

**Nadia Mbarushimana**  
**Joel Yepgang**

# **RAPPORT FINAL**

## **DÉVELOPPEMENT WEB**

### **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

A ce jour, ce projet est continué par deux membres donc Joël Yepgang et Nadia Mbarushimana.

Pour la réalisation de ce projet, 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.

# RI : Description du projet (client, besoin)

## Présentation du client

Les lionceaux situé au centre de Louvain-la-neuve avec comme directrice Madame Alice est une crèche pour enfants de moins de 5 ans qui apporte un suivi à chaque enfant tout en les accompagnant dans leurs éveils. C'est ainsi l'occasion de favoriser la sociabilisation de l'enfant puisqu'il est en contact avec d'autres enfants de son âge mais également d'autres adultes que ses parents.

## Présentation du projet

La crèche travaille actuellement avec les registres soit pour inscrire les nouveaux enfants ou la journée effectuée par chaque enfant. Concernant les annonces ou d'autres informations, elle passe par les réseaux sociaux pour atteindre les parents. La recherche d'information devient de plus en plus difficile autant pour les parents que pour le personnel. Pour numériser et centraliser les informations, le responsable de l'établissement souhaiterait disposer d'une interface qui lui permettrait de réaliser ces étapes.

## R2: user stories individuels

Voir les user stories et maquette sur le lien ci-dessous :

<https://github.com/bricekouetcheu/Les-lionceaux/wiki/R2:-user-stories-individuels>

## Technologie utilisées

### R3: justification choix back end

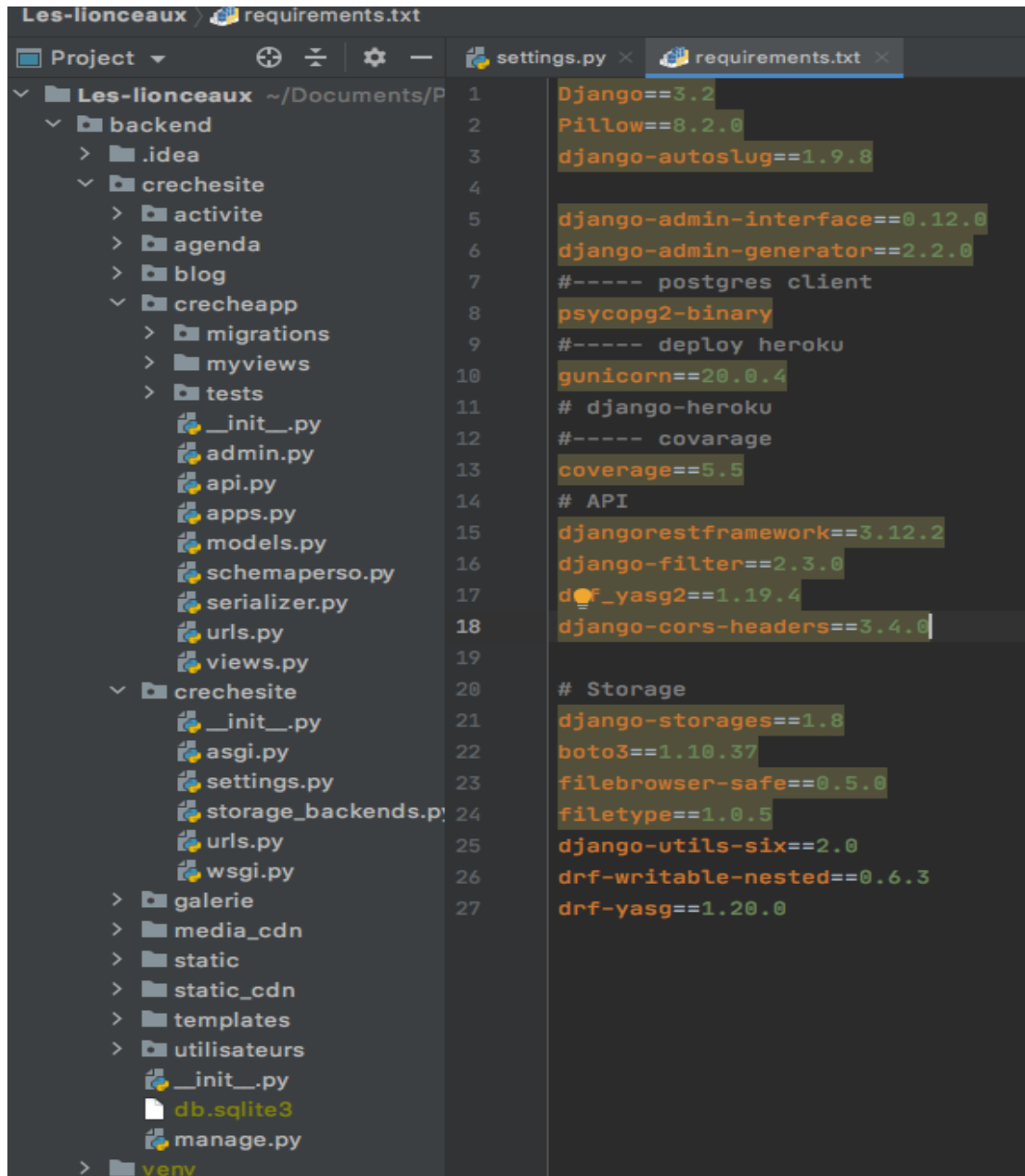
#### **Backend (4.1a)**

Pour notre projet afin d'approfondir nos connaissances avec le langage Python, nous nous sommes orientés vers les Framework Python. Nous avons constaté qu'il existe plusieurs variétés notamment Flask, Tornado, CherryPy, Web2Py, Bootle, Django. Cependant deux d'entre eux dont Flask et Django ont retenu notre attention parce qu'ils sont plus connus et plus utilisés dans l'ensemble assez similaires mais encore adaptés pour réaliser notre projet.

