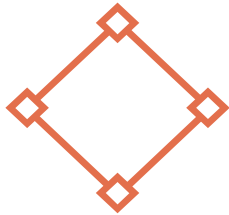




HOME
SYNCK



Getway

Dossier de projet

PJS4

Encadré par M. Denis Poitrenaud

AGUIAR Daniel
Jules DOUMECHE
Thibault HENRION
Gwénolé MARTIN
Anicet NOUGARET
Hélène TE
Jules VANIN

DEDICACE

Nous dédions ce travail à la communauté open source, pour son apport essentiel à la réalisation de ce projet, tant par le biais de programmes et de bibliothèques libres que par les forums d'entraide qui regorgent de ressources pour comprendre les problématiques auxquelles nous avons dû faire face.



REMERCIEMENTS

Nous souhaitons remercier l'ensemble des acteurs qui ont permis à ce projet d'exister, et qui nous ont apporté leur soutien pendant sa conception.

Merci en premier lieu à M. Denis Poitrenaud, qui nous a encadrés pendant le déroulement de ce projet et qui a donné des conseils très utiles pour son aboutissement.

Merci également à Mme Pauline Flepp, pour ses conseils précieux pour la rédaction de ce rapport et la préparation de notre soutenance.

Merci à Mme Bedos, et de façon plus générale à l'ensemble du personnel administratif de l'IUT, pour leur aide dans la recherche de salles pour nous réunir chaque semaine, malgré un contexte peu favorable.

Merci enfin à M. Jean-François Brette, pour la gestion de l'organisation générale de ces projets de fin d'études.

L'ensemble du groupe

Table des matières

Introduction

Présentation

Le protocole de synchronisation Homesynck	8
La base de données	8
Le micro-service de validation des numéros de téléphones	9
Le site internet de présentation	10
Le serveur de Homesynck	10
Le reverse-proxy	11
L'application mobile Getway	11
Le kit de développement Java Homesynck	12
Back-office de l'application	13
Front-office de l'application	14

Open source

Outils et techniques

Organisation

Conclusion

Difficultés rencontrées	25
Possibilités d'évolution	26

Bibliographie

Glossaire

Annexes

Résumés en anglais	32
Poster du projet.....	39

Introduction

Quelles sont aujourd'hui vos possibilités si vous souhaitez synchroniser vos données entre vos appareils ? Vos contacts par exemple ?

Il est possible de les stocker sur le cloud, gratuitement ou à faible coût, mais souvent au prix de quelques données, qui sont collectées par les fournisseurs (Google, Microsoft, etc.). L'autre solution consiste à installer un NAS à son domicile, et à paramétrer la synchronisation, mais cette solution est coûteuse et complexe à installer pour un utilisateur moyen.

C'est ici qu'intervient la solution que nous avons développée dans le cadre de ce projet. Homesynck est un protocole de synchronisation, qui permet de synchroniser en temps réel vos dossiers.

L'idée du protocole de synchronisation nous est venue en cherchant une solution pour faire une application des gestions des contacts, pour les particuliers, avec un transfert des contacts en temps réel. La synchronisation en temps réel est courante pour les gestionnaires de mots de passe par exemple, mais les protocoles ne sont pas disponibles pour la réutilisation (ce sont des protocoles dits « propriétaires »).

Pour illustrer l'usage de ce protocole, nous avons réalisé notre idée de départ, à savoir la création d'une application de contacts pour smartphone Android, en utilisant notre protocole, application que nous avons appelée Getway. Cette application mobile permet donc de gérer des contacts et de les synchroniser en utilisant le protocole Homesynck.

Pour ce projet, nous avons fait le choix de l'open source. C'est une pratique très répandue dans le milieu du développement qui consiste à rendre son projet accessible à tous, gratuitement. Un développeur souhaitant créer une application ou un logiciel pourra donc utiliser sans frais et sans limites Homesynck pour la synchronisation de fichiers, grâce aux kits de développements que nous avons développés et publiés sur le web.

- Annonce du plan

Dans ce rapport de projet, nous entrerons rapidement dans le cœur du sujet, avec une explication détaillée du protocole de synchronisation d'abord, et de l'application mobile Getway dans un second temps, en nous appuyant sur le schéma d'architecture du projet.

Nous pourrions ensuite compléter avec une présentation des nombreux outils utilisés pendant le projet et un point sur l'organisation de notre travail.

Nous consacrerons une dernière partie à l'open source et aux outils de développement que nous fournissons aux développeurs pour qu'ils puissent réutiliser notre protocole.

Jules Doumèche & Anicet Nougaret

Présentation

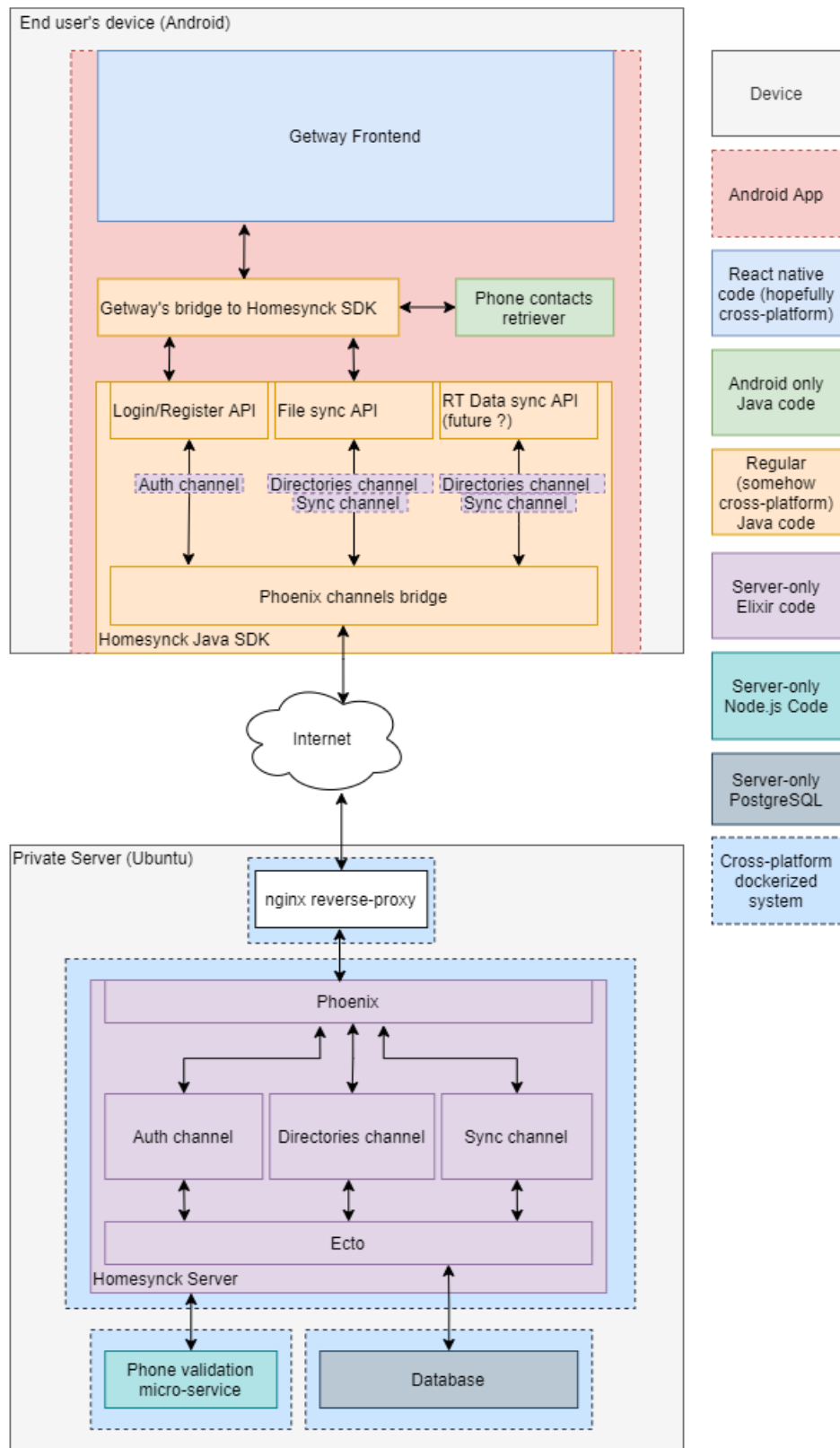


Schéma d'architecture de l'ensemble de notre projet

Point nomenclature : dans la suite de ce rapport, nous allons utiliser le mot « répertoire », nous l'utilisons ici dans le sens de « dossier » (de fichiers), et non pas dans le sens d'un répertoire de contacts.

Ce schéma d'architecture présente l'ensemble des composants de notre projet. Il nous servira de base pour la suite de ce rapport, et nous vous détaillerons dans les lignes qui vont suivre le fonctionnement de chaque élément.

Le protocole de synchronisation Homesynck

Le protocole de synchronisation est un élément majeur de notre projet. Il permet, comme son nom l'indique, de synchroniser des fichiers, et des dossiers, entre plusieurs ordinateurs/téléphones ou autres appareils.

Pour le nom, nous avons assemblé « Home », qui veut dire maison en anglais, et « sync » (en ajoutant un 'k') pour la synchronisation, car notre protocole sert à la synchronisation et il est « fait maison ».

Ce protocole est divisé en plusieurs parties que nous allons vous détailler ci-dessous.

Jules Doumèche

La base de données

La base de données fonctionne avec PostgreSQL, un système de gestion de bases de données (SGBD).

