# RAPPORT FINAL
# DÉVELOPPEMENT WEB

## Table des matières

# Introduction

Dans le cadre de notre apprentissage au cours de développement informatique web, il nous a été demandé de réaliser un projet web au choix ou application web selon un canevas bien défini par nos enseignants respectant des consignes et des contraintes.
Commencé le 05 février 2021 avec quatre membres Joël Yepgang, Nadia Mbarushimana,Aurelle Awountsa, Brice Kouetcheu sous l'encadrement de Virginie Van den Schrieck, nous avons commencé par chercher un client ayant un projet qui saurait être accompagné et être exploité.
C'est dans ce sens, que nous avons soumis nos compétences à la responsable de la crèche Les Lionceaux de Louvain-la-Neuve qui, nous a fait part de ses problématiques et de ses attentes.
Dans la suite de ce rapport, nous expliquerons les différentes phases qui nous ont conduit à la réalisation de ce projet.

## Chap 1 : Analyse du projet

Dans le cadre du cours de Développement Informatique III il nous a été demandé de développer une application web ayant une réelle utilité dans la vie sur base des besoins d'un client. Pour ce faire, notre choix s'est porté sur le développement du site web d'une crèche afin de faciliter la gestion voire la moderniser et de réduire la paperasse.
La crèche les lionceaux est une crèche située à Louvain-la-Neuve qui jusqu'à présent disposait encore de bloc-note, de cahier registre et des factures papier pour sa gestion.
Dans le cadre de notre projet, nous avons décidé de mettre sur pied une application web qui répondra à plusieurs besoins de notre cliente la directrice de la crèche à savoir:

- Pouvoir avoir et se connecter sur son espace personnel
- Pouvoir gérer l'agenda de la crèche
- Pouvoir gérer la galerie photo de la crèche
- Disposer d'un blog où elle pourra mettre des articles pour les parents
- ...

Pour ce fait, nous avons utilisé des framework pour la réalisation du projet.
Comme framework, notre choix s'est porté sur:

- Django pour le développement de la partie Backend

- React pour le pour le développement de la partie Frontend
- Nous avons utilisé Bootstrap pour le style des pages associé au CSS.

Nous avions aussi besoin d'une base de données pour le stockage de diverses données telles que les photos, les informations relatives au personnel de la crèche ainsi que des parents et de leur enfant et nous avons utilisé PostgreSQL.
Afin d'éviter tous bugs et de vérifier si le code écrit retourne bien le résultat attendu en fonction des paramètres renseignés, il nous a paru nécessaire de faire des testings avec des outils tels que postman; nous avons aussi assuré le coté sécurité de l'application web et sachant très bien que les données confidentielles sont en jeu (données personnelles des parents, enfants et personnel de la crèche) il nous a été important voire obligatoire de respecter les normes du RGPD (Règlement général sur la protection des données) . De plus, nous avons déployé le site sur internet afin qu'il soit accessible à tous.

# Chap 2 : Technologies utilisées Frontend.

R4, R12, R10

Dans le monde de l'informatique, il existe plusieurs Framework Frontend développés en JavaScript parmi lesquels : Angular, Vue, React et bien d' autres. Chacune de ces technologies possède une architecture propre à elle.
Cependant pour le développement Frontend de notre application web, nous avons décidé d'utiliser ReactJS .

## Architecture frontend

Notre projet comprend deux répertoires, backend et frontend, qui contiendront nos différentes applications. Dans ce chapitre, on va s'intéresser à la structure du répertoire appelée *frontend* qui contiendra notre application React.

### Outils utilisés

Pour développer notre Frontend, nous avons utilisé :

Visual code : IDE

Versionning :