Flask est un Framework simple, très léger, minimaliste mais qui offre moins de fonctionnalités que le Framework Django qui est très lourd mais qui facilite à ses utilisateurs certaines tâches communes de développement, comme l'authentification, le routage d'URL, ou encore la migration des schémas de base de données.

Revenons sur notre application, après avoir comparé ces deux Framework qui sont intéressants l'une à l'autre, notre choix s'est orienté vers Django comme Framework backend parce qu'il nous impose déjà une structure de projet (MVT) ce qui nous permettra d'emblée d'avoir une organisation rigoureuse de notre structure et un gain de temps contrairement à Flask qui nous laisse la liberté sur l'organisation de la structure du projet. Nous l'avons également choisi pour sa popularité, ce qui nous garantit un support en cas de problème.

## Architecture Backend (4.1a)



l'architecture suivante est constituée d'un répertoire backend dans lequel, nous avons créé notre projet **crechesite** 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.

# R4: justification choix front end

## Frontend (5.1)

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.

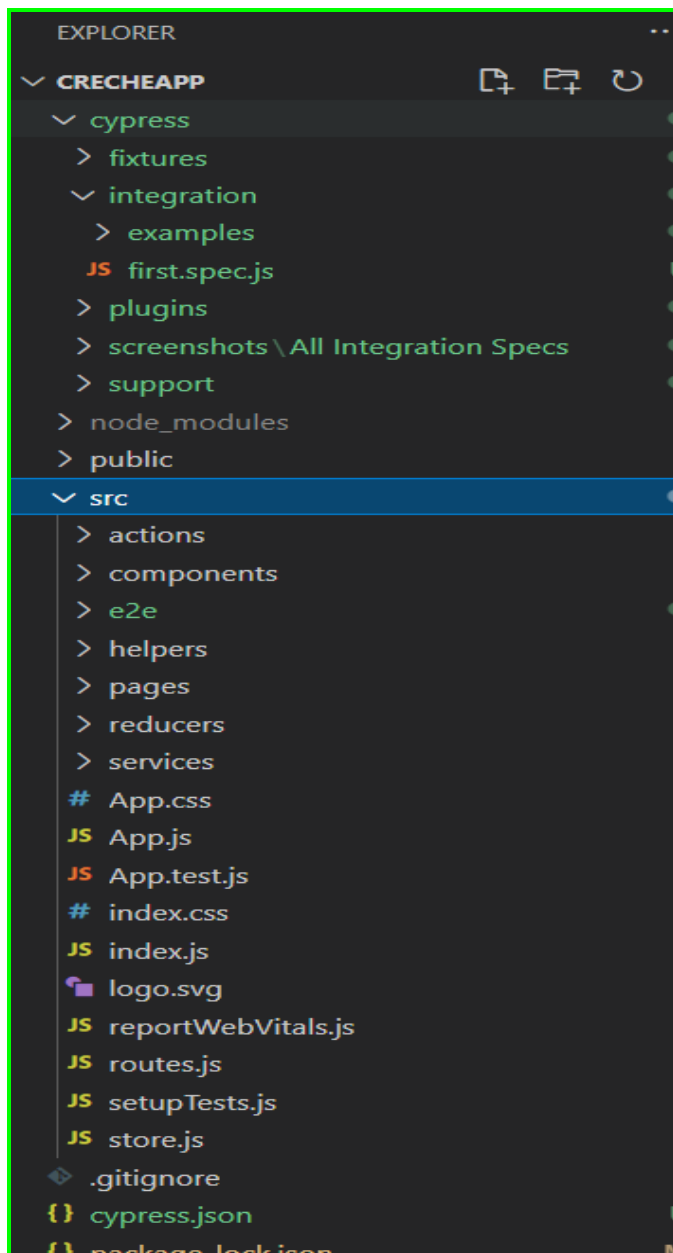
1. Angular.js est un Framework open-source qui a l'avantage d'être hyper compétitif sur le critère du coût. C'est aussi le choix idéal pour créer une application web en structure complète CRUD (Create, Read, Update et Delete) avec des possibilités avancées de modification
2. React est une librairie JavaScript qui a fait ses preuves sur Facebook, Instagram et d'autres plateformes de social community development. Son principal avantage est qu'il permet de visualiser une page web dynamique ou interactive de manière simple, directe et rapide.
3. Vue.js Direct et simple d'utilisation, Vue.js est parfait pour les débutants qui souhaitent apprendre le développement des applications web de façon intéressante.

Cependant pour le développement Frontend de notre application web, nous avons décidé d'utiliser ReactJS pour plusieurs raisons.

- Il possède un haut niveau de flexibilité et de réactivité maximum.
- Il possède DOM virtuel (modèle d'objet de document) qui permet d'organiser des documents au format HTML, XHTML ou XML dans un arbre à partir duquel est mieux accepté par les navigateurs Web tout en analysant différents éléments de l'application Web.
- Librairie JavaScript 100% open source qui reçoit beaucoup de mises à jour et d'améliorations quotidiennes en fonction des contributions des développeurs du monde entier.
- Il produit du code « propre » (simple à lire), sa lecture permet de déterminer immédiatement quelles sont les fonctionnalités de notre application. Ce qui est essentiel pour la maintenance et l'expansion de notre projet dans le temps.
- Il a un contenu référencé : C'est la fonctionnalité qui fait la différence par rapport aux autres frameworks. Grâce à l'utilisation d'un serveur Node, le code va pouvoir être généré côté client et côté serveur.

Comparé à Angular , ReactJS est beaucoup plus léger et facilite l'apprentissage cependant il aura comme inconvénient sa documentation qui n'est pas assez fournie.

