🏭 De l’idée à la prod

mon usine logicielle perso

Bienvenue dans l’ère du casual programming !



Frédéric Blanc · 2026

Qui suis-je ?

- 👋 Frédéric Blanc
- 💼 développeur depuis 2000
- 🧭 tech lead chez SCIAM
- ☕ co-organisateur du MarsJUG
- 🛠️ toujours eu envie de créer mes propres applis !



www.marsjug.org



Mon objectif :

- 🚀 **Développer et utiliser mes propres applis**, dans un cadre familial, pour résoudre **MES** problèmes

Mes contraintes

- ⏱️ peu de temps à y consacrer (y a trop de trucs à faire, et pas assez de temps!!)
- 📈 je veux voir des avancements
- 🔒 je ne veux pas me prendre la tête avec la sécurité, la maintenance…

En gros : des applis utiles, donc maintenues sur le long terme, sans y consacrer trop de temps.

Le premier problème 🏖️

Je pars souvent en vacances avec des groupes… compliqués :

- 📅 tout le monde ne reste pas le même nombre de jours
- 👶 avec ou sans enfants
- 🧮 qui doit combien à qui ?

Ma solution : un Tricount-like ! 💡

L’histoire de mes réponses à ce problème

- 👥 **Ma première appli** : 2001, un clone de Tricount en C++
- 🗄️ **Ma deuxième appli** : 2025, un clone de Tricount en Java, déployé sur internet

Pourquoi **DEUX** fois la même appli ? 🤔

- j’ai perdu l’appli…
- ainsi que les sources ;p



Pourquoi attendre 25 ans ? ⏳

Une appli web demande beaucoup plus d'énergie qu'une appli desktop :

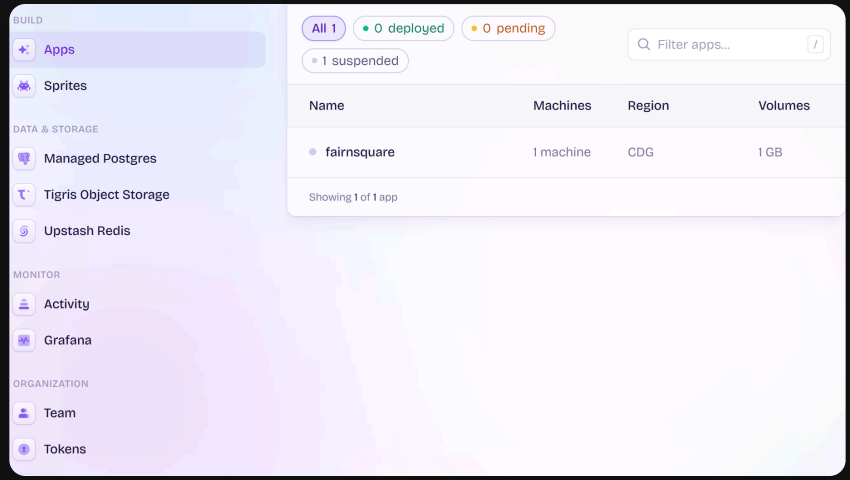
- 🔒 la sécurité
- 🧹 la maintenance
- 🏠 où la déployer ?
- 🤯 quel framework JS ? *(un nouveau chaque semaine)*

L'IA a permis de résoudre beaucoup de ces problèmes !

Ma première boîte à outils

Le but : ne pas payer cher, avec le moins de choses possible à gérer. 💸

- ☁️ Fly.io
- 💻 mon PC (Claude Code + IntelliJ)
- 🐙 GitHub
- ⚙️ stack :
  - Quarkus + Svelte
  - pas de base de données
  - pas d'auth
  - monitoring basique