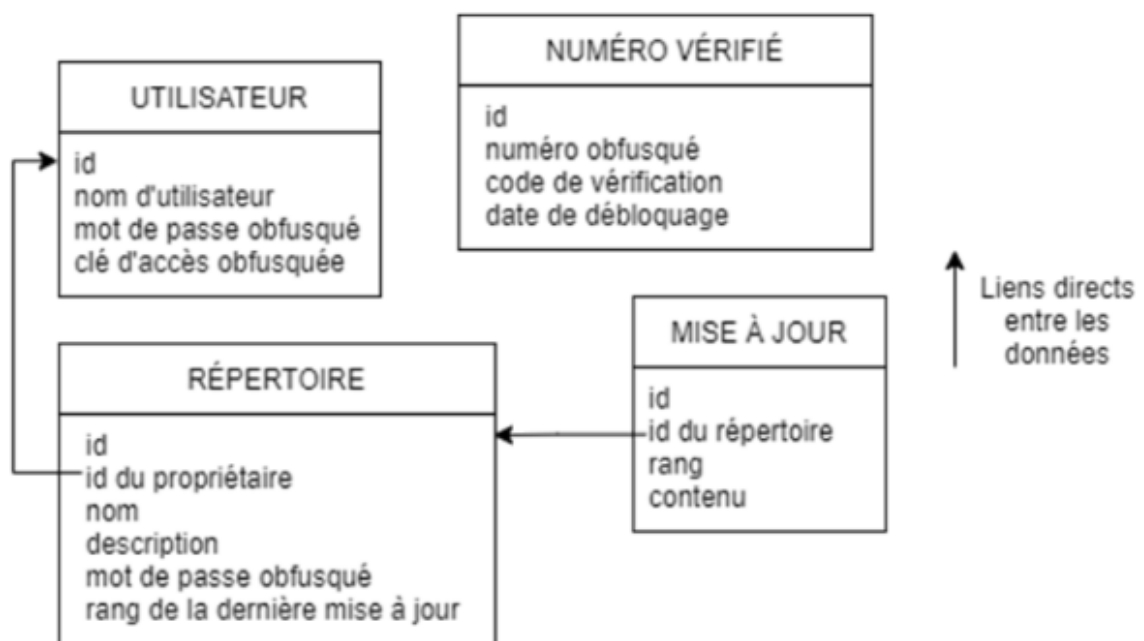


Schéma de la base de données

Elle est composée de 4 tables :

Une table Utilisateur qui contient les utilisateurs, avec entre autres le code d'identification (login), le mot de passe (chiffré) et une clé d'accès (chiffrée), qui est nécessaire pour sécuriser les échanges avec le dashboard (le protocole pour la communication web est beaucoup moins sécurisé que la communication au sein d'un socket).

Une deuxième table liée à l'utilisateur, qui contient les dossiers (table Répertoire) associés à l'utilisateur et les informations nécessaires pour permettre les mises à jour, elle-même liée à une troisième table (table Mise à jour) contenant les mises à jour.

Enfin, une dernière table séparée des précédentes stocke de façon sécurisée les numéros de téléphone qui ont déjà été utilisés pour créer un compte (un délai d'un mois a été mis en place entre deux créations de comptes avec le même numéro de téléphone pour éviter le spam).

Nous avons choisi de stocker les mises à jour dans notre base de données, car il nous était très compliqué de gérer autrement la mise à jour des dossiers. Il est possible d'envisager une évolution qui consisterait à supprimer les mises à jour antérieures pour améliorer la sécurité et l'efficacité du stockage.

Jules Vanin

Le microservice de validation des numéros de téléphone

Pour éviter que des utilisateurs malveillants tentent de créer de nombreux comptes, nous avons fait le choix de sécuriser l'enregistrement d'un nouveau compte pour le protocole de synchronisation avec une vérification du numéro de téléphone, effectuée par un serveur node.js.

Le serveur de validation reçoit du serveur « principal » (celui de Homesynck) un numéro de téléphone, un code de validation à 6 chiffres et une clé d'API que nous avons définie au préalable. Ces informations sont transmises par une requête POST, elles sont formatées en JSON.

Le serveur de validation envoie ensuite un SMS contenant le code de validation à l'utilisateur, qui devra le saisir dans l'application. Le serveur retourne un 0 au serveur principal si tout s'est bien passé, un autre chiffre permettant de reconnaître l'erreur dans le cas contraire. La clé d'API permet au serveur de validation d'authentifier le serveur principal, si la clé est bonne, il est sûr de « parler » avec le bon serveur. Pour une sécurité optimale, nous avons utilisé une clé de 256 caractères.

Actuellement, le serveur de validation fonctionne grâce à une API d'envoi de SMS, nommée Nexmo. L'envoi de SMS étant un processus coûteux, qui coûte environ une dizaine de centimes par SMS envoyé, nous avons choisi pour nos tests et pour le déploiement de retourner tout le temps 0 pour le moment, un développeur souhaitant utiliser le protocole pourra installer lui-même son propre valideur de numéro de téléphone, ou même supprimer cette fonctionnalité (mais il devra pour cela modifier le code du serveur principal).

Gwénolé Martin

Le site internet de présentation

Nous avons créé un site internet de présentation, qui regroupe tous les liens utiles pour démarrer le développement en s'appuyant sur le protocole Homesynck :

`homesynck.anicetnougaret.fr`

Pour les utilisateurs ayant un compte, il est aussi possible de se connecter pour voir en direct la synchronisation des fichiers.

Le serveur de Homesynck

Le serveur principal de Homesynck est un serveur Elixir OTP, c'est lui qui gère la réception et la redistribution des données aux différents appareils.

Pour faire simple, le serveur va gérer les différentes fonctions qui sont utilisées, comme l'inscription, la connexion et la synchronisation des fichiers pour un dossier et pour un compte utilisateur donné.

Le fonctionnement de ces fonctions étant détaillé dans la partie du kit de développement Java, et la partie Elixir étant complexe à expliquer en restant simple, nous nous concentrerons ici sur la communication entre le serveur et les autres composants.

Pour la communication entre les kits de développements (dont nous détaillerons le fonctionnement plus loin dans ce rapport) et le serveur, nous utilisons des WebSockets.

Un WebSocket est un canal de communication à double sens, contrairement à une simple requête HTTP. Le client (ici les appels aux fonctions des kits de développement jouent ce rôle) et le serveur peuvent donc communiquer quand ils le souhaitent sur ce canal.

Pour la communication avec le « dashboard » sur le site internet, un WebSocket à part est aussi utilisé.

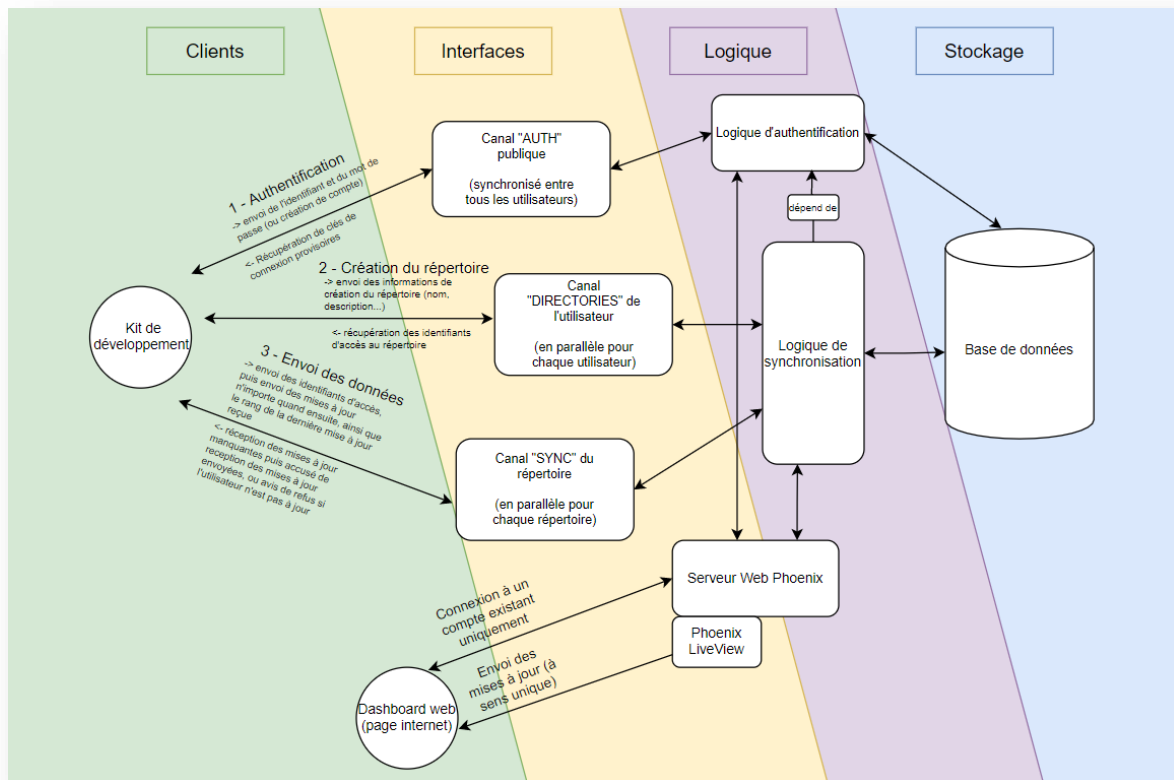


Schéma explicatif des échanges entre les différents composants

Anicet Nougaret

Le reverse-proxy

Pour sécuriser les échanges entre internet et le serveur, nous utilisons ce que l'on appelle un reverse-proxy, qui fonctionne grâce à nginx. Nous n'allons pas détailler ici le fonctionnement d'un reverse-proxy, mais il faut retenir qu'il sert à rendre accessible de façon sécurisée notre serveur principal depuis le web.

Jules Vanin

L'application mobile Getway

L'application mobile Getway est la partie « visuelle » de notre projet, elle nous permet de montrer ce que peut réaliser notre protocole, sans utiliser le maximum de ses capacités néanmoins.