## Architecture frontend



Description le projet frontend est répartie comme suit:

1. Répertoire **public** qui contient tous les fichiers html et les fichiers de styles
2. un répertoire src
3. **Components** qui contiendra évidemment tous les composants du projet

4. **page**: dans ce répertoire on crée toutes les pages du projet afin de faciliter la navigation d'une page à l'autre. c'est dans ces pages qu'on appellera les différents composants créés \*service: qui contient les fonctions d'appel a l'API
5. **APP.js** : qui est le fichier racine dans lequel on appellera toutes les pages créées
6. index.css : qui est le fichier racine css dans lequel on appellera les différents fichiers css liés aux composants
7. **App.test.js**: ce répertoire contiendra l'ensemble de nos test unitaires

# R5: présentation architecture, fonctionnement du site et techniques de déploiement

## Architecture et fichiers(4.1b)

Au niveau de l'architecture, le framework Django utilise l'architecture MVT (Model-view -Templates) qui s'inspire de MVC ( Model-view-controller ) qui est utilisé dans d'autres langages de programmation. nous nous sommes appuyés dessus pour réaliser notre travail.

## Techniques de déploiement

pour effectuer notre déploiement, nous avons effectué un comparatif Heroku VS VPS

Heroku :

- Cloud "géré" (environnement déjà configuré)
- Exécute des instances AWS
- Polyvalent (exécute n'importe quelle application)
- Propriété de Salesforce
- De nombreux modules complémentaires
- Très évolutif
- Déploiement facile ( git push heroku master)

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.

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 technique de déploiement

- HEROKU CLI
- GITHUB

Pour notre projet, nous avons créé une application dans Heroku qui correspond à l'application que nous souhaitons déployer par la suite nous avons eu besoin des pipeline 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 connecté 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 de la pipeline, est connectée directement à notre branche master et fera la même chose que pour la staging, Heroku déploiera automatiquement notre application en production.

voir :

<https://github.com/bricekouetcheu/Les-lionceaux/wiki/R5:-pr%C3%A9sentation-architecture,-fonctionnement-du-site-et-techniques-de-d%C3%A9ploiement>

Actuellement, notre code source est hébergé sur github (**frontend, backend**), nous utilisons le système CID de heroku pour faire le déploiement.

Le stockage de données est fait par postgres qui est hébergé sur amazon portant le nom de service RDS. Les images sont stockées également via le service **S3 d'amazone**.

## R6: diagramme de classes (Maquettes)

Voir Maquette sur le lien ci-dessus:

[https://github.com/bricekouetcheu/Les-lionceaux/wiki/R6:-diagramme-de-classes-\(Maquettes\)](https://github.com/bricekouetcheu/Les-lionceaux/wiki/R6:-diagramme-de-classes-(Maquettes))  
)

## R7: justification choix DB

### Base de données( 4.3)

Parmi les plus connus SGBDR (système de gestion de base de données relationnelles) existants:

- PostgreSQL,
- MySQL,
- SQLite,
- MariaDB,
- Oracle,
- Microsoft SQL Server.

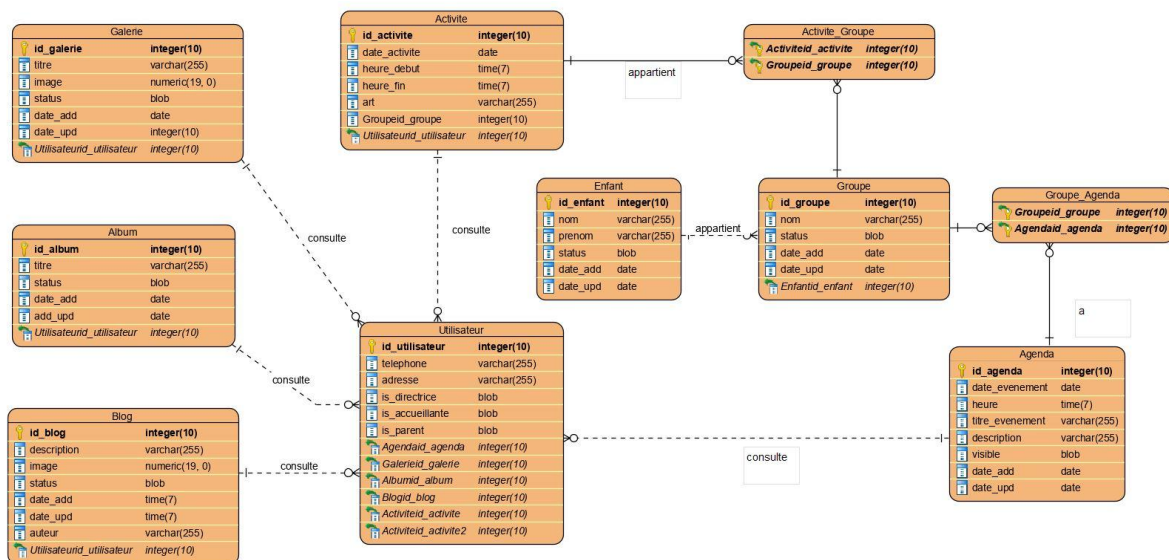
Par défaut, notre backend Django vient avec une BDD Sqlite3 mais qui est très rarement mis en production parce que c'est une base de données de développement ou de test . Cependant, notre choix s'est porté uniquement sur le SGBDR open source et orienté client/serveur en l'instar de MySQL, MariaDB et PostgreSQL.

Pour notre projet , nous avons choisi PostgreSQL non seulement parce qu'elle est basée sur le modèle client-serveur classique et compatible avec notre backend Django.

Elle permet l'interaction multiple utilisateurs sans pertes des données contrairement à la BD SQLite utilisée pour les petits projets mais qui ne répond pas à la logique client-serveur. C'est un SGBDR basé sur des fichiers, il stocke les données dans un seul fichier, ce qui signifie qu'il est simple de le copier, de le stocker et de le partager mais en cas de données volumineuses , ça peut ralentir l'accessibilité du site .

