2021

Rendu de groupe Ac'LAB

PROJET DNE – M1S1

JULIEN DUDEK - PIERRE DARCAS - MORGAN LOMBARD - GIANNI GIUDICE
TRISTAN COUSSAERT - GAUTIER COUTURE

Le projet

Le projet DNE, pour Dossier Numérique Étudiant, consiste en un POC d'une plateforme dont le but est de centraliser les informations qui concernent l'étudiant, sa communication et son suivi scolaire. Cette plateforme permet au responsable de formation d'avoir connaissance des présences/absences des étudiants, de leurs notes ainsi que les échanges que l'étudiant aura eu avec les différents services administratifs de l'Université Catholique de Lille. Le tuteur de l'étudiant aura également accès à cette plateforme et pourra ainsi suivre l'avancement de l'alternant sous sa responsabilité. Il pourra consulter ses notes, son emploi du temps ainsi que le pourcentage d'avancement dans les matières concernées.

Cette plateforme est donc axée sur l'étudiant, les notes sont enregistrées par le responsable de formation. Comme la plateforme Hyperplanning ne met pas à disposition une API pour accéder aux emplois du temps des étudiants, nous avons implémenté un système complet qui permet de recréer les différents diplômes, promotions, professeurs, séances planifiées et passées etc.
Les notes sont, dans un premier temps, confidentielles et seront rendues consultables pour le tuteur et l'étudiant via une option, ce qui permet au responsable de formation d'avoir une vision globale de l'étudiant vis-à-vis de sa promotion et de publier les notes uniquement après décision du jury.
Cette plateforme est dans un premier accessible via une application mobile compatible avec Android et iOS. Le choix de réaliser le backend sous forme d'API permettra d'interfacer facilement une application web par la suite.

Vous trouvez à cette adresse (https://app.conceptboard.com/board/di54-61k5-21gu-qpcr-4yik) un concept board qui nous a servi d'espace de brain storming au début du projet. Ce travail nous a servi de base pour la suite du projet et nous nous sommes basés dessus pour la conception de l'API et de l'application mobile.

Bilan général

Ce projet a commencé le 26 octobre 2020 et s'est terminé officiellement le 12 janvier 2021. Nous avions une charge de travail effective prévue dans l'emploi du temps de 32H par membre de l'équipe. Nous avons probablement vu trop grand pour ce projet et avons par conséquent largement dépassé le quota d'heures attribués pour la réalisation du projet et la production des documents du rendu final.

En ce qui concerne la répartition du travail, certaines tâches communes ont été réparties indifféremment entre les différents membres de l'équipe, par exemple le brain storming de départ, les user stories, la veille technologique, la rédaction et la mise en forme des éléments de rendu.

Pour la réalisation du projet nous avons constitué deux sous-équipes en fonction des aspects du projet sur lesquels nous avions envie de travailler. L'application mobile étant développée avec React Native et l'API avec Java Spring, deux technologies très différentes, il nous a semblé pertinent de nous organiser ainsi. Cette organisation en deux équipes auto-organisées a parfois rendu la communication

difficile mais cela nous a forcé à mettre en place plusieurs outils pour améliorer nos échanges :

- Les *issues*, système de gestion de tickets proposé par GitHub
- Un kanban, également proposé par GitHub pour organiser le travail sur les issues
- Git kraken pour faciliter la gestion des versions entre les membres de l'équipe
- Des réunions régulières pour faire un point sur l'avancement du projet, remonter les difficultés rencontrées et résoudre les points bloquants.
- Un serveur Discord pour faciliter la communication directe.

Les membres de l'équipe et leur compte GitHub :

Fonctionnel

DNE-MOBILE

Nous avons pris la décision de développer le frontend du projet sous forme d'application mobile car, selon nous, c'est cette forme qui nous paraissait la plus pertinente pour l'utilisation d'un étudiant. Après avoir maquetté l'application sous Adobe XD nous avons pu développer les différentes pages de cette dernière puis nous avons récupéré les différents endpoints de l'API (via le Swagger) fournis par l'équipe backend. Cela nous a permis de rentre le frontend fonctionnel.