-Github

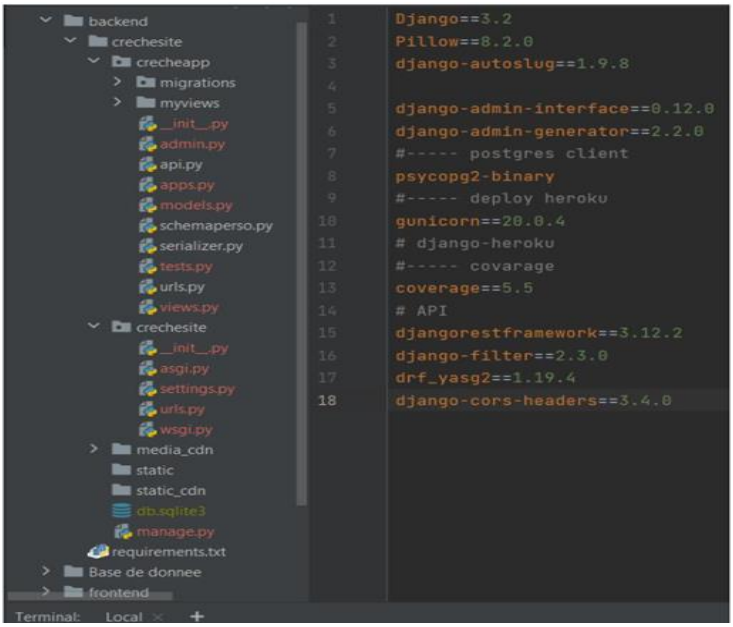
-Git

# Chap 3 : Technologies utilisées Backend

R3,R8

Le Backend, c'est la partie du code qui est exécutée par le serveur, il s'agit donc du travail qu'il réalise sur les pages Web des sites dynamiques avant de les envoyer au client. Il existe de nombreuses technologies backend telles que : symphony , flask , django, node js etc. et de nombreux serveurs de base données dont : mysql, oracle, mongodb, postgresql.
Dans la continuité du cours de python du 1ier quadri, nous avons décidé de travailler avec le Framework django. De plus , nous avons choisi postgres comme serveur de base de données

## ARCHITECTURE ET ORGANISATION DES FICHIERS




architecture ci-dessus est constituée d'un répertoire backend dans lequel, nous avons créé notre projet crèche **site** à l'intérieur se trouve notre application **crecheapp**.

Dans le même répertoire "Backend", à l'intérieur se trouve deux fichiers :

- requirements : pour les installables ou packages nécessaires pour faire tourner le projet .

- le procfile : ce dernier informe de l'emplacement du WSGI dans le projet lors du déploiement sur heroku.

# Chap 4 : API

Dans notre projet, nous avons fait intervenir un web service communiquant au travers du www (world wide web). que nous nommions API est une abréviation et signifie Application Programming Interface (ou interface de programmation d'application, en français). c'est un moyen de communication entre deux logiciels, que ce soit entre différents composants d'une application ou entre deux applications différentes. Dans notre cas nous avons utilisé l'API REST de Django

API REST

L'objectif de son utilisation dans le cadre de ce projet est que l'application fasse appel des url de manière bien définie, bien structurée pour récupérer des données et lancer des actions.
il bénéficie de nombreux avantages tels que:
la séparation du client et du serveur, qui aide à caler plus facilement les applications ;
·le fait d'être stateless, ce qui rend les requêtes API très spécifiques et orientées vers le détail ;
la possibilité de mise en cache, qui permet aux clients de sauvegarder les données, et donc de ne pas devoir constamment faire des requêtes aux serveurs.
Son principal inconvénient réside au niveau de la sécurité car il n'est pas directement intégrable. Il faut donc le développer par soi-même.

**Documentation de l'API**

Dans ce contexte, nous avons souhaité avoir une documentation accessible et facilement exploitable. Et parmi les nombreuses qui existent à l'instar de RAML , WADL et WSDL, nous avons opté pour SWAGGER car il était important pour nous d'avoir une documentation but des API est d'être utilisé par de nombreux développeurs, souvent même extérieurs au projet toujours à jour quand le code/les fonctionnalités de l'API évoluaient

-**SWAGGER**

Swagger est un langage de description d'interface pour décrire les API RESTful exprimées à l'aide de JSON
il offre un ensemble d'outils comme:

*__Swagger Inspector__ qui nous a permis de tester et générer automatiquement la documentation OpenAPI

*__Swagger Editor__ qui permet d'écrire de la documentation sur l'API, de concevoir et de décrire de nouvelles API et de les modifier.