PostgreSQL reste intéressant pour l'intégrité et la fiabilité des données mais aussi pour notre hébergement et déploiement sur Heroku (plateforme d'hébergement gratuite en ligne ).

## R8: Architecture de la DB



Pour la base de donnée, nous avons créer plusieurs tables à savoir:

- La table 'Enfant' qui contient les informations relatives à l'enfant(nom, prénom, groupe...) et qui est liée:
  - A la table 'Groupe' par une clé étrangère 'id\_Groupe'
- la table 'Utilisateur' qui contient les informations relatives à la directrice, aux parents, à l'accueillante est liée:
  - A la table Galerie par la clé étrangère 'id\_galerie'
  - A la table Album par la clé étrangère 'id\_album'
  - A la table Blog par la clé étrangère 'id\_blog'
  - A la table Activité par la clé étrangère 'id\_activite'
  - A la table Agenda par la clé étrangère 'id\_agenda'
- La table 'Groupe' qui contiendra les différents groupes d'enfants de la crèche sélectionné par tranche d'âge
- La table 'Galerie' qui contient les différents Album contenant des images.
- La table 'Album' qui contient les différentes images de la crèche.
- La table 'Blog' qui contient les différents articles édités par la Directrice.

- La table 'Activité' qui contient l'ensemble des programmes de la crèche, qui est liée:
  - A la table 'Groupe' par une clé étrangère 'id\_Groupe'
- La table 'Agenda' les dates des activités des enfants en fonction des groupes auxquels ils appartiennent.

## R9 : API

### Choix de l'API

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.

On en distingue deux grands types qui sont les API REST (representational State Transfer ) et les API SOAP (simple Object Access Protocol).

### ETUDE COMPARATIVE

SOAP est l'acronyme de Simple Object Access Protocol, ou protocole simple d'accès aux objets, en français. Contrairement à REST, il est considéré comme un protocole, et non comme un style d'architecture.

REST utilise le protocole HTTP pour communiquer, SOAP d'un autre côté peut utiliser de multiples moyens de communication.

### REST

il bénéficie de nombreux avantages tels que:

- la séparation du client et du serveur, qui aide à scaler 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

## SOAP

Son principal avantage est qu'il bénéficie d'une couche de sécurité et de confidentialité facilement intégralement.

Par contre, il a quelques inconvénients. tels que la complexité qui en ressort, car les développeurs doivent se coordonner pour s'assurer qu'ils communiquent de la même manière afin d'éviter les problèmes. De plus, le SOAP peut demander plus de bande passante, ce qui entraîne des temps de chargement beaucoup plus longs.

Dans le cadre de notre projet nous avons donc décidé de mettre en place une API REST pour sa nature plus légère et plus flexible.

## Documentation de l'API : SWAGGER

Malgré qu'il existe plusieurs frameworks pour documenter notre api notamment RAML, APIBlueprint, Summation, Wrapper , Swagger et autres . Notre choix s'est orienté vers Swagger qui est le plus grand framework pour la conception d'API utilisant un langage commun et permettant le développement tout au long du cycle de vie de l'API, y compris la documentation, la conception, les tests et le déploiement. Il peut être utilisé avec succès pour les tests d'API et la correction des bug.

Nous l'avons également choisi parce qu'il offre un ensemble d'outils comme:

- Swagger Inspector qui nous a permis de tester et générer automatiquement la documentation OpenAPI
- Swagger Editor permet d'écrire de la documentation sur l'API, de concevoir et de décrire de nouvelles API et de modifier celles existantes

crechesite.herokuapp.com/swagger/

Applications Service clientèle Hello bank! Amazon.fr : Votre a... Mon AliExpress : gé... Mon eBay Enchères... IBZ Immigration, Divers... MySQL = MySQL 5...

Autres favoris Liste de lecture

Swagger

https://crechesite.herokuapp.com/swagger/?format=openapi Explore

# Lionceaux API <sup>v1</sup>

[ Base URL: crechesite.herokuapp.com/ ]  
https://crechesite.herokuapp.com/swagger/?format=openapi

Test description  
Terms of service  
Contact the developer  
BSD License

Schemes  
HTTPS

Django Django Logout Authorize

Filter by tag

api-token-auth

POST /api-token-auth/ api-token-auth\_create

api

GET	/api/activite/	api_activite_list	🔒
POST	/api/activite/	api_activite_create	🔒
GET	/api/activite/{id}/	api_activite_read	🔒
PUT	/api/activite/{id}/	api_activite_update	🔒
PATCH	/api/activite/{id}/	api_activite_partial_update	🔒
DELETE	/api/activite/{id}/	api_activite_delete	🔒
GET	/api/agenda/	api_agenda_list	🔒
POST	/api/agenda/	api_agenda_create	🔒
GET	/api/agenda/{id}/	api_agenda_read	🔒
PUT	/api/agenda/{id}/	api_agenda_update	🔒
PATCH	/api/agenda/{id}/	api_agenda_partial_update	🔒
DELETE	/api/agenda/{id}/	api_agenda_delete	🔒

DELETE	/api/agenda/{id}/	api_agenda_delete	🔒
GET	/api/album/	api_album_list	🔒
POST	/api/album/	api_album_create	🔒
GET	/api/album/{id}/	api_album_read	🔒
PUT	/api/album/{id}/	api_album_update	🔒
PATCH	/api/album/{id}/	api_album_partial_update	🔒
DELETE	/api/album/{id}/	api_album_delete	🔒
GET	/api/blog/	api_blog_list	🔒
POST	/api/blog/	api_blog_create	🔒
GET	/api/blog/{id}/	api_blog_read	🔒
PUT	/api/blog/{id}/	api_blog_update	🔒
PATCH	/api/blog/{id}/	api_blog_partial_update	🔒
GET	/api/enfant/	api_enfant_list	🔒
POST	/api/enfant/	api_enfant_create	🔒
GET	/api/enfant/{id}/	api_enfant_read	🔒
PUT	/api/enfant/{id}/	api_enfant_update	🔒
PATCH	/api/enfant/{id}/	api_enfant_partial_update	🔒
DELETE	/api/enfant/{id}/	api_enfant_delete	🔒
GET	/api/galerie/	api_galerie_list	🔒
POST	/api/galerie/	api_galerie_create	🔒
GET	/api/galerie/{id}/	api_galerie_read	🔒
PUT	/api/galerie/{id}/	api_galerie_update	🔒
PATCH	/api/galerie/{id}/	api_galerie_partial_update	🔒
DELETE	/api/galerie/{id}/	api_galerie_delete	🔒



