Stage 1 — Idea Development / Développement de l’idée

English · Français

English Version

Working title: TrueNAS Mobile
Author: Ivane Bagashvili (Holberton School)
Date: *2025-08-18 to 2025-08-29*

---

0) Team Formation

This is a solo project. I act as Product Owner, Mobile Engineer, API Integrator, and QA/Release. I define scope and acceptance criteria, build the Flutter app, integrate the TrueNAS REST API with secure token-based auth, test endpoints in Postman, and prepare releases. I keep a brief 09:00 daily check-in note (yesterday/today/blockers). Git workflow uses `main`, `dev`, and `feat/...` branches merged via PR. Work is tracked in GitHub Issues/Projects (list view); decisions in Notion; development in Android Studio + Flutter.

---

1) Research & Brainstorming

Methods: Mind Mapping, SCAMPER, “How Might We…”, and a competitive scan (app stores and TrueNAS/self-hosted forums).

Evidence included: sketches, a mind map, and a raw ideas list (Crazy-8s).

---

2) Initial Ideas (explored & rejected)

Two alternative concepts were explored; summaries follow.

Multi-NAS/Cloud backup orchestrator. Orchestrate backups across ZFS/BTRFS/SMB/S3 from one app. Rejected for very high cross-platform complexity, critical restore paths and data-loss risk — too big for an MVP.

“Jarvis” assistant (ASR/NLP). General voice assistant (wake-word, STT/NLU, dialog). Rejected for heavy ML stack, latency/privacy concerns, and long integration time.

Rejected ideas — summary

Idea	**+ Strengths**	**– Weaknesses**	**Why rejected**
Multi-NAS/ Cloud backup orchestrator	High value for complex environments; clear differentiation; strong learning (ZFS/BTRFS/ SMB/S3).	Very high cross-platform complexity; **critical restore** scenarios; data-loss risk; heavy responsibility/support.	**Out of scope for MVP**: risk and effort too high for the timeline; huge test matrix.
“Jarvis” assistant (ASR/ NLP)	Wow effect; ML/NLP learning; extensibility potential.	Heavy ML chain (wake-word, STT/NLU, dialogue); latency/privacy; long integration time.	**Time-to-MVP too long** and complexity too high for the stage timeline.

> **If revisited later:** Orchestrator limited to **TrueNAS → S3** only (single target, documented restore path).
> **Jarvis trimmed:** simple **local** commands (service ON/OFF) without dialogue/NLU, or voice → fixed intents.

---

3) Idea Evaluation

Weighted criteria: Impact/Need 25%, Feasibility 20%, Time-to-MVP 20%, Monetization 20%, Differentiation 10%, Learning 5%.

Decision matrix (1–5, higher is better):

Idea	Impact	Feasibility	Time	Monetization	Differentiation	Learning	**Weighted Total**
TrueNAS Mobile	5	4	4	4	4	4	**4.25**
Multi-NAS/ Cloud backup orchestrator	4	2	2	3	4	4	3.00
“Jarvis” assistant	4	2	1	2	4	5	2.65

Reading: TrueNAS Mobile offers the best value-to-risk-to-speed trade-off for Stage 1.

---

4) Decision & Refinement

Why **TrueNAS Mobile** was retained: clear mobile need (see status and toggle a service in seconds), direct feasibility with TrueNAS v2 REST over LAN, and an obvious roadmap (notifications, multi-server, WebDAV file browser, remote access via Tailscale/Cloudflare Tunnel).

Problem statement: too many steps on mobile, no secure token storage + biometrics, no quick alerts (planned post-MVP).

MVP solution: dedicated app with encrypted-token login and biometric lock, instant dashboard (CPU/RAM/uptime), and reliable ON/OFF switches for core services.

MVP success criteria: status visible $\leq$ **2 s** after app open; a service toggled in $\leq$ **3 taps** with clear confirmation; $\geq$ **80%** of testers find it faster than the Web UI.

Target users: home-lab/self-hosters and small teams that want a quick, secure way to manage TrueNAS from a smartphone.