DNE-API

Nous avons conçu le backend sous forme d'API REST de sorte que toutes les fonctionnalités demandées soient facilement implémentables lorsqu'une demande de feature par l'équipe frontend est formulée.
Pour réaliser cette API, nous avons mis en place une architecture multicouche. L'ensemble des entités, leur mapping avec l'ORM JPA a été réalisé ainsi que toute la couche service et les controllers demandés par l'équipe frontend.
Nous avons également implémenté un swagger qui suit la norme Open API afin de pouvoir consulter à tout moment une documentation à jour des endpoints de l'API. Ce Swagger permet également à l'équipe frontend de connaitre le type des paramètres passés dans les requêtes ainsi que les objets retournés et leur structure. Cette interface facilite également les tests de l'API.

Commun

Après quelques réglages, la communication entre le front et le back est fonctionnelle. Ce qui donne un rendu conforme à ce que nous avions prévu initialement et au maquettage effectué.

Technique

DNE-MOBILE

React Native

Nous avons maintenu le choix de ce framework, même si la plupart d'entre nous avions peu de connaissances en React mais de très bonnes bases en Javascript. L'apprentissage du framework fût donc plutôt rapide ce qui était un des gros atouts de celui-ci. React Native nous a également permis un gain de temps considérable dû au fait qu'il soit cross-platforme Android/IOS.

Expo

Nous avons également maintenu le choix de ce framework qui permet de développer, construire, et déployer rapidement sur IOS, Android et applications web. Son point fort est le fait de pouvoir tester en direct les modifications appliquées au code, directement sur notre smartphone.

Adobe XD

Nous voulions tout d'abord utiliser Adobe Illustrator afin de réaliser le maquettage puis un des membres du groupe a proposé d'utiliser Adobe XD qui est une bien meilleure alternative, notamment grâce à ses liens dynamiques entre les différentes pages qui permettent un rendu beaucoup plus réaliste de l'application finale. Ces maquettes sont consultables ici : https://xd.adobe.com/view/54702321-ff21-4b5b-bbce-e3e56ab45cb7-0123/?fullscreen

DNE-API

Java

Toute la pile technique de l'API dépendant de ce langage, il a évidemment été conservé.

Spring Boot

Le choix de ce framework a été conservé, étant néophytes en début de projet il nous a demandé un grand investissement mais cela nous a permis de fortement monter en compétences sur cette technologie. Spring Boot nous a permis de gagner beaucoup de temps sur la configuration et la construction des différentes couches de l'API grâce à sa prise en charge native de l'injection de dépendances. La bibliothèque Spring Security nous a permis de mettre en place la sécurité de l'application notamment pour la communication entre l'API et l'application mobile avec un standard du marché.

Hibernate

Au démarrage du projet, nous avions prévu d''implémenter l'ORM (Object Relationnal Mapping) Hibernate afin de gérer les transactions avec la base de données. La

conception de l'API et du modèle de données étant déjà suffisamment complexes, nous avons préféré nous contenter de JPA qui, en plus d'être l'ORM standard de Java, est une bibliothèque pour laquelle nous avions déjà des bases.

MySQL

Le choix initial de développer la base de données avec le SGBDR MySQL a été mis de côté pour des questions pratiques. En effet, cette API étant un POC, nous avons préféré utiliser une base de données H2 embarquée. Une base de données H2 est regénérée et stockée en mémoire à chaque redéploiement de l'application, ce qui est très pratique pour du test. Nous avons donc créé un jeu de données chargé à chaque lancement de l'API. Ce choix nous a permis, lors de développement côté front et back, de repartir avec des données « propres » à chaque redémarrage de l'application. Nous avons tout de même préparé un script SQL de création de base et effectué des tests avec un conteneur Docker embarquant MySQL.

Vous trouverez le Modèle Conceptuel de Données, réalisé avec Merise 2 à cette adresse : https://github.com/AcLabM1/DNE-Documentation/blob/main/mcd_dne.jpg

DevOps

Nous avons mis en place plusieurs outils dans une démarche DevOps afin de mettre à profit les connaissances acquises au premier semestre notamment CircleCI pour l'intégration continue (vérification de compilation, dockerisation et mise en ligne sur Dockerhub) et Docker pour la containerisation de l'application.

SonarCloud et Sonarlint

Nous avons également ajouté le plugin SonarCloud afin d'avoir un retour sur la qualité et la sécurité du code à chaque nouveau pipeline d'intégration de CircleCI déclenché par un commit sur le repository distant.
Nous avons également utilisé Sonarlint sur nos environnements de développement afin d'améliorer la qualité de notre code et de prévenir les erreurs de programmation en amont. Cet outil est utilisable sur plusieurs langages.