# ENDPOINT

blog

"blog": "<http://crechesite.herokuapp.com/api/blog/>",

galerie

Permet d'afficher la galerie photo des activités de la crèche

"galerie": "<http://crechesite.herokuapp.com/api/galerie/>",

utilisateur

Concerne tous les utilisateurs susceptible de se connecter

"utilisateur": "<http://crechesite.herokuapp.com/api/utilisateur/>",

enfant

Permet d'afficher les enfants inscrits à la crèche

"enfant": "<http://crechesite.herokuapp.com/api/enfant/>"

groupe

Permet d'afficher les deux groupes auxquels l'enfant appartient à la crèche

"groupe": "<http://crechesite.herokuapp.com/api/groupe/>",

album

Permet d'afficher les différents Albums photos se trouvant dans la galerie .

"album": "<http://crechesite.herokuapp.com/api/album/>",

activité

Permet d'afficher les différents programmes de la crèche .

"activité": "<http://crechesite.herokuapp.com/api/activite/>",

agenda

Permet d'afficher les différentes dates des activités des enfants en fonction des groupes.

"agenda": "<http://crechesite.herokuapp.com/api/agenda/>",

# R10:manuel d'installation (et non d'utilisation)

## Introduction

Il s'agit d'une simple application Web Les lionceaux construite à l'aide de React.js (front-end) et Django (back-end) .

La mise en place de l'application commence par la création d'un environnement de travail IDE: pycharm (backend) , visual code (frontend) ainsi qu'un github pour héberger notre code source .

## Organisation de travail

Notre méthode nécessite de créer deux applications séparément: d'abord l'application Django (views.py) et les urls (urls.py) et ensuite React (npm run build).

Voir technologies utilisées ([outils utilisés](#))

## Condition préalable

### Backend

1. Installation et configuration du backend
  - création d'un répertoire backend qui contiendra notre application django
  - mise en place et activation d'un environnement virtuel :  
python3 -m venv venv  
source venv/bin/activate
  - Installez les dépendances : pip install -r requirements.txt
  - création d'un nouveau projet Django nommé crechesite
  - création d'une application django appelé crecheapp  
ensuite exécuter des migrations et démarrer le serveur.

Voir architecture backend ([backend-architecture](#))

## 2. Configuration API

- mise à jour de la section `INSTALLED_APPS` dans le fichier `backend/settings.py`
- Pour permettre les opérations API CRUD entre le frontend et le backend, nous devons utiliser le Framework Django REST ainsi que les en- têtes Django CORS .

En plus du framework Django REST puissant et flexible pour la création d'API Web, Django CORS Headers est une application Django permettant de gérer les en-têtes de serveur requis pour le partage de ressources cross-origin (CORS) .

Django exporte les API REST à l'aide de Django REST Framework et interagit avec la DB à l'aide de Django Model.

Après l'installation des dépendances requises, nous avons ajouté `rest_framework` et `corsheaders` à la liste des applications installées( voir le fichier `backend/crechesite/crechesite/settings.py`) et tout en adaptant les sections `INSTALLED_APPS` et en `MIDDLEWARE` en conséquence.

- Création de sérialiseurs pour le modèle et les vues Django  
Nous avons implémenté les sérialiseurs pour convertir les instances de modèle en JSON afin que le frontend puisse fonctionner facilement avec les données reçues, ce qui nécessite l'importation des sérialiseurs et la définition `serializers.ModelSerializer` comme paramètre de classe de modèle (voir `backend/crechesite/crecheapp/serializer.py`).
- Mise à jours le `backend/crechesite/crechesite/urls.py`  
nous sert de routeur pour effectuer les requêtes : `crecheapp/api/id`

voir API([API-documentation](#) )

## Frontend

### 1. Création d'une nouvelle application React

Après l'installation de Node.js, vous pouvez démarrer rapidement la création de votre première application React dans un répertoire Frontend à l'aide des commande suivante :

```
npx create-react-app frontend
```

```
npm start
```

- ### 2. Développement l'application React mise en place des dépendances nécessaires (voir [outils utilisés](#))
- ### 3. Connecter Django à React

Pour effectuer les requêtes aux points de terminaison de l'API sur le serveur Django, nous aurons besoin d'installer une bibliothèque JavaScript appelée Axios

Axios est un client HTTP populaire basé sur la promesse, doté d'une API facile à utiliser et pouvant être utilisé à la fois dans le navigateur et Node.js

Nous allons installer Axios : `npm install axios` et vérifier dans le package json si l'installation réussie et ensuite nous procéderons à la configuration d'un proxy dans le dossier `services/blog.service.js`

Le proxy aidera à tunneliser les requêtes API vers http:

<https://crechesite.herokuapp.com/api/> où l'application Django les gèrera, afin que nous puissions simplifier l'écriture des requêtes.