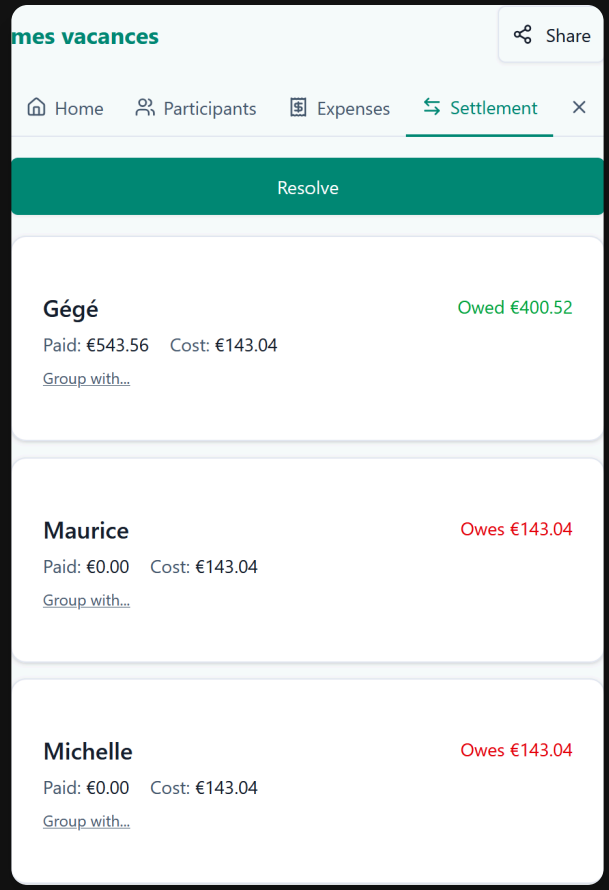
Process initial 🎈

1. imaginer une feature
2. ouvrir le projet dans IntelliJ
3. demander à Claude de faire un plan
4. relire le plan
5. demander à Claude d’implémenter le plan
6. tester sur mon PC
7. faire la PR
8. merger la PR
9. déployer sur Fly.io
10. recommencer

Le résultat : FairNSquare 🎉

- ⚙️ Techno : Quarkus + Svelte
- 📄 Persistance : fichier JSON
- 🧩 Captcha maison…

👉 fairnsquare.web.asymetrik.org



Fly.io : le bilan

👍 Les plus

- ✅ `fly deploy` et c'est en ligne
- ✅ Domaine perso + TLS
- ✅ Rien à maintenir

👎 Les moins

- ajouter une base : ❌ cher
- ajouter l'OAuth : ❌ cher
- ajouter une autre appli : ❌ cher 💸💸💸

Bref, très bien pour commencer, mais vite limité, et ça demande pas mal d'investissement personnel : quelques soirées et de gros bouts de week-end passés sur FairNSquare.

Le vrai problème 😩

Ça n’avance que lorsque je suis devant l’écran.

Or je veux que ça avance dès que j’ai 5 minutes.

Bref, ce que je veux, c’est du **casual programming** ! 🎮

Mes nouvelles exigences

- 🏃 mes projets doivent avancer tout le temps
- 🗄️ je dois pouvoir utiliser une base de données
- 🔑 je dois pouvoir utiliser de l'OAuth
- 🧹 je veux me débarrasser de la compilation du frontend

