2024-2025

Rapport

Test et sécurité logiciel

Projet : Application de gestion des emprunts de matériel informatique

1. Introduction

Ce rapport décrit la démarche adoptée, les problèmes rencontrés et leurs solutions, ainsi que l'organisation de l'équipe pour le développement de l'application de gestion des emprunts de matériel informatique. Il met en avant les stratégies utilisées pour structurer le travail, optimiser la collaboration et surmonter les défis techniques et organisationnels. L'objectif est d'exposer les choix faits tout au long du projet et de valoriser les approches qui ont contribué à sa réussite.

2. Méthodologie et organisation de l'équipe

Méthode de Travail : Cycle en V

Nous avons opté pour un cycle en V, garantissant une séparation claire entre les phases de spécification, développement et tests. Chaque phase validée servait de base à la suivante, assurant une meilleure maîtrise des exigences et des corrections précises en cas de défauts.

Répartition des Rôles

L'équipe était composée de six membres avec des responsabilités spécifiques :

- Chef de projet : Coordination et suivi du projet
- Responsable IHM : Développement et conception de l'interface utilisateur
- Responsable Base de données : Gestion du stockage des données et intégration
- Responsable des tests : Élaboration et exécution du cahier de tests
- DevOps / CI-CD : Mise en place et automatisation des tests via GitLab CI/CD
- Rédacteur technique : Rédaction des spécifications et documentation

3. Choix techniques

Nous avons utilisé Angular comme framework de développement car un des membres du groupe travaille déjà en entreprise avec et nous avons décidé d'utiliser pour le projet.

Pour la base de données (Firebase), l'automatisation des tests (Cypress) nous sommes restés sur ce qui a été proposé par le prof.

Pour ce qui est de la rédaction des tests, nous avons décidé de travailler avec le logiciel Squash test management parce que c'est un outil ad hoc pour les tests et aussi parce

qu’avec Excel, nous n’avons pas toutes les options proposées par Squash et qu’au niveau personnel, nous n’allons rien acquérir comme expérience puisque nous l’utilisons toujours pour nos tests dans les autres projets.

Pour la gestion du code source et des versions, nous avons utilisé GitHub car nous maîtrisons tout cet outil.

Pour l'automatisation des tests, nous avons choisi Cypress car il s'intègre parfaitement avec Angular et TypeScript, technologies utilisées pour notre projet. Cypress offre une approche moderne et intuitive pour les tests end-to-end, avec une exécution rapide et une prise en charge native d'Angular, ce qui simplifie les tests des composants interactifs. De plus, son architecture basée sur un moteur JavaScript nous permet d’écrire des tests de manière fluide sans configurations complexes, rendant ainsi notre workflow de test plus efficace et naturel dans notre environnement de développement.

Enfin, la communication et la coordination de l’équipe ont été facilitées grâce à Microsoft Teams, qui permet un échange rapide et structuré des informations, améliorant ainsi l’efficacité du travail collaboratif.

L’application repose sur une architecture modulaire, où chaque composant (gestion des réservations, affichage des appareils, authentification) interagit de manière fluide via Angular, garantissant ainsi une meilleure maintenabilité et évolutivité du projet.

4. Développement et Fonctionnalités

L’application a été conçue avec plusieurs fonctionnalités essentielles :

- **Authentification** : Page de connexion avec gestion des rôles (Administrateur/Emprunteur).
- **Gestion des matériels** : Ajout, modification, suppression et consultation.
- **Réservation** : Un utilisateur peut réserver un appareil en spécifiant une période.
- **Recherche et affichage** : Une barre de recherche permet de filtrer les matériels disponibles.

Ce sont les fonctions de base qui étaient imposés dans l’expression des besoins. Toutefois, nous avons apporté des rajouts sur certaines fonctions pour plus de cohérence. Par exemple, le fait de rajouter des contraintes pour un mot de passe, ajouter un mot de passe lors de la création de l’utilisateur ; ce sont des paramètres qui ne figuraient pas dans l’expression des besoins et que nous avons rajouté.