Expected outcomes (MVP): secure login, clear dashboard, reliable ON/OFF service control.

---

5) MVP — Description & Stack

Summary: secure Flutter app to monitor/control a TrueNAS over LAN (token auth, encrypted storage, dashboard, services).
Challenges: self-signed certificates, error UX, tests.

MVP stack (Stage 1):

- **Language:** Dart · **Framework:** Flutter (Material 3, dark by default) · **Android min SDK:** 24
- **Navigation:** go_router · **State:** flutter_riverpod · **HTTP:** Dio
- **Models/JSON:** Freezed + json_serializable
- **Secure storage:** flutter_secure_storage (Keystore/Keychain)
- **Biometrics:** local_auth
- **Local cache (optional):** Isar (recommended) / Hive; SQL: sqflite/drift
- **API:** TrueNAS v2.0 (REST/JSON)
- **CI/CD:** GitHub Actions · **Quality:** flutter_lints, flutter_test, mocktail

---

6) Process & Documentation

The idea funnel is documented with strengths, weaknesses, and reasons for rejection. The MVP summary captures goals, risks, and success metrics. Traceability is ensured via GitHub Issues/Projects (list view), linking issues to commits/PRs, and maintaining a release changelog.

---

7) Plan & Milestones

- **W1 — Foundation:** wireframes, data contracts, Material 3 theme, navigation/state, Mock API.
- **W2 — Core MVP:** login + secure storage, dashboard, reliable ON/OFF for services, basic controller tests, lint/CI.
- **W3 — Polish & demo:** error/loading states, biometrics, README, short demo video, tester feedback.
- **(Stretch) W4 — Read-only view:** disks/datasets.

---

8) Risks & Mitigation

Self-signed certificates and home-network quirks are handled with safe paths (valid certs or secure tunnels) and controlled HTTP client behavior. Scope creep is constrained by a prioritized backlog and strict MVP boundaries. Monetization compliance (if Pro) is planned early. Solo workload is managed by small tasks and CI automation.

---

9) Technologies & Tools (detailed stack)

App: Flutter/Dart, Material 3, go_router, Riverpod, Dio (interceptors), Freezed/JSON, secure storage, local_auth, Isar/Hive.
Later: FCM (notifications), WebDAV (files), Tailscale/CF Tunnel (remote), relay (SaaS).
Project: GitHub repo + Projects (list view), GitHub Actions, flutter_lints/tests, mocktail, README/Notion.

---

10) Appendices

User stories

- As a NAS owner, I securely store URL/token to avoid re-entry.
- As a user, I see CPU/RAM/uptime on one screen.
- As a user, I quickly (de)activate SMB/NFS/SSH.

Acceptance criteria (MVP)

- Valid URL/token → redirect to Dashboard after login.
- Pull-to-refresh → updates < 2s on LAN for CPU/RAM/uptime.
- Toggling SMB → state persists after refresh.
- Invalid token → clear error, remain on login.

🇫🇷 Version Française

Titre provisoire du projet : TrueNAS Mobile
Auteur : Ivane Bagashvili (Holberton School)
Date : *18–29 août 2025*

---

0) Formation de l'équipe (Team Formation)

Je mène le projet en solo. J'assure le rôle de **product owner** et de **développeur mobile / intégration API**. Je clarifie le besoin et les critères d'acceptation, je code l'app Flutter, je connecte l'API TrueNAS et je prépare les builds. Je gère aussi la **QA/Release** avec un plan de tests simple et les éléments de publication.

Côté organisation : une courte note quotidienne à 09:00 (hier / aujourd'hui / blocages). Dans Git : `main` (stable), `dev` (intégration) et des branches `feat/...`, avec merges via PR. Outils : GitHub Issues/ Projects (vue liste), Android Studio + Flutter, Postman, Notion/Docs.