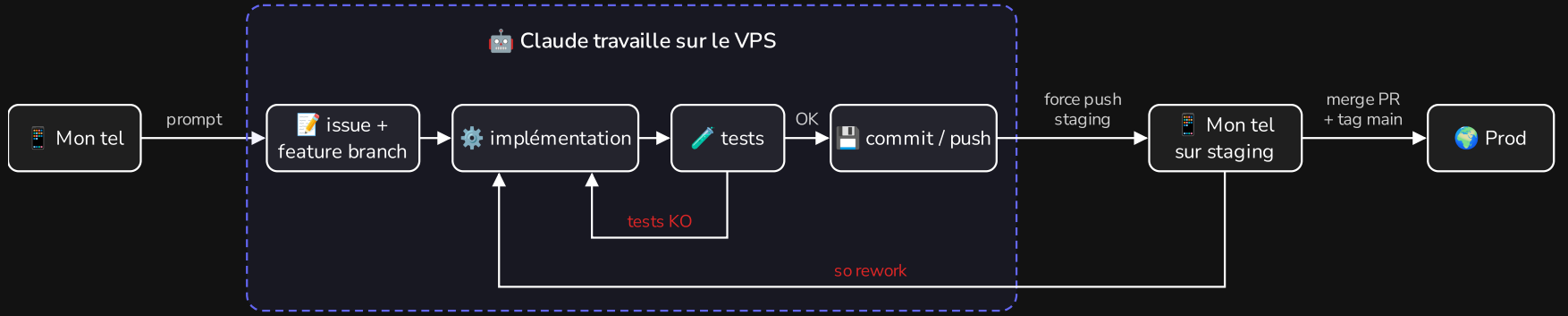
Deuxième boîte à outils

- 🖥️ un VPS (OVH)
- 📦 PaaS auto-hébergé : Dokploy
- 🤖 tmux + Claude en remote sur le VPS, avec Java et Maven
- 🐙 GitHub / Codefloe (vive l'Europe 🇪🇺)
- ⚙️ stack :
  - Quarkus / Vidocq
  - Modular
  - frontend : web components (pas de compilation 🙌)
  - base : Postgres, liquibase
  - auth : Pocket ID
  - monitoring JVM

📊 **Grafana Cloud free tier** *(en cours d'exploration)*

Résultat : une pipeline très efficace 🔧

Plus souple que Fly.io : on peut faire ce qu'on veut ! 💪
Un investissement initial plus conséquent… mais Claude rend tout cela accessible.



déploiements automatiques par Dokploy 📦

✨ simple ! ✨

Avant / après

🎈 Avant (Fly.io)

1. imaginer une feature
2. ouvrir IntelliJ
3. faire un plan avec Claude
4. relire le plan
5. implémenter avec Claude
6. tester sur mon PC
7. faire la PR
8. merger la PR
9. déployer
10. recommencer

🚀 Après (VPS + Dokploy)