5. Mise en place des tests automatisés

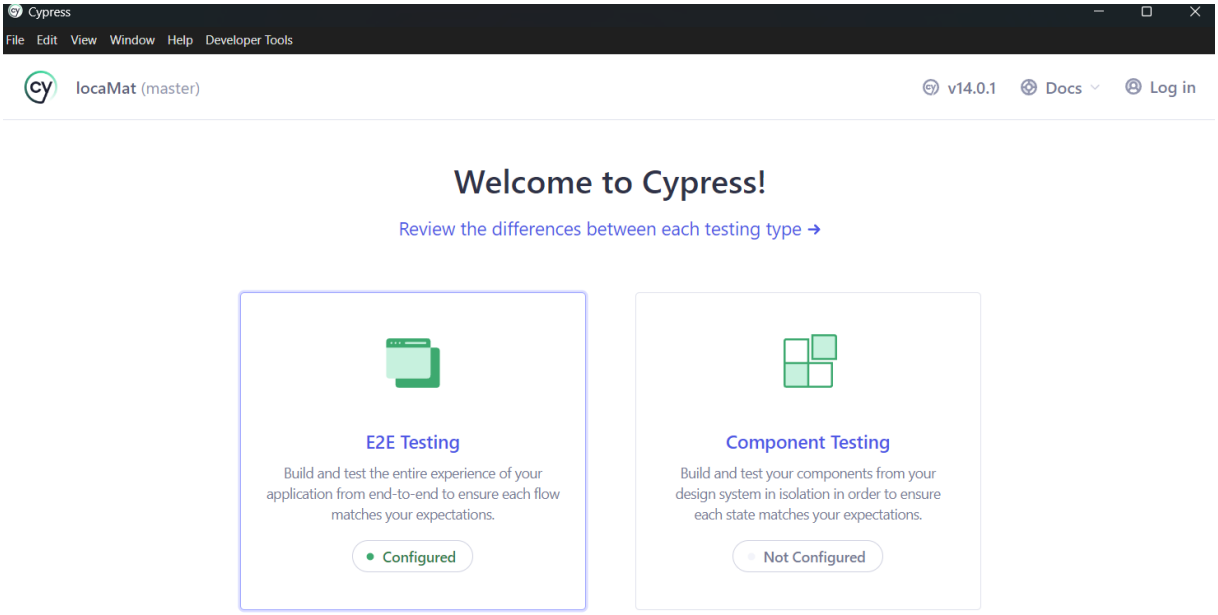
Pour automatiser les tests end-to-end de notre application Angular, nous avons opté pour **Cypress**, un outil puissant et intuitif, bien adapté à notre stack technique.

Installation et configuration

Nous avons commencé par installer Cypress dans notre projet Angular en exécutant la commande suivante :

```
Npm install cypress
```

On ouvre l'interface liée à notre projet avec la commande suivante :
`Npx cypress open`



Cela a généré un dossier `cypress/` contenant les fichiers de configuration et des exemples de tests. Nous avons ensuite adapté le fichier `cypress.config.js` pour correspondre aux besoins de notre projet, en définissant l'URL de base et les paramètres des tests.