-**ENDPOINT**

Lors de la création de notre API, le naming (nommage) fut un élément clé , pour faire simple, nous avons uniquement inclus les ressources que nous voulions mettre à jour et donc nous avons utilisé le nom de la ressource dans l'URL car l'action se trouvait déjà dans la requête http.

**blog** "https://crechesite.herokuapp.com/api/blog/"
**galerie :** Permet d'afficher la galerie photo des activités de la crèche"https://crechesite.herokuapp.com/api/galerie/"
**utilisateur:** Concerne tous les utilisateurs susceptibles de se connecter "https://crechesite.herokuapp.com/api/utilisateur/"
**parent :**Permet d'afficher les parents inscrits à la crèche https://crechesite.herokuapp.com/api/parent/"
**enfant :** Permet d'afficher les enfants inscrits à la crèche "https://crechesite.herokuapp.com/api/enfant/"
**groupe :** Permet d'afficher les deux groupes auxquels l'enfant appartient à la crèche"https://crechesite.herokuapp.com/api/groupe/"
**album:** Permet d'afficher les différents Albums photos se trouvant dans la galerie ."https://crechesite.herokuapp.com/api/album/

En conclusion, cette partie nous a permis de définir le type d'API adéquat pour notre projet, de définir le Framework à utiliser, ainsi que de faire le choix du type documentation pour mieux spécifier les différentes modifications au collaborateur lors de l'évolution du projet.

# Chap 5 :Testing

R13

## A quoi servent les tests?

Lors du développement d'une application web, nous écrivons des codes qui sont censés retourner des valeurs ou des résultats bien précis.
Dans le cadre de notre projet, nous avons été amenés à faire des tests unitaires, des tests d'intégration et des tests End 2 End

# 1)- Tests d'intégration

Pour pouvoir réaliser ces tests, nous avons utilisé le logiciel POSTMAN qui est un environnement de développement d'API complet. On peut facilement voir le résultat qu'on attend d'une API cela sous plusieurs formats: Json, Html, XML, text en fonction de notre préférence et de comment on souhaiterait utiliser ces données. Nous avons utilisé le logiciel ici pour tester l'API et automatiser ces tests
Pour le test et l'automatisation de l'API:

- Nous avons créé une collection et dans cette collection nous avons stocker les tests

-Nous avons écrit des tests sur base de certaines méthodes telles que GET,POST,PUT, le but ici est lorsque nous lançons la collection (Run collection) que nous puissions de façon automatique lancer tous les tests prévus.

# 2)- Tests unitaires

Pour les unitaires de Django nous avons utilisé le module unitest de la bibliothèque Python standard. Ce module définit les tests selon une approche basée sur des classes dans tous les fichiers dont le nom commence par test , puis de construire automatiquement une suite de tests et d'exécuter cette suite. Pour notre projet , nous avons mis en place des tests unitaire manuels pour vérifier que la requête vers l'api répondait correctement. et les tests automatisés pour ajouter les objets via différents paramètres (get, post ,delete). En ce qui concerne les tests unitaires de react, ils n'ont pas été fait.

# 3)- test end 2 end

un test e2e est une méthode qui consiste à tester l'ensemble de l'application du début à la fin pour s'assurer que le flux d'application se comporte comme prévu. Le but est de créer des cas d'usages réels ou très proches du réel.

Nous avons utilisé l'outil cypress mais nous n'avons malheureusement pas pu réaliser ces tests

# Chap 6 : Sécurité

**La Sécurité informatique** : c'est une discipline qui vise la protection de l'intégrité et la confidentialité des informations stockées dans un système informatique, en outre le contrôle des utilisateurs et des droits d'accès aux données.

## 1- Aspects sécurité

Dans cette partie , il était question pour nous d'effectuer une analyse de risque notamment au niveau du serveur web, serveur de base de données. Il a donc été nécessaire de définir une liste d'éléments à protéger, les vulnérabilités liées à la non-protection de ces dernières.

## 2- Éléments à protéger

### a- site web

Nous avons opté pour le Protocole HTTPS et le certificats SSL pour lutter contre une intrusion et garantir ainsi que l'utilisateur n'a pas été tracé. Ce qui empêche les fichiers d'être corrompus lors du transfert.