#### 4. Testez l'application Web

Vous pouvez vérifier si tout fonctionne en exécutant simultanément React et Django

```
npm start
```

```
python manage.py runserver
```

Voir arborescence des fichiers ([Frontend React](#))

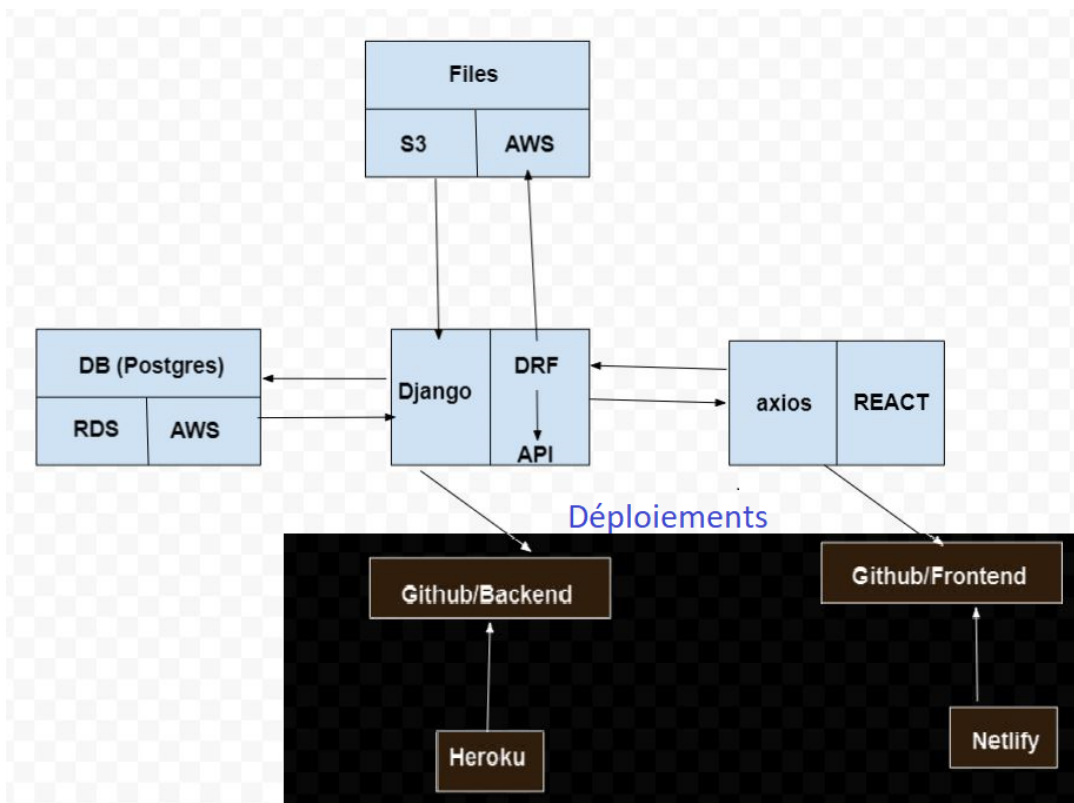
## Base de données et stockage de fichiers

- Configuration d'un compte AWS
- Création d'une base de données PostgrésqI sur le cloud AWS Amazon via le service RDS.
- Mise à jour de notre fichier settings et lancer les commandes de migrations.
- Migration de la base de données SQLITE vers PostgresQL
- Création d'un serveur de stockage de fichiers et images via le service S3 sur AWS Amazon
- Mise à jour du fichier `crechesite/storage_backend.py` et installation des librairies nécessaires ([outils utilisés](#))

## Déploiement

Voir technique de déploiement [déploiement Héroku](#)

## Architecture du site



## Résumé du fonctionnement 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.

R11: tous les liens du projet  
(site, trello, timesheet, etc...)

## SITE

<https://crechesite.herokuapp.com/>

<https://les-lionceaux.netlify.app/>

## TRELLO

<https://trello.com/b/UfAbsgVu/projet-de-d%C3%A9veloppement-informatique-iii>

## CLOCKIFY

<https://clockify.me/tracker>

# R12: utilisation de git et github : interactions, workflow, outils utilisés

## IDE

- Pycharm ( backend)
- Visual Code ( Frontend)

## Versionning

- Github
- Git

## Workflow

Pour notre projet, nous avons choisi d'utiliser un github flow ayant comme principe une seule branche master et plusieurs autres branches.

Pour notre cas , nous avons une seule branche master et quatre branches .

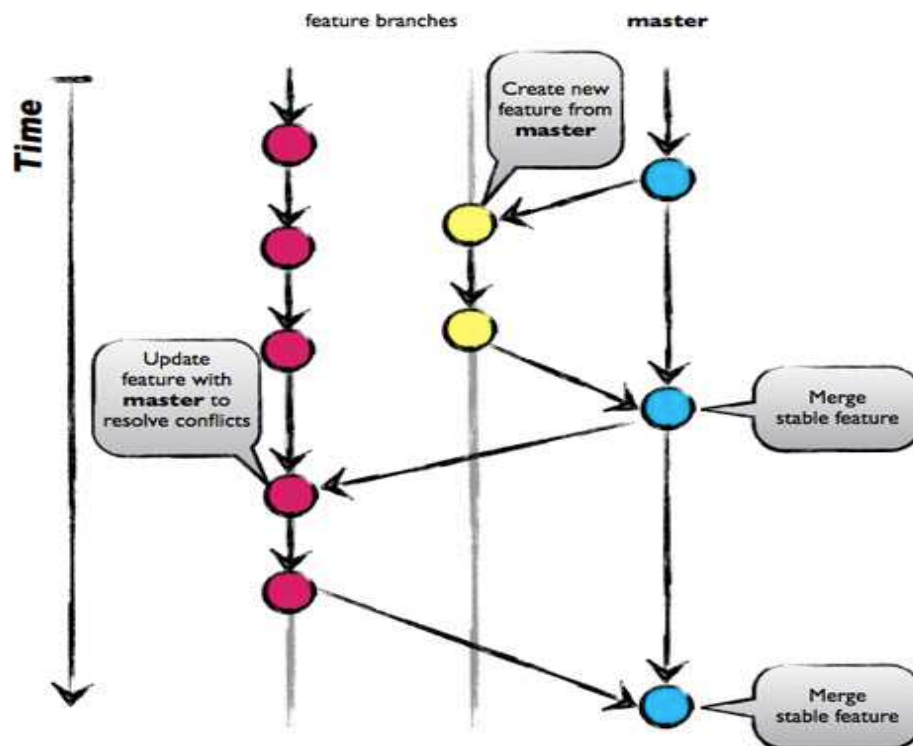
Chaque branche est créée pour chaque membre de l'équipe qui servira à recevoir régulièrement ses commits .