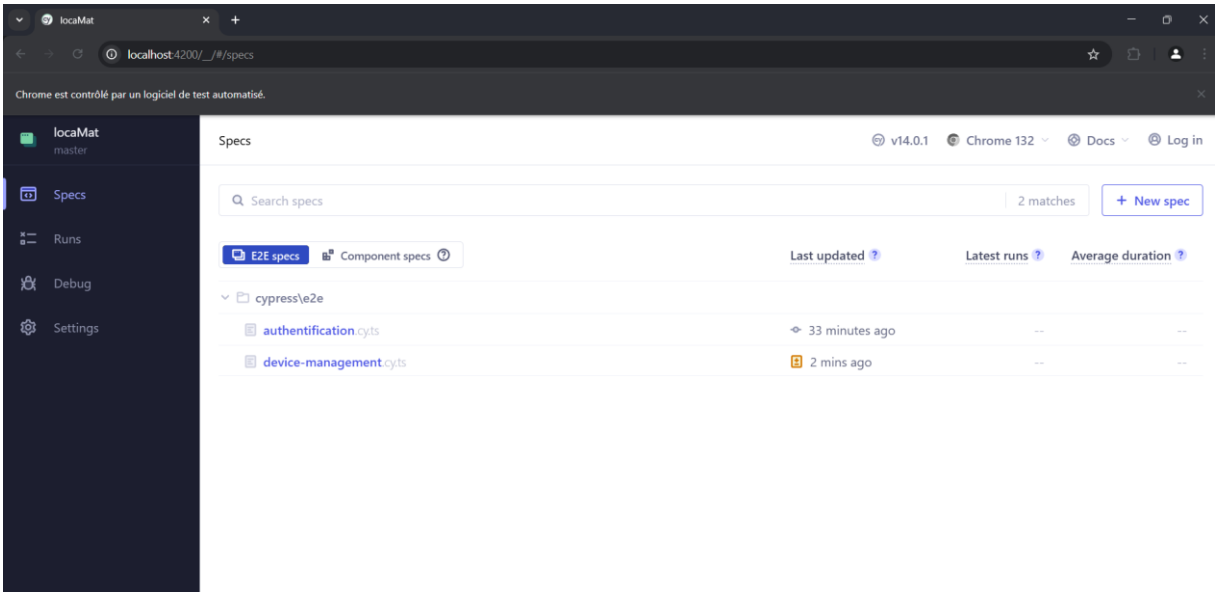
Écriture des tests

Nous avons commencé par implémenter les premiers cas de tests essentiels, notamment ceux liés à l'authentification (connexion avec des identifiants valides ou invalides) et à la gestion des matériels (ajout, modification et suppression de matériels). Ces fonctionnalités étant cruciales pour l'application, nous avons choisi de les automatiser en priorité afin d'assurer leur bon fonctionnement dès le début du projet.

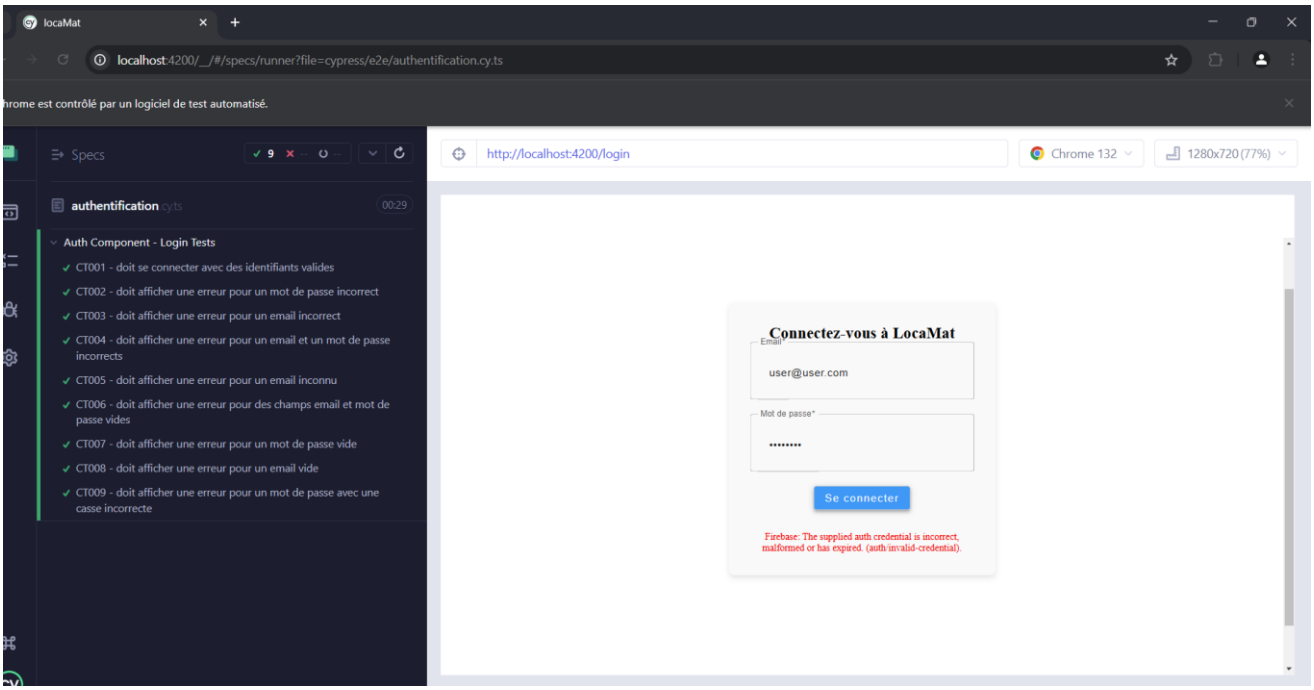
Grâce à cette première phase, nous avons pu bien comprendre le fonctionnement de Cypress, son système de sélection des éléments, ses commandes (`cy.get()`, `cy.click()`, `cy.type()`, etc.) ainsi que la manière de structurer nos scénarios de test de façon claire et efficace.

Bien que nous ayons rédigé un grand nombre de tests dans Squash TM pour couvrir l'ensemble des fonctionnalités de l'application, nous n'avons pas automatisé tous ces tests avec Cypress. Nous avons principalement automatisé les tests qui nous semblaient les plus critiques et ceux qui apportaient une réelle valeur ajoutée en termes de validation et de gain de temps. Les autres tests ont été réalisés manuellement, notamment ceux nécessitant des interactions complexes ou des validations plus spécifiques.