---

1) Recherche & Brainstorming

Je suis parti de mon usage quotidien : quand je veux juste vérifier le NAS depuis le canapé, l'interface Web n'est pas agréable sur téléphone et ça prend trop de clics. J'ai listé ces irritations, puis réalisé un **mind map** et quelques **croquis (Crazy-8s)** pour explorer les pistes. J'ai aussi regardé ce qui existe sur les stores et sur les forums TrueNAS/self-hosted pour voir où une petite app pouvait réellement aider. De là, plusieurs idées sont sorties, dont deux "grosses" que j'ai finalement écartées (voir ci-dessous), et TrueNAS Mobile qui restait la plus simple à rendre utile vite.

J'ai utilisé le mind mapping et formulé quelques questions « How might we » pour explorer les opportunités (ex. : Comment réduire les étapes pour désactiver un service ? Comment sécuriser un accès mobile sans stocker le mot de passe ?).

---

2) Idées initiales (concepts explorés & rejetés)

Au démarrage, deux pistes fortes ont été étudiées puis écartées pour des raisons de périmètre et de risque.

Orchestrateur de sauvegardes multi-NAS/Cloud. Piloter des sauvegardes vers ZFS/BTRFS/SMB/S3 depuis une seule app. **Intéressant**, mais **trop complexe et risqué** pour un MVP (interop multi-fournisseurs, restauration critique, risque de perte de données).

Assistant « Jarvis » (ASR/NLP). Assistant vocal généraliste (wake word, STT/NLU, dialogues). **Attractif**, mais **chaîne ML lourde** et **temps d'intégration trop long** pour le calendrier.

Idées rejetées — synthèse

Idée	**+ Forces**	**– Faiblesses**	**Pourquoi rejetée**
Orchestrateur de sauvegardes multi-NAS/Cloud	Forte valeur pour environnements complexes ; différenciation nette ; apprentissage sérieux (ZFS/BTRFS/SMB/S3).	Très forte complexité inter-plateformes ; scénarios de **restauration critiques** ; risque de perte de données ; responsabilité/ support élevés.	**Hors scope MVP** : risque et effort trop importants pour le délai ; matrice de tests énorme.
Assistant « Jarvis » (ASR/NLP)	Effet "wow" ; apprentissage ML/NLP ; potentiel d'extensions.	Chaîne ML lourde (wake-word, STT/NLU, dialogue) ; latence/ confidentialité ; intégration longue.	**Time-to-MVP trop long** et complexité élevée pour le calendrier du stage.

> **Si revisité plus tard :** Orchestrateur limité à **TrueNAS → S3** uniquement (une cible, un chemin de restauration documenté).
> **Jarvis réduit :** commandes **locales** simples (ON/OFF services) sans dialogue/NLU, ou voix → intents fixes.

---

3) Évaluation des idées (Idea Evaluation)

Afin d'objectiver le choix, une grille pondérée a été utilisée. Elle met l'accent sur l'impact utilisateur et la rapidité d'atteindre un MVP fonctionnel, sans négliger la faisabilité et la monétisation.

Matrice de décision (1–5, plus grand = meilleur) :

Idée	Impact	Faisabilité	Temps	Monétisation	Différenciation	Apprentissage	**Total pondéré**
TrueNAS Mobile	5	4	4	4	4	4	**4.25**
Orchestrateur multi-NAS/ Cloud	4	2	2	3	4	4	3.00
Assistant "Jarvis"	4	2	1	2	4	5	2.65

La lecture de cette matrice montre que **TrueNAS Mobile** offre le meilleur compromis entre valeur, risque et délai.

---

4) Décision & affinage

J'ai choisi **TrueNAS Mobile** parce qu'elle répond à un besoin très concret : sur mobile, je veux voir l'état du serveur et activer/désactiver un service en quelques secondes, sans galérer dans l'UI Web. Techniquement, c'est aussi celle que je peux livrer rapidement : l'API REST v2 de TrueNAS est documentée et testable facilement sur le réseau local. Le périmètre du MVP est volontairement petit (connexion avec token chiffré, un tableau de bord clair, la liste des services avec un interrupteur) pour tenir le délai tout en offrant une vraie valeur. Les extensions à moyen terme sont déjà évidentes : notifications (SMART/températures), multi-serveurs, explorateur de fichiers via WebDAV, et un accès distant guidé (Tailscale/Cloudflare Tunnel).