Cette application, réalisée à l'aide de React Native, utilise aussi le kit de développement Java que nous avons réalisé, comme un exemple pour les développeurs qui voudraient utiliser l'application dans le futur.

Nous allons vous présenter ci-dessous les composants de cette application :

Le kit de développement Java Homesynck

Le kit de développement logiciel Homesynck (SDK) est une bibliothèque permettant de gérer la communication avec le serveur. Il donne accès à une panoplie de classes utilitaires (qui correspondent aux fonctionnalités du protocole).

Pour commencer, il permet de créer un compte pour les nouveaux utilisateurs ou bien de s'authentifier si l'on possède déjà un compte. Il permet ensuite de se connecter à un répertoire que l'on peut créer si celui-ci n'existe pas.

Il est également possible de définir un mot de passe à ce répertoire afin d'en sécuriser l'accès. Une fois connecté au répertoire, il est possible de modifier les fichiers grâce aux fonctions proposées par le SDK : une fonction pour éditer un fichier, ou le créer si celui-ci n'existe pas, et une fonction pour le supprimer.

Pour la synchronisation des fichiers, il suffit au développeur de faire appel à une fonction et le SDK se charge du reste. Lors de la première connexion, le SDK va créer deux dossiers, un dossier possédant les fichiers dans l'état dans lequel le serveur les lui a envoyés à la dernière connexion et un dossier qui contiendra les fichiers que l'utilisateur pourra modifier (voir schéma ci-dessous).

Lors de l'appel à cette fonction, le SDK va vérifier les différences entre ces deux dossiers. S'il trouve un fichier qui a changé, cela signifie que le serveur possède une version plus ancienne. On va donc générer un fichier de différences entre le fichier du serveur et celui modifié. Ce fichier de différences permet de recréer le nouveau fichier à partir de l'ancien, celui stocké sur le serveur. Ce fichier de différences est ensuite envoyé au serveur qui le renvoie à tous les appareils connectés au même compte. Les appareils qui reçoivent le fichier l'appliquent ensuite sur leurs propres fichiers. Afin d'éviter tout conflit, l'appareil qui envoie le fichier de différences l'applique uniquement au moment où il le reçoit depuis le serveur, comme tous les appareils, on pourrait résumer en disant que les mises à jour sont appliquées séquentiellement.

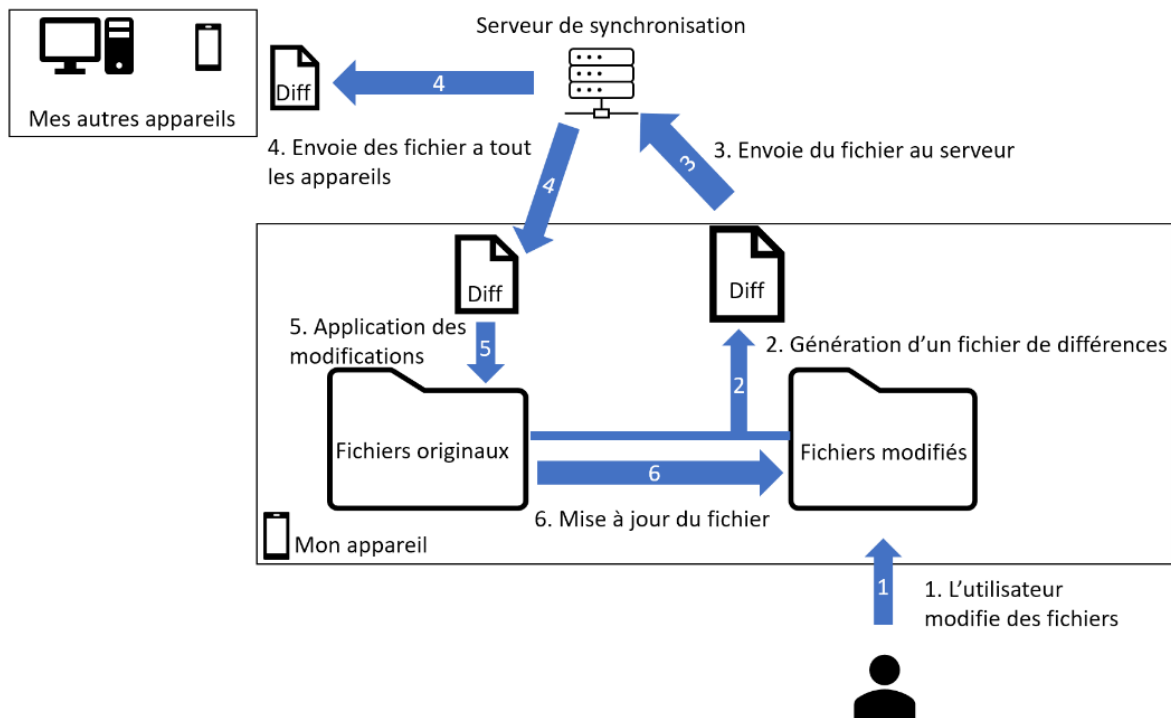


Schéma explicatif de la synchronisation des fichiers entre les appareils

Thibault Henrion

Back-office de l'application

Comme dit précédemment, l'application est propulsée par Homesynck, elle a besoin de communiquer avec le serveur Elixir.

Afin de communiquer avec le serveur, nous utilisons le kit de développement Java que nous vous avons détaillé précédemment. Il peut être compris par un module de React Native sous Android plus communément appelé Android Native Modules.

Un module sous Android correspond à une fonctionnalité présente sur l'application, ceci permet d'enrichir le paquetage utilisé en JavaScript en créant une nouvelle instance de celui-ci. Nous avons décidé de développer quatre modules afin de satisfaire les exigences de l'application Getway.

- Un module d'inscription permet de valider son numéro de téléphone et d'enregistrer une inscription d'un utilisateur.
- Un module de connexion permet à un utilisateur de s'authentifier.
- Un module récupère le nom de l'utilisateur authentifié sur l'application
- Et un module de synchronisation qui n'est pas des moindres puisqu'il permet d'interagir avec le serveur grâce au kit de développement logiciel Java.

Dans ce dernier module, des méthodes sont disponibles pour un client React Native.

Notre application est capable d'ouvrir un répertoire, de commencer la synchronisation avec le serveur, d'appeler une fonction demandant une fonction réagissant lorsqu'une mise à jour est reçue depuis le serveur. Pour pouvoir synchroniser les données de l'application, une

méthode est disponible ainsi que pour créer/modifier ou supprimer un fichier. Pour finir, une méthode est disponible pour pouvoir récupérer les données du répertoire courant.

Comme énoncé plus haut ces modules ne seraient pas fonctionnels sans l'utilisation du kit de développement logiciel en Java. Pour pouvoir utiliser cette bibliothèque, il suffit d'ajouter une dépendance à l'application, puisque ce kit de développement logiciel est disponible sur un serveur central.

Daniel Aguiar

Front-office de l'application

Pour développer cette application mobile utilisant notre protocole de synchronisation, il fallait trouver un environnement permettant de communiquer avec du Java puisque c'est le langage de programmation que nous avons choisi pour le kit de développement logiciel. Après de nombreuses recherches, le Framework React Native exploitant ReactJS était le premier choix, puisqu'il met à disposition une API permettant de créer des modules en langage de programmation Java ou Objective-C.

Il permet d'exploiter facilement un même code pour le compiler sur IOS comme sur Android. De plus, grâce à l'exploitation de composants natifs comme la bibliothèque UI Kitten, utilisée pour l'application, le temps de développement est réduit.

Dans un premier temps, nous nous sommes concentrés sur la partie conception de l'application.

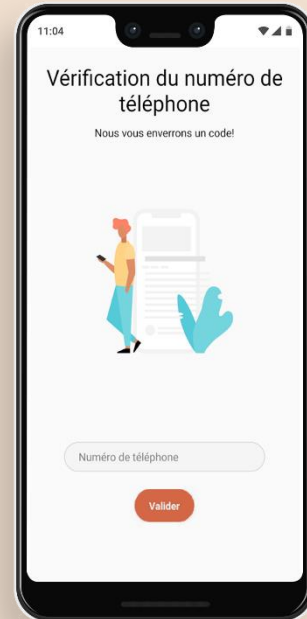
Vers quel type de design allions nous nous diriger ? Après de nombreuses discussions et modèles fait à partir de Figma, nous avons décidé de créer une charte graphique épurée et moderne que nous vous détaillons à la fin de ce rapport.

Par la suite, nous nous sommes documentés sur React Native, la documentation officielle nous a beaucoup aidée, de plus, React Native étant basé sur ReactJS, une bibliothèque JavaScript, il a fallu consolider nos bases de React.

Une application React Native est composée d'écrans. Chaque écran correspondant à une vue de l'application mobile. Un écran est composé d'un ou plusieurs composants qui communique entre eux grâce au système d'état de l'application, nous ne rentrerons pas ici dans les détails de son fonctionnement. Comme dit précédemment, l'utilisation de UI Kitten nous a permis de gagner un temps considérable dans le développement de l'application, puisque cette bibliothèque propose une collection de composants que nous avons pu réutiliser.