### b- Les données

Sachant, qu'une donnée est la représentation d'une information dans un programme : soit dans le texte du programme, soit en mémoire durant l'exécution. Nous avons choisi de protéger notre **SGBD** (système qui stocke et gère des données de façon organisée et cohérente) car ce dernier renferme les données utilisateurs et des images.

## 3 - Présentation des Solutions

### a- Partie logicielle

Pour pallier les attaques liées aux sessions et aux transferts d'informations, nous avons utilisé le framework Django car ce dernier apporte des modules de sécurités à l'instar de **SESSION_COOKIE_SECURE** et **CSRF_COOKIE_SECURE** que nous avons mis à **True** dans notre projet.

Ceci indique au navigateur qu'il ne doit envoyer les cookies que par des connexions HTTPS.
Cependant, cela signifie que les sessions ne fonctionneront pas en HTTP et que la protection CSRF empêchera l'acceptation de données POST par HTTP.
En ce qui Concerne les attaques malveillantes , nous avons également protéger notre site contre les attaques de type **Cross site request forgery** (CSRF) en activant

**django.middleware.csrf.CsrfViewMiddleware**. Car ce type d'attaque se produit quand un site Web malveillant contient un lien et /ou un bouton de formulaire destiné à effectuer une action sur votre site Web ,en utilisant les informations d'identification d'un utilisateur connecté qui visite le site malveillant dans son navigateur.

### b- Partie DB

**pour notre projet, nous avons utilisé AWS d' Amazon, et les mesures des Sécurités de la DB prévus par AWS sont entre autres :**

L'utilisation des connexions SSL (Secure Socket Layer) ou TLS (Transport Layer Security) avec les instances de base de données exécutant les moteurs de base de données MySQL, MariaDB, PostgreSQL, Oracle ou Microsoft SQL Server..

·Utilisation Amazon RDS pour sécuriser les informations de la base de données.

Le chiffrement Amazon RDS utilise l'algorithme standard de chiffrement AES-256 pour chiffrer les données sur le serveur qui héberge nos instances de base de données.

L'utilisation des déclarations préparées pour lutter contre les injections SQL ce qui permet de rendre les requêtes de bases de données plus fiables et plus sûres.

# Chap 7 : Déploiement

R5, R10

## 1. Analyse

Le déploiement est une étape cruciale du projet qui consiste à **une mise en production,** qui signifie le moment auquel vous migrez le code de développement en production.

Dans ce projet, nous avons choisi d'effectuer une analyse comparative de deux plateformes de déploiement de Heroku et VPS.

**Heroku :**

Heroku est une plateforme cloud en tant que service (PaaS[1]) axée sur les conteneurs. Les développeurs utilisent Heroku pour déployer, gérer et faire évoluer des applications modernes.

- Cloud "géré" (environnement déjà configuré)
- Exécute des instances AWS
- Polyvalent (exécute n'importe application)
- Propriété de Salesforce

- De nombreux modules complémentaires
- Très évolutif
- Déploiement facile ( git push heroku master)

**VPS :**

Un serveur privé virtuel (VPS, pour Virtual Private Server) est un serveur dédié virtualisé. Contrairement à un hébergement web (dit mutualisé) où la gestion technique est prise en charge par OVHcloud, c'est vous qui administrez totalement votre VPS.

- Hébergement partagé
- Nous devons installer nous-même l'environnement
- Très difficile de garder les gemmes / rubis / rails à jour
- Coûte plus que Heroku
- Si quelqu'un d'autre cause des problèmes sur le serveur, notre application est également touchée.

## -Techniques de déploiement

Pour notre projet côté Backend, nous avons utilisé Heroku qui est une entreprise créant des logiciels pour serveur qui permettent le déploiement d'applications web.
Il offre deux techniques de déploiement

- HEROKU CLI
- GITHUB

Pour notre projet, nous avons créé une application dans Heroku qui correspond à l'application que nous souhaitions déployer par la suite nous avons eu besoin des pipelines ce qui nous a permis de définir la façon dont nous devrions déployer notre application.
Cependant, notre pipeline est composé de trois étapes :