1. je prompte Claude
2. je teste sur staging
3. je merge
4. je taggue main

👉 Claude travaille **pendant que je ne suis pas là.**

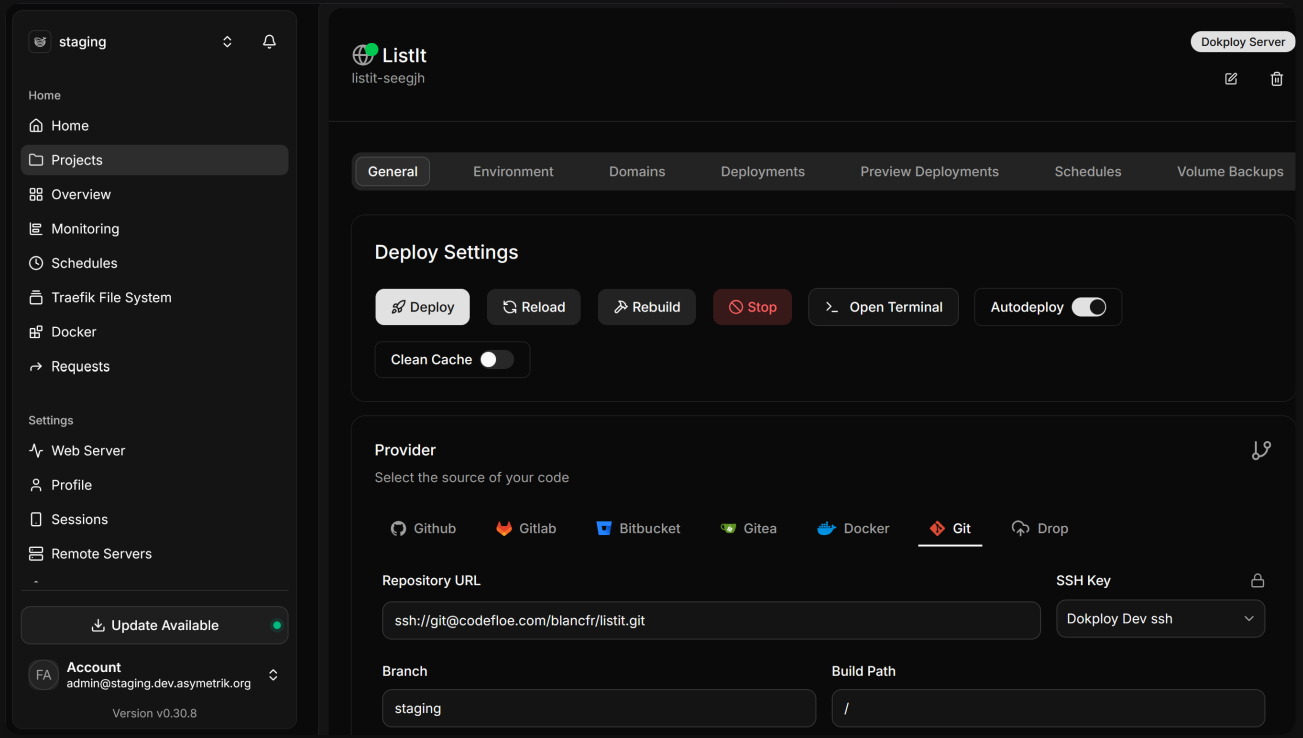
Ça, c'est du **casual programming** ! 🎮

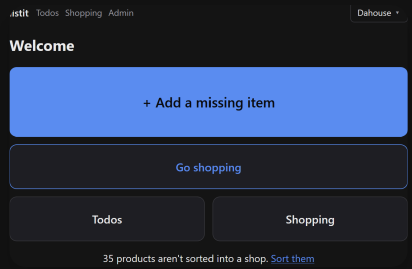
Les règles pour que ça marche 📏

- 🧪 un environnement de staging
- 🧱 une architecture en monolithe modulaire
- 🐜 des petites features, livrées une par une
- 🛡️ pas d’appli ni de données critiques : je ne suis pas expert sécurité !

👀 Je surveille plus l’archi que le détail du code (merci Modular !).

En images : Dokploy & ListIt 📸





Modular : l’archi sous contrôle 🧱

OFF

```java
/**
 * The family's shopping list, one kind of list: a single list per tenant that never ends, filled from the
 * family's catalog of products, which remembers every product ever added, its variants, its category and how
 * often it was bought, and with one-off items, never in the catalog, each for a shop or none; sorted, in a shop,
 * by the shop's aisles ({@code shop}).
 */
@Module(name = "shoplist", exports = {},
        requires = {"shop", "tenant", "idgenerator", "permission", "jakarta.cdi", "jakarta.inject", "jakarta.data",
                "jakarta.persistence", "jakarta.transaction", "jakarta.ws.rs"})
@Req(id = "SHP-1", value = "When a product is added to the list by name, the shoplist shall add the catalog's product of that name, ignoring case and accents, or
@Req(id = "SHP-2", value = "When a product already on the list in the same variant, ignoring case and accents, or alone, is added, the shoplist shall keep its on
@Req(id = "SHP-3", value = "If a product name or a one-off item's name is missing, blank or longer than 100 characters, then the shoplist shall reject it.")
@Req(id = "SHP-4", value = "If a quantity is longer than 30 characters, then the shoplist shall reject it.")
@Req(id = "SHP-5", value = "When a product name, a one-off item's name, a variant name, a quantity or a comment is stored, the shoplist shall store it without le
@Req(id = "SHP-6", value = "The shoplist shall show the list's entries, each with its product's name, its variant, its quantity and its comment, or, for a one-of
@Req(id = "SHP-7", value = "When an entry's quantity is changed, the shoplist shall store the new one in its place.")
```

chaque module déclare ce qu'il exporte, ce dont il dépend… et ses exigences

🔗 github.com/fblan/modular

Modular : le rapport 📊

shoplist

`org.asymetrik.listit.biz.shopping.shoplist`

Exports	nothing
Requires	shop tenant idgenerator permission jakarta.cdi JPMS module jakarta.inject JPMS module jakarta.data JPMS module jakarta.persistence JPMS module jakarta.transaction JPMS module jakarta.ws.rs JPMS module
Size	47 types, 215 public methods, 562 lines of code
Exported	0 of 47 types, 47 internal

Uses

idgenerator	▸ 3 uses
permission	▸ 4 uses
shop	▸ 25 uses
tenant	▸ 14 uses
jakarta.cdi JPMS module	▸ 13 uses
jakarta.data JPMS module	▸ 18 uses
jakarta.inject JPMS module	▸ 11 uses
jakarta.persistence JPMS module	▸ 12 uses
jakarta.transaction JPMS module	▸ 5 uses

Used by

No other module.