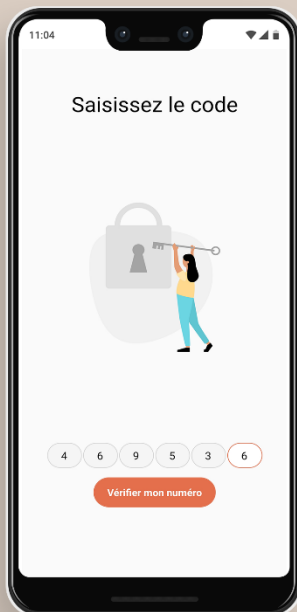
Jules Doumèche

Présentation rapide de l'interface et du fonctionnement de l'application pour l'utilisateur

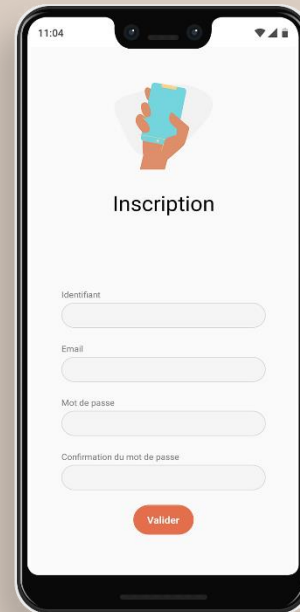


Lorsque l'utilisateur ouvre notre application, une connexion est requise pour accéder à son compte. S'il n'a encore jamais eu de compte, il peut s'inscrire en cliquant en bas de l'écran.

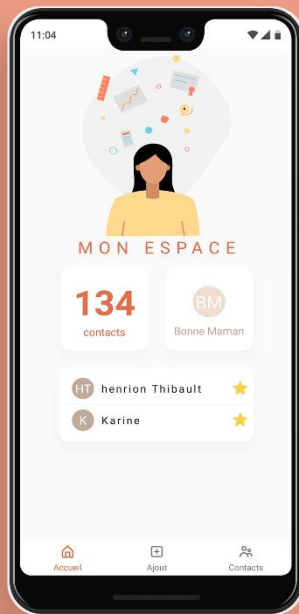
Dans le cas d'une nouvelle inscription, l'utilisateur doit renseigner son numéro de téléphone afin de créer un compte.



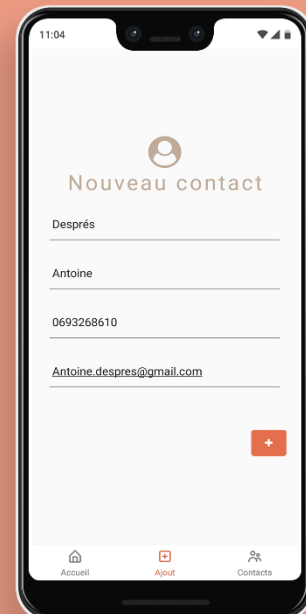
Un code de vérification est envoyé pour s'assurer de la validité du numéro de téléphone renseigné.



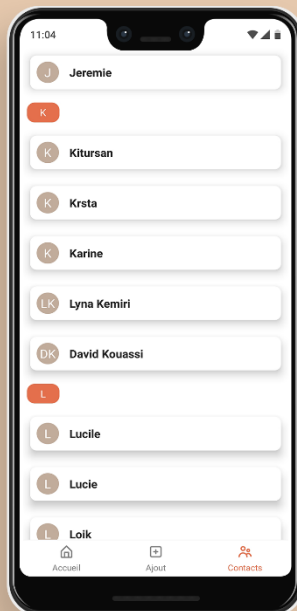
Après validation, le formulaire d'inscription est accessible pour enregistrer ses données.



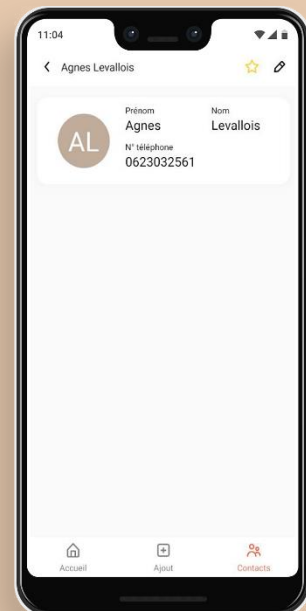
Lorsque l'utilisateur se connecte, ses contacts sont synchronisés préalablement avant que le page d'accueil s'affiche. Le nombre de contacts total, les contacts favoris ainsi qu'un contact choisi aléatoirement sont affichés dans des conteneurs.



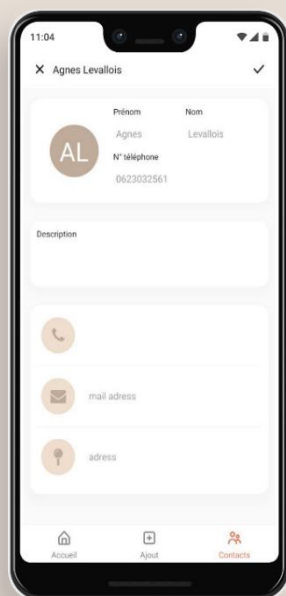
Il est possible d'ajouter rapidement un contact dans l'onglet 'Ajout'.



Il peut également consulter tous ses contacts dans l'onglet 'Contacts' qui sont rangés par ordre alphabétique.



Lorsqu'un utilisateur consulte un contact précis, il peut accéder à plusieurs détails comme son nom, prénom ou numéro de téléphone. Il peut ajouter ce contact en favori notamment en appuyant sur l'étoile.



Il peut également modifier les caractéristiques du contact, mais aussi ajouter plus de détails comme un autre numéro de téléphone, son adresse, son mail ou encore une description.

Hélène Te



Tout au long de notre projet, nous avons souhaité rendre accessible à tous notre code source, pour qu'il soit réutilisable par d'autres développeurs pour leur propre projet. Cette pratique, très courante dans le domaine de la programmation est appelée « open source ».

Pour rendre notre code « open source », nous avons choisi d'y appliquer une licence MIT qui permet aux développeurs qui utiliseront notre code de ne pas être limités dans la réutilisation (la vente d'un logiciel utilisant notre code est possible par exemple).

Pour partager le code, nous utilisons une plateforme très connue dans le milieu du développement : GitHub. Cette plateforme, rachetée par Microsoft en 2018, est un pilier pour l'open source, car elle permet d'héberger gratuitement son code source, à condition de le rendre disponible pour tous (il est aussi possible de le stocker en privé, mais des limitations s'appliquent).

Partager du code en open source a plusieurs avantages, en plus de permettre la réutilisation du code :

- Une meilleure transparence, et donc plus de confiance des utilisateurs dans l'application : avec les problématiques actuelles en termes de protection des données personnelles, les utilisateurs, et particulièrement le public visé par notre protocole, sont très sensibles à la transparence du code, qui permet à chacun de vérifier (ou de faire un article pour expliquer à ce qui ne sont pas développeurs) que le code ne contient pas d'éléments de collecte de données personnelles.
- Un code de meilleure qualité : cela prend du temps, et le code doit constamment être amélioré et mis à jour, mais mettre son code en open source permet à tous les développeurs qui le souhaitent de partager leurs modifications, et après vérification, de les ajouter dans des mises à jour (on dit que l'on « push » du code sur GitHub). Il est possible de signaler simplement les bugs, qui pourront ensuite être discutés entre développeurs pour réfléchir à la meilleure solution.
- La « sécurité du code », autrement dit les failles de sécurité qui seraient présentes dans le code : L'amélioration de la qualité du code allant, la plupart du temps, de pair avec la sécurité, cela améliore également ce point.

Nous avons séparé ce code en plusieurs *repositories* (ce mot est couramment utilisé dans le domaine de la programmation pour remplacer les mots « dossier » ou « répertoire de fichiers »), pour compartimenter notre code et permettre une meilleure compréhension par les futurs usagers.

Pour que le protocole Homesynck puisse être réutilisé facilement dans des programmes, nous avons mis à disposition des kits de développement permettant d'utiliser de façon simple notre protocole.

Un kit de développement, en programmation, est un ensemble de fonctions, mises à disposition par des développeurs pour qu'un autre développeur puisse intégrer ces fonctions dans son code sans les reprogrammer. Le développeur qui veut utiliser une fonction d'une bibliothèque pourra, simplement en lisant la documentation, comprendre ce que fait la fonction et s'en servir, sans avoir besoin de comprendre le code « à l'intérieur ».

Un premier kit de développement (dit SDK, pour Software Development Kit) pour Java :

Nous avons déjà détaillé son fonctionnement dans ce rapport, les deux paragraphes qui suivent sont des rappels pour appuyer sur la simplicité avec laquelle un développeur peut réutiliser notre code.

La synchronisation des fichiers est gérée sans que le développeur doive s'en soucier, ce qui permet d'avoir une interface très simple, avec seulement un appel à une fonction pour envoyer une commande. De même pour la réception, un appel à une fonction suffit pour démarrer l'écoute et recevoir les informations envoyées par le serveur.

Il est aussi possible de programmer des routines, qui sont des tâches effectuées automatiquement lors d'un évènement (de nombreuses plateformes font des sauvegardes la nuit grâce à des routines). Dans notre cas, cela permet d'actualiser l'affichage pour l'utilisateur du service lors de la réception d'une mise à jour d'un fichier depuis le serveur.

Pour rendre ce kit de développement facilement accessible aux développeurs, nous avons choisi de le publier sur Maven, une référence dans le monde du langage Java. Maven est un outil qui permet de publier sous forme de « plugin » son code, et les développeurs n'ont qu'à insérer une ligne dans un fichier de configuration de leur projet pour que l'import soit effectué automatiquement, en utilisant un autre outil appelé Gradle.

Un second pour JavaScript :

Ce second SDK propose lui aussi une interface très simple, mais ne possède pas autant de fonctionnalités, voici les fonctionnalités principales :

- La connexion à un compte
- La connexion à un dossier pour la synchronisation
- La création/modification/suppression de fichiers dans un répertoire donné

Il est à destination des développeurs web qui utilisent des technologies basées sur le langage JavaScript, en front-office avec React, Angular, etc ... et en back-office avec Node.js ou équivalent.