Dès que le travail est suffisamment avancé voir terminé , il sera plus facile depuis une branche, de faire un diff avec master et d'ouvrir une pull-request et un autre membre doit valider un pull-request.

Une fois que la pull-request est validée, la branche est mergée dans master ce qui signifie que le tout est déployé en production.

Seule notre branche master peut être déployée en production.

## model workflow utilisé



*Le modèle GitHub flow illustré*

## Interaction

- Discord
- Teams
- outlook ( organisation réunion)
- googledoc ( partage document)

## Maquette

- Pencil

## architecture DB

- visual Paradigm

## autres outils



- Trello (organisation des tâches)
- clockify (gestion du temps)

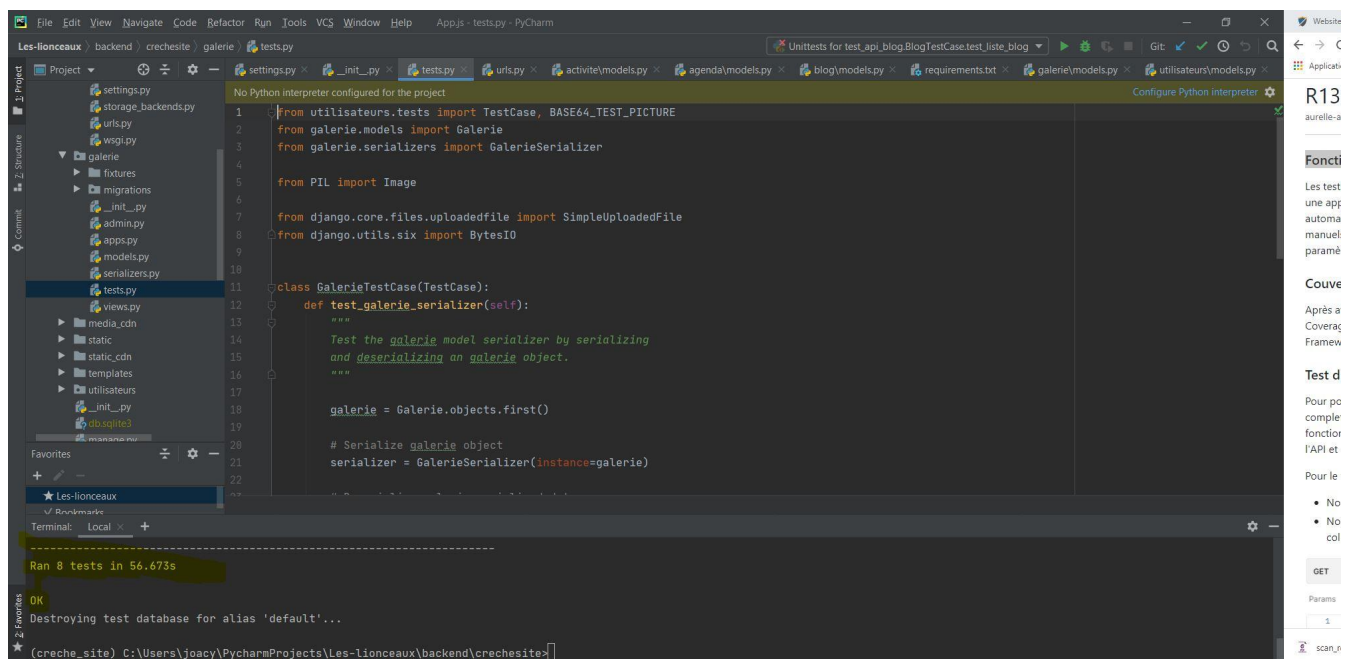
## R13: présentation tests unitaires

### Fonctionnement

Les tests unitaires de Django utilisent le module unittest 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 unitaires sur chacune de nos classes.

### Couverture du code

Après avoir effectué les tests unitaires, nous avons voulu mesurer la partie du code testé par la mise en place d'un outil Coverage qui permet de surveiller quelles parties de code sont testées. Grâce à une librairie Unittest installée par défaut dans le Framework Django qui permet de lancer les tests écrits.



```
1 from utilisateurs.tests import TestCase, BASE64_TEST_PICTURE
2 from galerie.models import Galerie
3 from galerie.serializers import GalerieSerializer
4
5 from PIL import Image
6
7 from django.core.files.uploadedfile import SimpleUploadedFile
8 from django.utils.six import BytesIO
9
10
11 class GalerieTestCase(TestCase):
12     def test_galerie_serializer(self):
13         """
14         Test the galerie model serializer by serializing
15         and deserializing an galerie object.
16         """
17         galerie = Galerie.objects.first()
18
19         # Serialize galerie object
20         serializer = GalerieSerializer(instance=galerie)
```

Ran 8 tests in 56.673s

Destroying test database for alias 'default'...

(creche\_site) C:\Users\joacy\PycharmProjects\Les-lionceaux\backend\crechesite>

# R14: présentation aspects sécurité, avec petite analyse de sécurité

## Présentation aspects sécurité

Par mesure de sécurité de l'application, nous entendons la confidentialité des données échangées, leur intégrité et l'authentification.