Dependabot

Nous avons mis en place Dependabot, plugin GitHub qui scanne automatiquement les dépendances du projet et vérifie si une mise à jour de celles-ci est nécessaire.

Conclusion

Nous sommes satisfaits de nos choix en termes de technologies. Ils correspondent aux besoins du projet et ont permis la réalisation d'une application fiable, efficace et de monter en compétence sur des standards du marché, que nous utilisons pour certains d'entre-nous, en entreprise.

Tutoriel d'installation

DNE-API

Comme mentionné précédemment, une version en ligne est actuellement disponible, les étapes suivantes ne sont pas nécessaires si vous n'avez pas de modification à

faire. Si vous souhaitez vérifier si la version en ligne est encore disponible il suffit de se rendre sur http://juliendudek.synology.me:8080/swagger-ui.html
Dans tous les cas, l'image docker et le repository restent à votre disposition, il vous suffira de suivre les étapes suivantes :

Via Docker :

A chaque évènement sur la branche main une nouvelle image docker est pushée, afin de la récupérer et la lancer vous pouvez utiliser ceci :
`docker run -it -p 8080:8080 julienm1/aclab-m1s1-dne-back:latest`

En clonant le repo :

`gh repo clone AcLabM1/DNE-API`
Puis, en passant par votre IDE préféré :
`mvn spring-boot:run`

DNE-MOBILE

Prérequis (valable pour les deux types d'installation) :

Installer l'application **Expo Go** sur votre smartphone (Android ou iOS)

Via Docker :

Installation de l'application

L'application est très facilement testable via le container docker que nous avons créé. La commande suivante permet de le récupérer :
`docker pull giannigiux/dne-mobile:latest`

Avant de lancer votre image docker, il faudra récupérer l'IP privée de votre ordinateur. (Par exemple 192.168.0.31)
Pour la trouver, voici un tutoriel : https://fr.wikihow.com/v%C3%A9rifier-son-adresse-IP-sur-Linux

Le tutoriel est prévu pour linux mais vous pourrez retrouver les mêmes informations via la commande ipconfig sous windows, en remplacement de ifconfig sous linux.

Vous pourrez ensuite simplement lancer l'application avec la commande suivante :
`docker run --env REACT_NATIVE_PACKAGER_HOSTNAME=192.168.0.31 -it -p 19000:19000 -p 19001:19001 -p 19002:19002 giannigiux/dne-mobile:latest`

/! En remplaçant 192.168.0.31 par votre adresse IP privée.

Le terminal vous affichera alors un QR code que vous pourrez scanner via l'application Expo Go. (Sélectionnez Scan QR Code et scanner le QR code)

L'application se lancera sur votre smartphone après un court chargement.

En clonant le repo :

Node

Il faudra installer nodejs et npm. Voici les commandes à lancer sous Ubuntu 20.04 :

```
sudo apt update
sudo apt install nodejs
sudo apt install npm
```

Expo

Il faudra également installer expo. Voici la commande à lancer sous Ubuntu 20.04 :

```
npm install expo-cli --global
```

/!\ ATTENTION : Pour calibrer votre application Expo Go avec le serveur lancé sur votre local, il faudra que votre ordinateur et votre smartphone soient connectés sur le même réseau Wifi. Sans cela, l'application ne se lancera pas sur Expo Go. Il faut également vous assurer que votre réseau soit bien en privé.

Liens

GitHub

Pour organiser les différents codes source du projet et sa documentation nous avons utilisé la plateforme GitHub sur laquelle nous avons créé une organisation : AcLabM1 (https://github.com/AcLabM1).

Sur cette organisation vous trouverez trois repositories :

Documentation : qui contient les documents produits pendant la réalisation du projet (https://github.com/AcLabM1/DNE-Documentation)

DNE-API : contient le code source de l'API ainsi que les instructions pour lancer l'application. (https://github.com/AcLabM1/DNE-API)

DNE-Mobile : contient le code source de l'application mobile ainsi que les instructions d'installation. (https://github.com/AcLabM1/DNE-Mobile)

L'API est provisoirement hébergée et le Swagger qui permet de la tester est consultable à cette adresse : http://juliendudek.synology.me:8080/swagger-ui.html

Démonstration

Pour une vidéo de démonstration du Swagger :
https://www.youtube.com/watch?v=YaZqoArtTss

Pour une vidéo de démonstration de l'application mobile :
https://www.youtube.com/watch?v=-iBeXJOgVUc