Pour utiliser ces kits de développement, il est nécessaire de posséder un serveur pour pouvoir faire les échanges de données. Pour permettre à chacun d'installer facilement le serveur élixir nécessaire au bon fonctionnement du protocole, nous avons choisi d'utiliser Docker et sa technologie de « conteneurs », la plus fiable aujourd'hui pour un déploiement de ce type.

Qu'est-ce que Docker et les conteneurs : Docker est un logiciel permettant de créer et de déployer (mettre en ligne) des ensembles de logiciels, sans avoir besoin de les paramétrer à chaque installation. Cela ressemble à la restauration des paramètres automatiques que l'on peut trouver sur certains smartphones. Cette technologie est différente de la « virtualisation », car ici l'affichage n'est pas géré, il y a uniquement les processus qui permettent de faire fonctionner les logiciels.

Dans notre conteneur Docker, il y a donc la base de données, qui fonctionne avec le Système de Gestion de Bases de Données (SGBD) PostgreSQL, la bibliothèque qui nous permet de communiquer entre le serveur, le microservice de validation des numéros de téléphone, et le « reverse-proxy » que nous avons évoqué dans la partie du protocole.

Une personne voulant utiliser notre protocole peut donc simplement installer le conteneur Docker sur un serveur ou sur son ordinateur, et pourra créer son propre environnement de développement simplement.

Nous avons aussi créé un client permettant d'interagir avec le protocole, il permet d'effectuer les opérations disponibles sur le SDK JavaScript à partir de lignes de commandes. Ce type de programme est très prisé des utilisateurs du système d'exploitation Linux, ce système étant principalement utilisé par le biais de commandes.

Nous avons utilisé une bibliothèque pour embellir l'affichage, et faciliter l'utilisation.

```
=====
| HOMESYNCK CLI |
=====

? Main menu » - Use arrow-keys. Return to submit.
  Connect & Sign in
  Open directory
  Create directory
  Show messages
  Push message
> Exit
```

Captures d'écrans du client

Gwénolé Martin, Hélène Te, Jules Vanin

Outils et techniques

Ci-dessous une liste non exhaustive des principaux outils et techniques que nous avons utilisés pendant ce projet.

Langages et bibliothèques

Elixir OTP : Langage de programmation côté serveur, qui permet de gérer les flux de données concurrents. Il est basé sur la machine virtuelle BEAM, qui est très utilisée dans le secteur des télécommunications et par des applications avec un grand nombre de communications à gérer, notamment WhatsApp et Discord. Elixir est utilisé pour le serveur. Nous l'avons choisi car il nous permet d'avoir une gestion du parallélisme optimale pour le protocole de synchronisation.

- **Ecto** : Bibliothèque pour la communication avec la base de données, sécurisée. Il s'agit de la bibliothèque la plus utilisée pour ce type de tâche en Elixir.
- **Argon2** : Bibliothèque permettant de hasher de mots de passes avec l'algorithme de génération de clé sécurisée Argon 2. Argon 2 a gagné un concours de cryptographie majeur en 2015, et nous semble approprié pour la sécurité des dossiers dans notre projet.
- **HTTPOison** : Bibliothèque permettant d'utiliser un client HTTP, avec toutes les options nécessaires pour la sécurité, elle nous permet notamment de communiquer avec le serveur de validation des numéros de téléphone.

HTML5 et CSS3 : Langages de programmation très utilisés pour créer des sites web, nous les utilisons pour la page de présentation de notre projet.

Java : Langage de programmation très utilisé, bien documenté, et qui nous permet de développer à la fois l'application Getway et le kit de développement pour Java.

- **Java Diff's Util** : Bibliothèque qui permet d'utiliser des fonctions de comparaison entre des textes, essentielle pour le protocole de synchronisation.
- **Java-phoenix-channel** : Phoenix et Java communiquent avec des WebSockets, et cette bibliothèque permet d'adapter côté client (Java donc) les informations à transmettre.
- **OpenJSON** : Bibliothèque permettant d'analyser et de traiter les données JSON.

JavaScript : Langage de programmation utilisé côté client et côté serveur. Dans le projet, il est utilisé pour le serveur de validation des numéros de téléphone et pour le kit de développement pour JavaScript.

- **React** : Bibliothèque JavaScript permettant de créer des interfaces utilisateurs. Nous l'utilisons dans l'application mobile Getway.
- **Phoenix channel** : Fonctionne comme la bibliothèque Java du même nom, permet d'adapter les informations pour le SDK JavaScript.

Environnements et outils de développement

Insomnia : Client qui permet d'envoyer des requêtes vers des API ou des pages web pour tester leurs réponses. Cela nous a servi pour le serveur de validation des numéros de téléphone.

Android Studio et **Android Debug Bridge (ADB)** : Android Studio est un environnement complet de développement qui permet de développer des applications mobiles fonctionnant sous Android. Android Debug Bridge est l'émulateur intégré, il permet de créer un téléphone virtuel sur notre machine. Nous avons utilisé cette solution pour développer Getway, notre application mobile.

GNU Privacy Guard ou gpg : Outil dont la fonction principale est le chiffrement. Il est nécessaire ici pour SonaType (pour signer les envois).

GitHub : Une plateforme permettant d'héberger et de gérer des fichiers, qui est très utilisée dans le développement logiciel car elle permet de coder à plusieurs, avec un système de versions de fichiers efficace. Nous avons utilisé cette plateforme quotidiennement pour le développement de notre projet.

GitHub Pages : Un service permettant de générer des sites statiques pour la documentation.

Visual Studio Code : Un éditeur de code, riches en fonctionnalités pour le développement, qui était particulièrement adapté pour le langage Elixir OTP.

IntelliJ IDEA : Un éditeur de code, à l'image de Visual Studio Code, mais qui propose de nombreuses intégrations, et qui est particulièrement adapté pour développer du Java.

Code With Me : Outil intégré à IntelliJ permettant de développer en temps réel en groupe.

Docker : Permet de déployer un programme sur n'importe quelle plateforme, grâce à une virtualisation de Linux. Cela nous permet de déployer le serveur en quelques minutes en cas de problème (très utile quand l'hébergement est chez OVH).

Docker-compose : Gestionnaire de conteneurs Docker qui permet à la base de données et au serveur de communiquer et d'être installés ensemble.

Npm : Gestionnaire de paquets officiels de node.js, il permet d'installer les bibliothèques souhaitées pour une application. Nous nous en servons dans toute la partie JavaScript de notre projet. Nous avons aussi déployé notre propre paquet sur npm.

Nginx et Certbot : La combinaison de ces deux outils permet de renouveler automatiquement les certificats HTTPS sans devoir modifier le code ou faire des opérations sur le serveur. Nginx permet d'installer un reverse-proxy et Certbot de générer des certificats.

Gradle : Outil permettant d'importer des dépendances. Il nous a permis d'importer la dépendance que nous avons créée grâce à Maven.

Frameworks et autres

Express (Node.js) : Framework JavaScript que nous utilisons pour le serveur de validation des numéros de téléphone.

Phoenix : Framework web pour développer sous Elixir, qui nous a permis de réduire considérablement le temps nécessaire au développement du serveur. Nous avons essayé de nous en passer au début du projet, car un framework ajoute toujours une certaine « lourdeur » au développement, mais cela n'a pas été concluant.

React Native : Framework dédié aux applications mobiles permettant au développeur d'utiliser React avec les fonctionnalités natives des systèmes.

Canva : Site internet qui propose des maquettes pour les sites web, nous l'avons utilisé pour le site de présentation.

WebSocket : Les WebSockets permettent d'envoyer et de recevoir des requêtes HTTP en restant sur l'interface de ce protocole de communication, ce qui est nécessaire pour communiquer de façon sécurisée. Nous les utilisons pour la communication avec le serveur.

VPS : Un VPS est un serveur hébergé dans des datacenters en général, c'est ce que nous utilisons pour héberger le serveur et le site de présentation.

Heroku : Permet de déployer facilement des applications sur des serveurs. Nous nous en servons pour le serveur de validation des numéros de téléphone.

UI Kitten : Composant permettant de créer des applications React plus rapides et agréables.

SonaType et Nexus : SonaType est un outil qui permet de mettre notre bibliothèque Java en ligne pour pouvoir l'importer avec Maven. Pour fonctionner, SonaType utilise Nexus, un gestionnaire de fichiers spécialisé pour la programmation et il est nécessaire de signer les envois avec gpg.

Daniel Aguiar & Anicet Nougaret

Organisation

Pour organiser notre travail, nous avons utilisé principalement Git, pour coder ensemble sur les différents composants de notre application, et également le « code à plusieurs en temps réel », avec les outils intégrés sur Visual Studio Code et IntelliJ.

Pour la répartition des tâches, nous avons divisé en 3 grandes parties :

- le front office, avec Jules Doumèche, Hélène et Gwénolé (dans une moindre mesure)
- le back office, avec Daniel et Thibault
- le serveur, avec Anicet, Jules Vanin et Gwénolé

La gestion du projet a été assurée par Anicet et Gwénolé, même si les membres du groupe se sont auto gérés la plupart du temps, notamment grâce à la répartition en sous-équipes.