Cible utilisateur : utilisateurs de TrueNAS à domicile (home-lab) et petites équipes/PME qui veulent un accès rapide et sécurisé à leur NAS depuis un smartphone.

---

5) MVP — Description & Stack

Le MVP fait trois choses, mais il les fait bien : il se connecte à un serveur TrueNAS avec un token que je garde chiffré sur l'appareil ; il affiche tout de suite CPU, RAM et uptime ; et il me laisse activer/désactiver les services essentiels (SMB, NFS, SSH) sans naviguer dans dix écrans. Je n'essaie pas de tout couvrir au début : je préfère une base solide, rapide et sûre.

Côté technique, je reste sobre : **Flutter/Dart**, **Material 3** pour le look, **go_router** pour la navigation, **Riverpod** pour l'état. **Dio** gère les requêtes et **Freezed/json_serializable** les modèles. Le token et l'URL sont dans **flutter_secure_storage**, avec **local_auth** pour le verrouillage biométrique. Si besoin d'un peu d'hors-ligne, **Isar** fera un cache léger. L'API ciblée est **TrueNAS v2 (REST/JSON)**. Je mets une petite chaîne **GitHub Actions** pour lancer lint, tests et builds.

6) Processus & Documentation

- **Idées considérées** : forces/faiblesses/raisons de rejet documentées.
- **Synthèse MVP** : impact attendu, risques, métriques de succès.
- **Traçabilité** : GitHub Issues/Projects (vue liste), issues ↔ commits/PR, changelog par release.

---

7) Plan & Jalons

- **S1 — Squelette :** maquettes, contrats de données, thème, navigation (GoRouter), état (Riverpod), Mock API.
- **S2 — Cœur du MVP :** login + stockage sécurisé, dashboard, ON/OFF services, tests de base, lint/CI.
- **S3 — Finitions & démo :** états d'erreur/chargement, biométrie, README, vidéo de démo, retours testeurs.
- **(Option) S4 — Lecture seule :** disques/datasets.

---

8) Risques & Atténuation

Le sujet sensible, c'est la gestion des certificats auto-signés et, plus largement, les aléas réseau à la maison. Je documente des chemins sûrs (certificat valide quand c'est possible, ou tunnel type Tailscale/Cloudflare) et je contrôle le comportement du client HTTP pour éviter les mauvaises surprises. L'autre piège, c'est l'emballement du périmètre : je garde un backlog avec des priorités et je m'interdis d'ajouter une feature si elle ne sert pas la démo MVP. Enfin, pour la publication avec une version Pro, j'anticipe la facturation Google Play pour ne pas tout découvrir au dernier moment.

---

9) Technologies & Outils (stack détaillé)

App : Flutter/Dart, Material 3, go_router, Riverpod, Dio (interceptors), Freezed/JSON, secure storage, local_auth, Isar/Hive.
Plus tard : FCM (notifications), WebDAV (fichiers), Tailscale/CF Tunnel (accès distant), relay (SaaS).
Projet : GitHub repo + Projects (vue liste), GitHub Actions, flutter_lints/tests, mocktail, README/Notion.

---

10) Annexes

User stories

- En tant que propriétaire de NAS, je stocke URL/token en sécurité pour éviter la ressaisie.
- En tant qu'utilisateur, je vois CPU/RAM/uptime sur un seul écran.
- En tant qu'utilisateur, je (dés)active SMB/NFS/SSH rapidement.

Critères d'acceptation (MVP)

- URL/token valides → redirection Dashboard après login.
- Pull-to-refresh → mise à jour < 2 s en LAN pour CPU/RAM/uptime.
- Bascule SMB → état persistant après refresh.

• Token invalide → message clair, on reste sur la connexion.