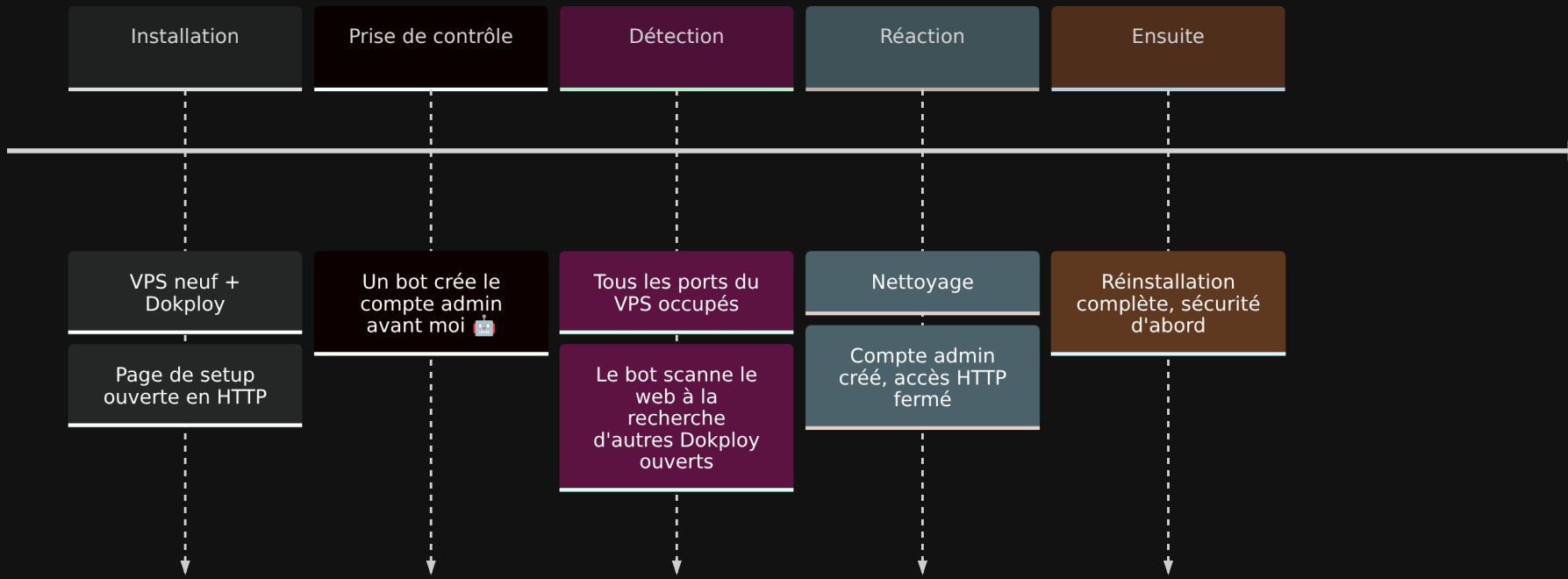
Un long fleuve tranquille ? 🛶

Pas vraiment...

🚨 L'incident 🚨

J'ai mis 6 minutes à finir d'installer Dokploy...
En 6 minutes, j'ai été hacké...
Et il m'a fallu une semaine pour m'en rendre compte...

Chronologie d’un hack 🕵️



Les bots scannent les nouvelles IP en quelques minutes. Sécuriser *avant* d’exposer. 🏃💨

Conclusion

Un VPS + Claude en remote + Dokploy => c'est top 🎉

- ⏱️ **peu de temps ? →** Claude avance sans moi
- 📈 **voir des avancements ? →** 3 applis en quelques semaines, livrées pas à pas
- 🔒 **sécurité, maintenance ? →** Dokploy + Claude s'en chargent… à condition de sécuriser dès le départ 😄
- 🚫 **et surtout :** pas d'applications critiques, pas de données sensibles : le vrai devops est un métier !
- 💶 **le coût ? →** VPS ~50 €/an + nom de domaine ~10 €/an + Claude ~20 €/mois

 Des questions ?

 Merci !

github.com/fblan