Tâche \ Semaine	Démarrage du projet	Semaine 1	Semaine 2	Semaine 3	Semaine 4	Semaine 5	Semaine 6	Semaine 7	Semaine 8	Semaine 9
Conception du projet et mise en place des plateformes de partage	Création du Github / Discord									
	Choix des environnements et technologies									
	Présentation powerpoint									
Mise en place d'un serveur Elixir et d'une base de données	Documentation Elixir									
	Documentation PostgreSQL									
	Essais avec PostgreSQL									
Écriture et préparation du rapport	Faire le plan du rapport									
	Écriture du rapport									
	Design du poster									
Conception interface utilisateur	Charte graphique									
	Maquettage sous Figma									
	Prototypage de l'expérience utilisateur									
Développement de l'application sur Android Studio	Utilisation de l'API Contacts									
	Utilisation de la bibliothèque (SDK) Java									
Conception de l'application mobile	Développement sous React Native									
	Développement du système d'authentification									
	Gestion des données									
Développement d'une bibliothèque Java	Conceptualisation de l'API									
	Programmation Java									
	Écriture de la documentation									
Implémentation d'un SDK Java	Connexion à la session									
	Communication avec les Web Sockets									
	Synchronisation des fichiers									
Divers	Création de logos									
	Compte rendu									

Gantt de la répartition des étapes de notre projet dans le temps

Hélène Te

Conclusion

Pour conclure, nous sommes satisfaits du résultat que nous avons obtenu à l'issue de ce projet. La charge de travail que nous nous sommes fixés était conséquente, et il n'était pas sûr que nous puissions faire fonctionner tous nos composants ensemble, malgré les précautions que nous avons prises.

Ce projet a été très enrichissant pour l'ensemble des membres du groupe, nous avons utilisé des technologies que nous n'avons pas vues en cours, et cela nous a appris à séparer de façon distincte les tâches, ce qui est un peu plus difficile dans les autres projets que nous avons réalisés tout au long de notre DUT.

Daniel Aguiar

Difficultés rencontrées

Même si nous avons réussi à atteindre nos objectifs, nous avons rencontré de nombreuses difficultés, notamment :

- Avec React Native : React Native nous a posé beaucoup de problèmes, avec une installation compliquée, et des dépendances qui demandaient constamment des mises à jour. Tant que les mises à jour n'étaient pas installées (et il y a souvent de nombreux bugs à l'installation), il n'était plus possible de lancer l'application.
- Pour la compréhension de Gradle : Cet outil est assez compliqué à comprendre, pour l'importation des bibliothèques et la configuration d'un projet. Gradle n'avait pas été vu pendant notre formation, à l'inverse de Maven que nous avons utilisé en Java.
- Pour conceptualiser le fonctionnement de la bibliothèque et la communication asynchrone avec le serveur : il était difficile de comprendre comment il était possible de communiquer avec une WebSocket et d'utiliser l'API Java 8 pour l'asynchrone, qui sont des éléments nous n'avons jamais vu (nous étudions Java 7 à l'IUT) et qui sont difficiles à appréhender.
- Avec Android Studio : ce programme très lourd pour un ordinateur, même assez puissant, est obligatoire pour faire une émulation d'un téléphone Android, et son utilisation très importante des ressources de nos ordinateurs a considérablement ralenti le développement (il fallait parfois jusqu'à 10 minutes pour réussir à lancer un test de l'application, si l'on multiplie par le nombre de bugs à résoudre lors du développement, cela devient très long).

Gwénolé Martin

Possibilités d'évolution

Il y a de nombreuses possibilités d'évolution pour ce projet, notamment à travers la réutilisation du protocole pour d'autres applications.

L'application de contacts n'utilise pas au maximum le potentiel du protocole, car il y a peu de conflits lors de la gestion des ajouts et suppressions de contacts. Il serait envisageable de faire des applications se rapprochant du temps réel (il y a une latence d'environ une à cinq secondes entre chaque mise à jour). On peut imaginer par exemple un clone de Google Docs, ou même de Netflix (des soucis de mises à l'échelle se poseraient sans doute, et il faudrait augmenter la taille de notre serveur).

Le nombre de machines qui peuvent se synchroniser sur un même dossier est lui aussi théoriquement illimité, cela pourrait servir pour mettre à jour dans un dossier les informations issues de nombreuses sources.

Pour l'application mobile Getway, il est possible d'envisager de nombreux ajouts, comme de nouveaux champs pour les contacts, une version web, une version iOS (que nous n'avons pas faite car le développement sous iOS est extrêmement coûteux).

Ces pistes ne sont qu'une infime partie de ce qu'il est possible de réaliser à partir de ce projet. Le projet étant complètement open source, il est désormais possible qu'il soit utilisé, si quelqu'un pense qu'il serait intéressant de l'insérer dans un programme, pour des usages auxquels nous ne penserions sans doute jamais.

Thibault Henrion

Bibliographie

Les ressources que nous avons utilisées sont principalement des sites contenant des articles ou de la documentation pour le développement

<https://medium.com/> - Medium

<https://reactnative.dev/> - React Native

<https://www.baeldung.com/> - Baeldung

<https://maven.apache.org/> - Maven

<https://gradle.org/> - Gradle

<https://kuon.github.io/java-phoenix-channel/index.html> - Bibliothèque phoenix channel

<https://java-diff-utils.github.io/java-diff-utils/> - Bibliothèque Java Diff Utils

<https://help.sonatype.com/docs> - Sonatype Help

<https://stackoverflow.com/> - Stackoverflow

<https://git-scm.com/doc> - Git - Documentation

<https://developer.mozilla.org/en-US/docs/Web/JavaScript> - Documentation officielle de JavaScript

<https://www.manning.com/books/the-little-elixir-and-otp-guidebook> : Livre sur Elixir « The little Elixir & OTP Guidebook » par Benjamin Tan Wei Hao, 2016

<https://www.manning.com/books/elixir-in-action-second-edition> : Livre sur Elixir « Elixir in action, Second Edition » par Saša Jurić, 2019

Thibault Henrion

Glossaire

Asynchrone : Un processus dit « asynchrone » n’attend pas qu’une instruction se termine avant d’exécuter l’instruction suivante.

Docker : Un logiciel permettant de virtualiser des ensembles de logiciels et de paramètres dans un conteneur. Il permet de déployer les images créées en un seul clic.

GIT : Un gestionnaire de versions en ligne très utilisé.

Kit de développement : Software Development Kit (SDK) en anglais, c’est une ou plusieurs bibliothèques permettant d’utiliser des fonctions grâce à une référence externe.

Licence MIT : Une licence libre permettant de partager sans restriction ses projets.

Maven : Un outil d’automatisation permettant de configurer un projet Java.

NAS : Un serveur de fichiers, qui sert principalement à stocker ces fichiers.

Programme propriétaire : Un programme dont le code source n’est pas libre.

WebSocket : Un canal de communication à double sens (multiplexé).

Gwénolé Martin & Jules Vanin

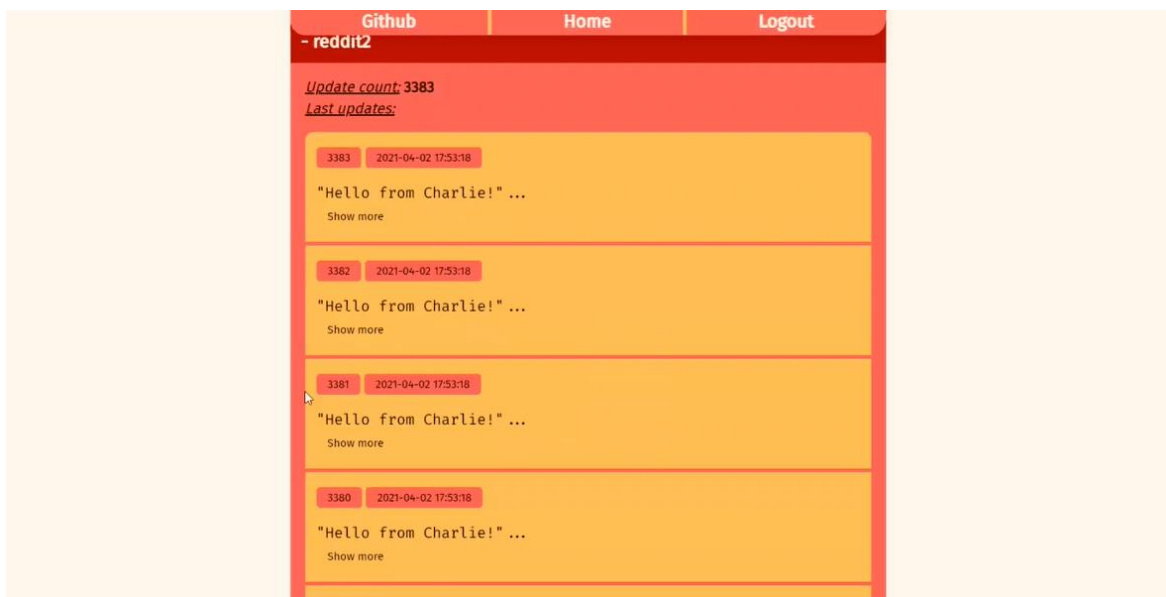


Lien vers notre site web : homesynck.anicetnougaret.fr


Lien vers notre GitHub : github.com/Homesynck



Exemple d'un test de charge réalisé avec le SDK JavaScript









Le dashboard avec les opérations de mises à jour en direct



Homesynck

Message synchronization made easy!

 IUT de Paris - France
  <https://homesynck.anicetnougaret.fr/>

 Repositories
  Packages
  People
  Projects

Type

Language

Sort

homesynck-server







Websocket server for Homesynck, a message synchronization system. Based on Phoenix Channels.

backend

synchronisation






realtime-messaging

message-broker

 Elixir
  MIT
  0
  3
  0
  0
 Updated 22 hours ago

homesynck-ts-js-sdk

ES6 SDK for Homesynck

 JavaScript
  0
  1
  0
  0
 Updated 22 hours ago






Getway

Mobile contact management app built on top of Homesynck

react-native

android-application

ui-kitten

 JavaScript
  0
  3
  0
  0
 Updated 2 days ago

homesynck-java-sdk

Java 8 SDK for Homesynck







communication

maven

back-office






asynchronous-programming

sdk-java




 Java
  MIT
  0
  3
  0
  0
 Updated 2 days ago

homesynck-cli

A simple CLI to use your Homesynck server

 JavaScript
  0
  0
  0
  0
 Updated 3 days ago


Top languages

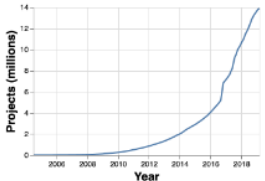
 JavaScript
  Java
  Elixir

People

This organization has no public members. You must be a member to see who's a part of this organization.