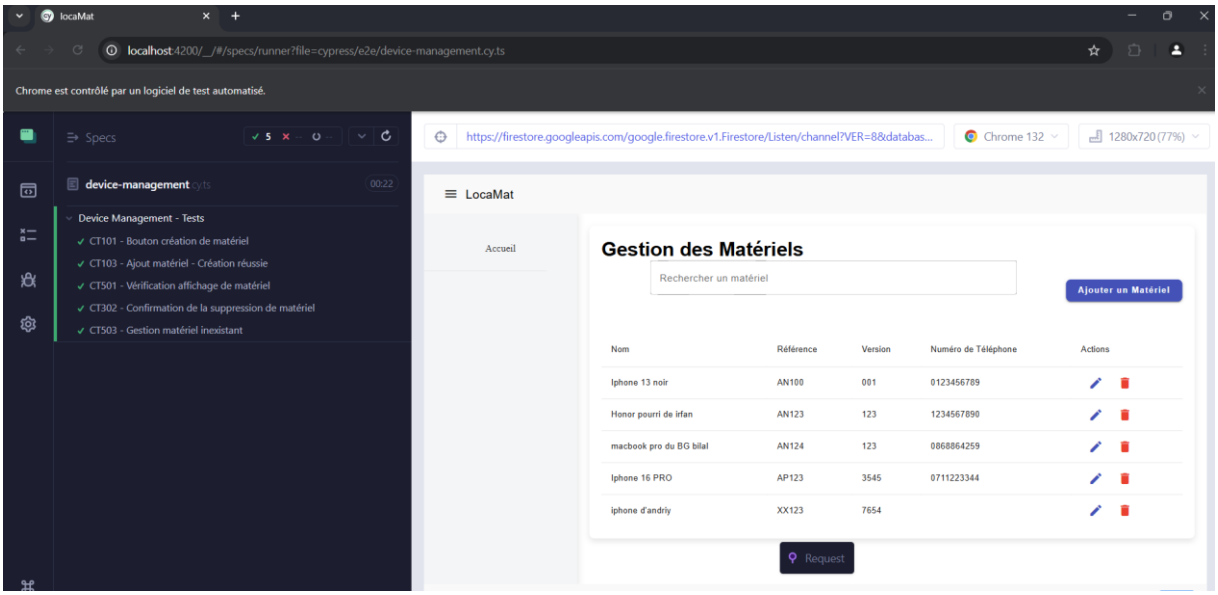
Les tests rédigés, on peut les retrouver directement dans l'interface dans le menu "Specs"



Nous avons validé les tests d'authentification conformément à ce qui a été listé dans SquashTM



De même pour les tests concernant la gestion des matériels :



On pourra retrouver le détail du code des tests directement dans le projet locaMat dans le dossier “cypress/e2e”

6. Problèmes rencontrés et solutions

Lors de l'intégration de Firebase avec Angular, nous avons rencontré plusieurs difficultés, notamment des incompatibilités de versions qui ont entraîné des erreurs lors de la récupération des données. De plus, la configuration des règles de sécurité Firestore a nécessité plusieurs ajustements pour garantir un accès sécurisé tout en permettant les opérations nécessaires. Pour résoudre ces problèmes, nous avons aligné les versions des

dépendances, utilisé AngularFire pour faciliter les requêtes Firebase et testé les règles Firestore afin de trouver le bon équilibre entre sécurité et accessibilité.

La mise en place du système de réservation a également posé des défis, notamment l'affichage automatique de la prochaine date disponible et l'interdiction de réserver une date antérieure à aujourd'hui. Pour y remédier, nous avons développé une fonction qui analyse les dates déjà réservées et suggère la plus proche encore disponible. Par ailleurs, une validation a été mise en place à la fois sur le front-end et le back-end afin d'empêcher la sélection d'une date passée, garantissant ainsi une gestion cohérente et fluide des réservations.

L'installation et l'initialisation de SquashTM n'a pas été évidente notamment parce que c'était la première que l'on utilisait Docker qui est utile dans notre cas pour lancer les containers SquashTM. Le plus gros souci a été la compréhension de à quoi servait Docker dans notre cas précis, une fois que c'est fait la prise en main été assez bonne grâce à des tutoriels notamment.

7. Conclusion

Ce projet nous a permis d'appliquer une approche avec le cycle en V. Malgré quelques défis techniques, nous avons su apporter des solutions et optimiser certaines fonctionnalités pour améliorer l'expérience utilisateur. Grâce à une répartition claire des rôles et l'usage d'outils adaptés, l'application est stable a beaucoup évolué. Cette expérience nous a renforcés en gestion de projet, développement et résolution de problèmes.