- Les Review Apps
- La staging
- La production

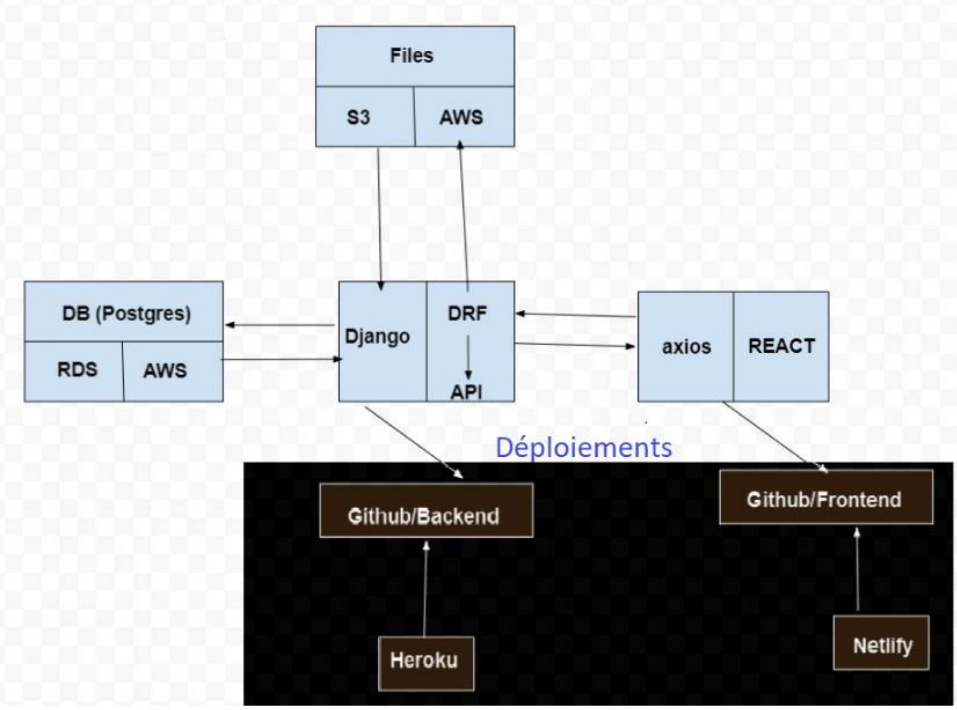
Et étant donné que notre code est sur notre github, nous avons autorisé Heroku à se connecter à notre repositories. Ainsi :

- Pour les reviews app : lorsque nous créons une Pull Request (PR) sur notre repository GitHub, Heroku détecte notre PR et fait un déploiement automatique de notre app pour cette PR.
- La staging : elle est connectée directement à notre branche master et lorsque celle-ci subit une mise à jour, Heroku déploie automatiquement une application de staging
- La production : la finalité du pipeline, elle est connectée directement à votre branche master et c'est la même chose que pour la staging, Heroku déploie automatiquement notre application en production.

Actuellement, notre code source est hébergé sur github (**frontend, backend**), nous utilisons le système *CID* [2] de heroku pour faire le déploiement.Le stockage de données est fait par Postgresql qui est hébergé sur Amazon portant le nom de **service RDS**. Les images sont stockées également via le **service S3 d'amazon.**

- Résumé du fonctionnement du site

### Architecture du site




**RDS AWS** : Relational Database Service (ou Amazon RDS ) est un service de base de données relationnelle distribué par Amazon Web Services (AWS). Il s'agit d'un service Web fonctionnant «dans le cloud» conçu pour simplifier la configuration, le fonctionnement et la mise à l'échelle d'une base de données relationnelle.

**DRF** : Django REST framework qui est une boîte à outils puissante et flexible qui nous facilite la création d'application web API
**Axios** :Composant react utilisé pour les rappels de fonction enfant, elle permet le rendu des requêtes asynchrones.
**Heroku** :Plateforme en tant que service (PaaS) qui permet de créer, d'exécuter et d'exploiter des applications entièrement dans le cloud.
**S3 AWS**: Simple Storage Service ,service de stockage d'objet offrant une évolutivité, une disponibilité des données, une sécurité et des performances de pointe.
**Netlify** : services d'hébergement et de cloud computing qui fait de la CI / CD.