Structuration du répertoire de développement sur GitHub




Indexed Artifacts (20.2M)


Popular Categories

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing
- Code Analyzers
- Collections
- Configuration Libraries
- Core Utilities
- Date and Time Utilities

Home » [com.github.homesynck](#) » homesynck-java-sdk



Homesynck Java SDK

A cross platform sdk for homesynck applications

License	MIT
Tags	github sdk

Central (8)

	Version	Repository	Usages	Date
1.0.x	1.0.3	Central	0	Apr, 2021
	1.0.2	Central	0	Apr, 2021
	1.0.1	Central	0	Apr, 2021
	1.0.0	Central	0	Apr, 2021
0.2.x	0.2.2	Central	0	Mar, 2021
	0.2.0	Central	0	Mar, 2021
0.1.x	0.1.1	Central	0	Mar, 2021
	0.1	Central	0	Mar, 2021

Site web de Maven Repository où l'on peut retrouver le SDK Java

[Github](#)
[Sign in](#)



Synchronize your frontends in real-time!

.....

- Simple & organised**

Clients connect to accounts and open directories. Directories receive ordered updates from clients in real-time. Clients receive updates on directories they opened. Clients process directories' updates in order and cache them.
- Highly reliable**

Built on top of Elixir and the Phoenix framework, it can handle concurrent streams of data like no other tech.
- Data agnostic**

It doesn't care about the kind of data you want to synchronize, its only job is to store it and avoid disorder.
- A foundation to build upon**

Homesynck is distributed as an open source product, meaning anyone can come and tweak it as they please. It offers basic customisation to manage accounts, security and even add specific features like phone validation.
- The only backend you need?**

Homesynck can act as the perfect backend for simple apps. That's what we tried with Getway, an Android contact management app. Homesynck acts as its only backend, managing data, persistence and authentication layers. • It's all real-time!

Site web de présentation du protocole Homesynck



Résumés en anglais

Firstly, we wondered how we can synchronize our contacts through an application and manage them. Based on knowledges of each person in our group we decide to make an entire system based on a haul powerful system. To describe the system that we set up, I will split the system in three parts, the server, back-office and to finish the front-office which is an android application. We decide to make an application which synchronizes all your contacts and manage them. This is a proof of concept of our file synchronization protocol.

This project was really exciting we were able to discover several technologies that have not been discussed. It was a good challenge for all of us build a whole eco-system. About my knowledges on software development, I was able to work on several sides. As far I am concerned, I was able to work on the Getway application, doing the back-office work. Since the application was developed on top of Homesynck I have to setup a bridge between the front-office and the back-office which is a software development kit that I will tell you about later. We decided to work with React Native a JavaScript framework which has a bridge to communicate with Java. On another side, I worked with Thibault Henrion, we were in charge of doing a software development kit in java to make communication between the server and the client. During the development of this API, we had a lot of struggles finding how we could set up a communication with the server and how to build the synchronization system.

Furthermore, we thank all this person who builds libraries to improve architecture systems, it was worth it for us, we used a library to communicate with the server and a library to check the differences between files. We thought it was over and the system will work, but that was a dream because we had problems with asynchronous programming in Java, thankfully Java gives a bunch of APIs to facilitate the development.

Otherwise, when this software development kit was advanced and reliable, we decided to publish this library to a public repository for improving the use of the library in Android. With my teammates, we used agile development it helps us to be coordinated, we cannot pass over this part. We used Git a useful tool because we want to keep it safe of fires.

Finally, this project allows me to improve knowledges in software development, teamwork spirit, my ability to solve problems. Indeed, it was the first time I bring a big project with six teammates. I had an extremely good experience, and I do not regret it. In this situation we had to deal with some issues, like the given time slot was a restriction for us, work at home means we are not able to find quickly if there is a problem. Fortunately, the school gives us three half days together per weeks in a room. This project was very ambitious, all of us had to work on overtime, I spent most of my time coding, but I loved it, and it is for that I am sure I want to continue in software development. We are proud to have succeed our project and make it available for anyone. The last words I can say is that we put each of our strengths to achieve something great.

Daniel AGUIAR



Our pjs4 first started with the idea of a synchronized contact application with local storage on the phone. We were quite ambitious and wanted to make a project that would make us proud, and tackle new technologies to challenge us. However, our idea evolved and was divided into two distinct parts: a first one "Getway" for the contact application, and a second one "Homesynck" for the synchronization protocol that can be used for many different projects and not limited to our case.

We then started to think about the overall architecture, identifying the tasks to be done and the complexity of each one. When we had a clear idea in mind, we were able to divide

the various tasks by preferences. I decided to work on the mobile application because I was very interested in the React Native framework. I was responsible for prototyping the application, reflecting on the visual identity and design, and programming all the logic of the application with my teammates H       and Daniel.

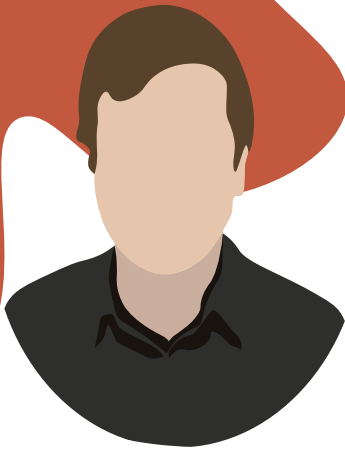
The beginnings seemed to me very long, because I had to appropriate a new technology and read a lot of documentation on the internet and follow hours of tutorials. The application was not making much progress and the results obtained were not necessarily those expected. In order to optimize the time, we realized in the same time the logic of the application, and the graphics of the application. It allowed us to advance on the functionalities of the application without worrying about the visual aspect. We could then gradually integrate all the design of the application.

To be able to collaborate efficiently on the same code, we used the git tool with the gitflow workflow. We split the code of the application in branches that each corresponded to a feature of the application (example: login feature, add-contact feature) and then we merged them into a single 'develop' branch at the end of each feature.

The last few weeks were a lot more smooth as I was more at ease with the use of React Native and everything went more quickly. The last difficulties were found towards the end, when we had to link all the parts of the project together and consequently adapt some parts of the code. But when we had a working product we were all very happy and excited because all those hours of work had finally paid off. Certainly some of the features that we had planned could not be implemented due to lack of time, however with all the difficulties encountered I was very proud to have a working and fully functional system.

In conclusion, this project was far from easy but we all managed to overcome the difficulties, and it was a lot of pleasure to work with this group because everyone was very friendly, supportive and helpful. I learned a lot on a technical level with the use of new technologies like React Native, but this project also taught me how to manage a project and work in a group. I noticed that letting everyone express themselves, always considering the other's voice as a potentially great idea, keeps everyone motivated and involved.

Jules DOUMECHE



At the start of our project, we decided to create an application for managing synchronized contacts between the user's different devices. We therefore decided to split the realization of the project in two parts. A first part which manages contacts in the form of files and a second part which synchronizes these files online. We therefore divided the tasks into three, a team took care of the contacts application, a part took care of the synchronization on the user's device and a last part took care of the synchronization server. I personally took care of the synchronization of files one the user's device and the login, the register, and the connection to the directory of the server parts.

To create this program, we first need to choose a programming language. We decide to take Java because we know this language well because we have used it for many projects, and it is well documented if we encounter a problem. It is also a cross-platform language as we can use it on Windows, Linux, Android, and Apple devices.

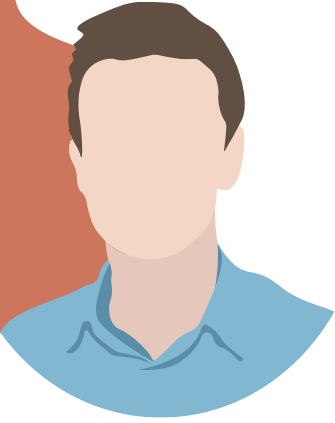
The first difficulty we encountered was to make Java on the client device and Elixir, the programming language of the server, to communicate. To overcome this problem, we found a library that make this communication. It was still a problem because this library was coded in Kotlin, fortunately, Kotlin and java are very similar, and they can interact easily with each other. I first create a code where we can connect or register to the server in java.

After that, I started to imagine a create a class that manage the file synchronization. To avoid generating excessive uploads to the server each time files are modified, we have found a file creation library that allows you to apply modifications to a file to obtain a new file. Then, only the added parts are sent.

Another difficulty that we have overcome is working remotely. Working in parallel on creating a server and creating the client that communicate requires defining the operation from the start and having a clear idea of the capabilities of the different languages. We were able to achieve these two parts simultaneously through working diagrams and try our different ideas by performing feasibility tests with different technologies.

Finally, this project allowed me to create a complex and functional application as a team. We had to respect conventions to create a professional application and we spent a lot of time writing English documentation on our tools to make its use accessible for everyone. This project also shows me that I liked programming and allowed me to confirm my choices to continue my studies in engineering schools and specializing me later in software genius.

Thibault HENRION



For this project, we wanted to create a mobile application allowing synchronizing contacts between several devices, with eventually a web application for compatibility on all supports, a bit like what is done in the field of password managers. Very quickly, we became interested in creating a synchronization protocol, which became a major part of our project. We decided to call our protocol Homesynck and our application Getway, which we developed as two separate products, the application, however, depending on the protocol.

During this project, I worked on multiple tasks. After the first design phase, during which I sometimes debated for several hours with other group members to think about the most suitable technical solutions. I developed the phone number validation microservice in node.js, and helped when needed for the server and front-end part (especially for the navigation).

