient à augmenter pendant la résorption du travail en attente. L'analyse associait délais métier, erreurs, tenue en charge et respect des seuils aux signaux Datadog : expérience navigateur, CPU et comportement des pods.

Transformer les constats en priorités

Les recommandations portaient sur le travail de synchronisation en attente, le débit des services en aval, le transfert et l'encodage. Un retour plus clair sur la progression devait aussi aider les utilisateurs à comprendre l'état de leur publication.

LIRE LES INDICATEURS AU BON NIVEAU

Dans un exemple de restitution k6, le seuil global de temps de réponse est respecté tandis qu'une opération dépasse son seuil plus strict. Les contrôles fonctionnels apportent un signal complémentaire sur les données et le comportement attendus.

Périmètre	Lecture du résultat
Temps de réponse globaux	Seuil global respecté
Opération individuelle	Seuil propre à l'opération dépassé
Vérifications fonctionnelles	Comparer données reçues, attentes et hypothèses du test

La démarche relie le profil de charge, le comportement observé et l'infrastructure. Elle rend explicites les critères de préparation d'une release, les corrections prioritaires et les vérifications suivantes.

Démarche et campagne Exemple de restitution k6

04 / R&D INDÉPENDANTE

QA Hub
Définir les bonnes garanties.

Une plateforme d'aide à la décision qualité pour déterminer ce qu'un système doit garantir, puis choisir les tests prioritaires. Elle relie expertise métier, contraintes du produit et méthodes de vérification.

POC produit fonctionnel / Architecture du moteur et règles de composition spécifiées

Confirmer
Contexte et hypothèses

→

Composer
Packs de qualité versionnés

→

Prioriser
Risques, propriétés et tests

→

Expliquer
Références et décisions

Un modèle adapté au système

Dans l'architecture définie, le Quality Model Engine sélectionne des Quality Model Packs et les compose par des règles déterministes. Les sorties relient risques, propriétés, métriques, stratégies, outils et artefacts. Chaque pack expose ses connaissances et son périmètre.

Des décisions inspectables

La Quality Evidence Layer relie contexte, versions des packs, références, hypothèses, conflits et arbitrages dans un enregistrement de décision. Les experts peuvent modifier les priorités avec traçabilité. L'IA accompagne l'interprétation et la génération ; le cœur de décision conserve ses règles explicites.

EXEMPLE / RECOMMANDATION MUSICALE SUR MOBILE

Un service qui recommande des titres et lance leur lecture combine plusieurs exigences : pertinence des recommandations, continuité d'écoute, interruptions réseau et budget de réponse. Le modèle doit faire apparaître leurs interactions : une méthode pertinente peut demander un coût incompatible avec le budget mobile.

La stratégie proposée relie critères de pertinence, jeux de données ou jugements humains, parcours d'écoute et observations en charge. Les conflits et les dimensions partiellement couvertes restent attachés au raisonnement pour permettre l'arbitrage.

Disponible / expérience produit

Un questionnaire de quatre questions et cinq secteurs contextualisent des stratégies et des artefacts exportables. Le POC emploie des modèles paramétrés et une activité d'agents simulée pour donner à voir le parcours utilisateur.

Prochaines étapes / moteur et preuves

Construire l'Evidence Layer, implémenter le schéma des packs et le moteur déterministe, puis valider décisions et provenance sur des cas de référence. La vision à terme croise application, DOM, API, documentation, Jira, Confluence, tests et traces pour cartographier parcours et preuves.

Produit, architecture et exemple complet

05 / R&D INDÉPENDANTE

Agent Assurance
Résultat, parcours et preuves.

Justifier la confiance dans le travail d'un agent : vérifier son résultat et les limites respectées au fil de l'exécution. L'hypothèse prioritaire étudie les capacités créées par la composition d'opérations autorisées séparément.

Architecture logique et protocole définis / Prochaine étape : expérimentation MCP contrôlée

Work Assurance / le résultat

Évaluer le changement livré au regard des exigences fonctionnelles, des tests de non-régression, de la sécurité, de la performance et des attentes métier. Le type de tâche détermine les preuves et les vérificateurs adaptés.

Path Assurance / le parcours

Relier opérations, identités, délégations, flux de données et ressources partagées. Des opérations localement autorisées peuvent, par leur enchaînement, créer une capacité interdite par le contrat de tâche. MCP constitue la première surface expérimentale.

SCÉNARIO DE RECHERCHE / UNE RESSOURCE ENTRE DEUX EXÉCUTIONS

L'exécution A peut écrire un livrable ; l'exécution B peut lire ce stockage. Les autorisations locales permettent ces opérations, tandis que les règles de la tâche imposent leur isolation. L'expérience doit suivre l'identité et la provenance de la ressource entre les deux exécutions.

A écrit
Livrable et origine identifiés

→

B demande à lire
Ressource reliée à A

→

Évaluer le chemin
Contrat, provenance et règle

Décision attendue : refuser la lecture avant son effet lorsque le canal interdit entre exécutions est établi. Un partage explicitement autorisé par le contrat sert de cas de comparaison.

SÉPARER LE DEGRÉ DE CERTITUDE DE L'ACTION

État de l'évaluation

VERIFIED : observations et intégrité suffisantes dans le périmètre déclaré.
INCONCLUSIVE : éléments disponibles insuffisants pour conclure.
DEGRADED : surface de contrôle ou d'observation affaiblie.

Décision d'exécution

ALLOW : exécuter après les contrôles applicables.
DENY : arrêter avant l'effet.
REQUIRE_APPROVAL : faire décider une personne autorisée.

Le protocole et ses 14 scénarios de référence définissent la comparaison entre autorisations individuelles et évaluation contextuelle : fausses alertes, violations manquées, latence, qualité des explications et observations incomplètes. L'expérimentation MCP contrôlée doit établir la faisabilité et l'apport pratique à la décision.

Hypothèse, architecture et protocole