# Chap 8 : Organisation du travail / équipe

R12, R18

Pour donner suite aux exigences du corps enseignant, nous avons constitué un groupe de 4 étudiants. Nous avons donc séparé le travail en 2 sous-groupes, le premier groupe était donc chargé de travailler avec travailler sur la partie backend, tandis que le deuxième était responsable de la partie frontend du projet.

## OUTILS UTILISÉS

Afin de garantir une bonne collaboration dans le groupe nous avons utilisés de nombreux outils tels que :

Discord : Nous avons le plus souvent utilisé cette plateforme pour assurer la communication textuelle entre les membres du groupe premièrement et le corps enseignant deuxième

Teams: Tout au long de ce projet il nous a permis de planifier des réunions afin de discuter oralement de l'état d'avancement du projet et de préparer les entrevues avec l'enseignante

Trello: C'est un gestionnaire qui nous a beaucoup aidé dans la séparation des tâches et le suivi de ces dernières

Clokify: C'est un minuteur qui nous a permis d'évaluer le temps consacré au projet

Git et Github

**Git** est un logiciel de gestion de versions décentralisé et **GitHub** est un service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git.

Conclusion

Ce projet était très intéressant et enrichissant. Il nous a permis d'apprendre à utiliser plusieurs technologies.
La recherche du projet s'est faite sans difficulté grâce à un client qui nous a soumis son projet. Les besoins du client et les solutions ont été mis en évidence et compris par tous les membres de l'équipe.
Cependant au cours de la réalisation du projet, nous avons rencontré des difficultés liées à un manque de cohésion au sein de l'équipe ce qui a ralenti considérablement le travail et créé des tensions au sein de l'équipe.
Bien que nous n'ayons pas réussi à atteindre l'objectif fixé de « livrer un projet fini », nous avons réussi à accomplir toutes les étapes du projet et du coaching avec succès.
On sort de ce projet avec une bonne expérience professionnelle

# References

## Frontend

https://www.ganatan.com/tutorials/routing-avec-angular

https://esokia.com/fr/blog/comparatif-angular-reactjs-vuejs

https://www.codeur.com/blog/choisir-framework-javascript/

https://formationjavascript.com/angular2-vs-react/

https://blog.dyma.fr/quel-framework-choisir-en-2020-angular-vue-js-ou-react/

https://www.youtube.com/watch?v=NPvjiYeLfVw

https://www.youtube.com/watch?v=D3oivlcoEvw

https://www.youtube.com/watch?v=D3oivlcoEvw

https://easypartner.fr/blog/que-choisir-entre-react-ou-angular/

https://waytolearnx.com/2019/03/difference-entre-angular-et-react.html

https://openclassrooms.com/fr/courses/4664381-realisez-une-application-web-avec-react-js

https://www.digitalocean.com/community/tutorials/react-axios-react

## Backend

https://www.ganatan.com/tutorials/django-avec-angular

https://openclassrooms.com/fr/courses/4425076-decouvrez-le-framework-django

https://alphacoder.xyz/deploy-react-django-app-on-heroku/

## Base de données

https://python.doctor/page-database-data-base-donnees-query-sql-mysql-postgre-sqlite

https://docs.djangoproject.com/fr/3.2/topics/security/

## Api

- https://www.django-rest-framework.org
- https://djangostars.com/blog/rest-apis-django-development/
- https://openclassrooms.com/fr/courses/6573181-adoptez-les-api-rest-pour-vos-projets-web/6817216-identifiez-les-avantages-d-une-api-rest

## Documentation api

- https://drf-yasg2.readthedocs.io/en/latest/
- https://swagger.io/
- https://dev.to/dianamaltseva8/why-use-swagger-for-creating-and-documenting-apis-115l

## US

https://www.all4test.fr/blog-du-testeur/criteres-dacceptation-objectifs-formats-best-practice/#:~:text=%20Principaux%20objectifs%20des%20critères%20d’acceptation%20%201,communication.%20Les%20critères%20d’acceptation%20synchronisent%20les...%20More

# Sécurité

https://www.intruder.io/?utm_source=referral&utm_campaign=geekflare_online_scan_website_security_vulnerabilities

https://help.intruder.io/en/articles/2367044-aws-integration