Also, I managed the group with Anicet, to organize the tasks and prioritize the elements to be done.

For the development part, the SMS microservice was not so easy to realize, I first tried to use AWS, to be able to send SMS, but our student accounts did not allow us to do it (we have an amount to be able to send SMS, but it is not possible to connect simply as we are not administrators of our accounts). So I had to use the Nexmo API, and quickly program an API for the server to make it calls when creating an account. This part of the code was done alone, so I didn't need to use GitHub for the development of this microservice.

For the rest of the development, we used, in addition to GitHub, the simultaneous code features offered by Visual Studio Code and IntelliJ, which allowed us to be more efficient.

Finally, I managed to achieve what I wanted on this project, especially in terms of project management, even if the group was working very autonomously, it was necessary to prioritize the tasks, justifying them so that everyone understands the usefulness of their work. With this project, I was able to reinforce my programming and project management skills.

This project reinforced my initial idea for my future studies: to attend an engineering school that is as general as possible, so that I won't be doing programming during my professional career. My skills in team and project management are a priority for me, and I want to focus my career on managing large groups in industry, finance or technology sectors. Seeing a project succeed is always very motivating, and we are very satisfied with what we managed to produce for this PJS4

To conclude, this project was very pleasant, especially since there was a very good understanding within the group.

Gwénolé MARTIN



This project was very beneficial to my computer science knowledge and skills. It helped me grow more than any other project I did during my prior 4 years of experience as a programmer.

My main roles in this project were to design our synchronization protocol, to work on the Homesynck server, the Homesynck web dashboard, the Homesynck JavaScript SDK and to continuously deploy all the pieces to the cloud.

By definition, a protocol implies many independent parts agreeing with each other. That is why designing our synchronization system required good communication skills across the whole team. Thibault and Daniel, my main partners on the other side of the protocol, were a pleasure to work with. We managed to stay clear, honest and efficient despite the rush. Although we had some desperate moments where we believed it would never work, we managed to embrace an iterative approach that enabled us to implement the whole system as it is today.

I am also satisfied with the variety of topics I had a chance to work on. It required many different skills that I got to improve, various technologies I had to learn from scratch. In order to help us with that, I took the huge responsibility of finding the best open-source building blocks we could use throughout the whole project and making sure they integrate well within it. Those choices greatly increased the efficiency and the scope of the project.

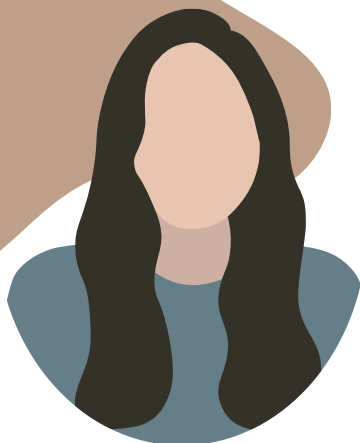
Indeed, I am usually a very curious programmer. I like to explore many techniques and technologies without trying to go too much in depth about them. Just so that I know what they could be useful for in my projects. And this mindset paid off this time, since I introduced the team to the unorthodox Elixir programming language. It is the language that we used in order to write the server's code, and one that I barely knew how to use before.

I am proud of having taken this risky decision. Committing to it enabled us to tackle complex topics like concurrent algorithms, data synchronization and code reliability easily. Elixir is such a powerful tool. To learn it, I had to change my way of thinking about software, which took me a lot of trial and error. But thanks to this effort we solved the problem in an easier, more elegant and efficient manner.

Another important part I played in was building the server deployment infrastructure. Coding and documenting a piece of software is great, but to me if you do not know how to deploy, run and distribute it properly then it stays unfinished. Deployment for instance is something most people find scary, but it was a skill I always wanted to learn. That is why I am glad to have continuously deployed our entire infrastructure, along with learning Docker and docker-compose which are industry standard tools for doing this job.

Everything I learned in this project made me improve a lot as a programmer. I now feel capable of solving more difficult problems than before, as well as bringing innovation to existing projects. I am also glad we turned this ambitious idea we had in a real Open-source product. I think it is a success and will certainly use it in my next projects.

Anicet NOUGARET



For our project, we decided to develop a synchronization protocol called Homesynck which allows to synchronize files from one device to another. In order to illustrate this technology, we created a mobile contact management application called Getway.

It mainly allows to synchronize contacts from one device to another thanks to the Homesynck protocol. The idea of this application is based on a need of a team member who could not find an application with this synchronization system.

In order to realize this project in the best conditions, we divided the tasks. Personally, I was in charge of the front-end development part of the Getway application with other teammates because I feel more comfortable with web development than back-end development. However, during the project, there was a major difficulty for me. Indeed, we used the framework React Native based on the programming language JavaScript for the development of the application. We didn't studied this framework in class, that's why we had to acquire the basics in a short time to be able to realize this project. The first weeks were complicated, because all the people working on the front-end had to learn this new language. There were times when learning was really difficult, but I could count on my friends to help me when I didn't understand. Moreover, one person in the group had already used this programming language, which allowed us to ask him for advice or help.

Futhermore, I had never done a programming project with so many people before. So, I was not used to using Github to share our lines of code. Moreover, at the beginning of the project, I was afraid to code and therefore create problems in the code. However, with time, I got a little more confident thanks to my classmates and I felt more comfortable coding. Finally, we managed to finish this very ambitious project, even if there are still some areas for improvement.

This project, which concludes my year at the IUT, was very enriching. I was able to participate in the development of an ambitious project that used many resources. As a result, I learned a new language and other technologies that will probably be useful for next year. I also strengthened my programming skills because even though I was programming in a new language, the logic remained the same.

In addition to these acquired skills, it brought me a lot personally. I gained a little more confidence in coding, because I was afraid to do it and ruin the work of others. Moreover, as an introvert, I sometimes had difficulty affirming myself within the group and this is a point that I noticed during the development of the project. I will be careful to overcome this shyness for the next time.

Concerning my studies, I knew in advance that I did not want to continue in web development and my choice did not change during the project. However, it was a very nice project, and I am very happy to have done it with this group because I learned a lot during the project.

Hélène TE



Our project started when we first thought of making an application which purpose was to manage our phone contacts. The goal was to create an open-source application which could manage all our contacts, even if they were merely people registered with their email. We needed an application which could access our contacts, make a copy of them, and then let the user modify them or start adding or deleting contacts from this copy. That way, our application cannot mess with someone's contacts directly since it only uses a replica. But almost halfway through, we decided to split the project in two parts.

And so, we made one application called Getway that could manage our contacts, and a service named HomeSynck which goal was to synchronize in real time modifications users make.

I mostly worked on the server-side of our application. I started working on making a proper database for our project that we chose based on its compatibility with our server. This means that I had to know every type of data we needed from users to make our application work, how to manage it in our database, but also how we can prevent privacy issues. Throughout this project, I needed to always make sure that I never broke any privacy policies. Then, I had to abruptly change what I was working on since we chose another way to implement our database. I started learning how we could manage the database with a language I never used before. This was definitely one of the most complicated aspect of this project for me. Indeed, I almost only programmed in unknown environments (using new frameworks, or technologies I never tried to learn before) and I only had the basics I learned in school to help me. The way I solved this issue was to document myself a lot on the technologies I was working with before trying to use them properly. Then, it was merely an issue of trying to understand what I was working on, which was solved by sometimes asking help from other members of the group or documenting myself a little more.

After working on the server-side of our application, I had to find a way to retrieve the list of every contacts a user has. This part was easier to work on, since it only used technologies I learned in school, except for the Java library I had to use. This was the perfect chance for me to showcase the improvements I made in searching useful informations and finding the best ways to add it to our project. This part did not last very long, but it somehow helped me solidify what I already studied in school and made me a bit more confident in my ability to produce useful and, hopefully, well written code.

In conclusion, I believe this project helped me figure out many things I did not know about the computer-science world. I learned the steps of creating an open-source project and I also improved my searching methods, my teamwork and communication, and finally, my coding capabilities. But I think more importantly, this project was for me a fabulous way of working with talented and passionate people, share ideas about how to handle things, and gain experience in group-based projects.

Jules VANIN

Poster du projet



Notre poster met en avant notre application mobile Getway qui est l'aspect le plus visuel de notre projet, et nous indiquons les mots clés du projet. Pour représenter au mieux notre application, nous avons utilisé principalement la même palette de couleurs et les caractéristiques pour la conception du poster.

MISE EN FORME

Au centre du poster, nous avons représenté un téléphone avec pour fond, la page d'accueil de notre application qui permet de résumer les fonctionnalités de celle-ci (l'ajout de contacts, la consultation de la liste des contacts).

Sur les côtés, nous retrouvons, comme sur la page d'accueil de l'application, des conteneurs de différentes tailles regroupant les outils utilisés, mais également des mots clés pour décrire notre projet.

Enfin, pour rester dans le thème épuré et simple de notre application, nous avons créé nos propres personnages unidimensionnels afin de représenter chaque membre du groupe et notre enseignant qui nous a encadré tout au long du projet.

CHARTE GRAPHIQUE



Le blanc représente la pureté et est facilement associable avec d'autres couleurs.



Ces deux teintes de marrons représentent la douceur et la neutralité. Cela correspond tout à fait au côté moderne et simple de notre application.



L'orange quant à lui, permet d'ajouter à cette palette une pointe de dynamisme qui contraste justement ces couleurs douces et pures.

Nous avons également joué sur l'opacité des couleurs afin d'obtenir une palette plus riche.

Pour la police, nous avons utilisé **Source sans variable** qui correspond au thème de notre poster. Une police simple et fine pour accentuer le côté épuré de notre application.

Hélène Te