## Partie logicielle

Pour pallier tous les problèmes de sécurité, nous avons utilisé un framework Django qui apporte des modules de sécurités comme par exemple les réglages `SESSION_COOKIE_SECURE` et `CSRF_COOKIE_SECURE` que nous avons mis à `True` dans notre projets (voir settings middleware). Cela indique au navigateur qu'il ne doit envoyer les cookies que par des connexions HTTPS. Notez que cela signifie que les sessions ne fonctionneront pas en HTTP et que la protection CSRF empêchera l'acceptation de données POST par HTTP.

Concernant 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`. Ce type d'attaque se produit quand un site Web malveillant contient un lien, 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.

## Partie DB

Les mesures des Sécurités de la DB prévus par AWS sont entre autres : Utilisez 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. Pour de plus amples informations sur l'utilisation de SSL avec une instance de base de données, veuillez consulter Utilisation de SSL/TLS pour chiffrer une connexion à une instance de base de données.

Utilisez Amazon RDS pour sécuriser vos instances de base de données et les instantanés au repos. Le chiffrement Amazon RDS utilise l'algorithme standard de chiffrement AES-256 pour chiffrer vos données sur le serveur qui héberge votre instance de base de données. Pour en savoir plus, consultez Chiffrement des ressources Amazon RDS.

## Partie serveur

Au niveau du serveur , nous avons opté de migrer nos données sur une base de données PostgreSQL se trouvant dans le cloud Amazon Web Services (AWS) et de stocker tous nos fichiers sur un serveur de stockage S3 .

AWS est la plateforme cloud la plus complète conçue pour permettre aux fournisseurs d'applications d'héberger rapidement et de manière sécurisée les applications.

Nous avons choisi d'héberger les données sur le cloud Amazon pour éviter le risque de perdre les données des utilisateurs et pour la disponibilité de notre site.

En plus de la sécurité de l'application qu'offre AWS, il propose également une infrastructure sécurisée et fiable.

Pour accéder aux données personnelles des utilisateurs, nous avons mis en place un système d'authentification. Grâce à AWS les mots de passe seront automatiquement chiffré et respecte les exigences de complexité des mots de passe.

# Rapport test sécurité du site

[rapport sécurité site](#)

R16: toutes les références de sites et autres utilisés pour vous aider dans le projet

## Références

### 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%20crit%C3%A8res%20d%27acceptation%20%201,communication.%20Les%20crit%C3%A8res%20d%27acceptation%20synchronisent%20les...%20More>

## Sécurité

[https://www.intruder.io/?utm\\_source=referral&utm\\_campaign=geekflare\\_online\\_scan\\_website\\_security\\_vulnerabilities](https://www.intruder.io/?utm_source=referral&utm_campaign=geekflare_online_scan_website_security_vulnerabilities)

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

## R17: interactions avec le client

Pour tester notre Frontend, nous avons créé un nouveau projet angular "mon-premier-projet" qui permet de montrer à la directrice de la crèche si les places sont disponibles dans une classe ou non. Et au bout de 9 mois(cycle scolaire) les statuts de toutes les classes devraient passer à disponible .

Pour ce fait, nous avons créé un component "classes" Nous avons aussi utilisé les directives structurales (ngif et ngfor) et les directives par attribut (ngclass...)

## R18: Fonctionnement du groupe, outils utilisés

### Répartition des rôles

- Joel: Gestion des équipes
- Nadia : Gestion des réunions
- Aurelle : Organisation des tâches et dispatching(trello)
- Brice : Gestion du versionning ( contrôle, rappel et mise à jours)

Les réunions sont planifiées et structurées selon les horaires et disponibilités de chaque membre.

Pour des travaux de recherches, chaque collaborateur veille à ramener une solution et une synthèse commune est effectuée.

le choix du responsable de la présentation du coaching est effectué par tour de rôle

Aspect pratique à améliorer en équipe.

Pour que nous puissions avancer dans la réalisation de ce projet, nous devons:

- Revoir notre planning de travail
- Revoir notre méthode de travail
- Revoir l'organisation et les techniques d'apprentissages de différents langages à utiliser.

## Fonctionnement du groupe pour le projet

Jusqu'à présent, pour la réalisation de notre projet, nous faisons chaque mercredi soir une réunion afin de remédier aux remarques faites par madame Van den Schrieck lors du coaching .

Nous nous répartissons les tâches à faire avant différent coaching  
nous tablons dessus le weekend avant le coaching

Tous les mardis, veille du coaching, nous passons en revue ce qui a été fait et ce qu'il reste à faire.

Si l'un des membres lors de la réalisation de sa tâche rencontre des difficultés.  
Nous travaillons tous dessus afin de pouvoir avancer .

### update (23-06-2021)

Pour cette seconde session , nous avons adopté ( nadia et joël) une nouvelle approche de travail.

Nous avons préalablement identifié les difficultés de notre première présentation et enfin nous avons planifié nos séances de réunions. Mais aussi répartir les différentes tâches.

## Résolutions Prises par le groupe

Nous avons pris comme résolution:

\* Avoir des jours fixes qui nous permettront de tableer les états d'avancement du projet.

\* pendant les congés de juillet, août informer le membre sur l'avancée des réalisations enfin d'éviter d'éventuel conflits sur nos branches respectives.

## R19: respect du cadre légal (GDPR, propriété intellectuelle, ...)

### IMPACT DU GDPR SUR LE SITE

pour maintenir la mise en conformité avec les règles de protection des données dans notre site. nous allons procéder en plusieurs étapes:

#### 1-mise en place d'un registre de traitement de données

Dans notre registre, on créera une fiche pour chaque activité recensée, en précisant :

Les catégories de données utilisées (exemple pour la l'inscription : nom, prénom, date de naissance etc) ;

Qui a accès aux données (exemple : la directrice, le gestionnaire de site) ;

La durée de conservation de ces données (durée durant laquelle les données sont utiles d'un point de vue opérationnel, et durée de conservation en archive).

#### 2-Faire le tri dans vos données

Pour chaque fiche de registre créée, nous vérifions :

que seules les personnes habilitées ont accès aux données dont elles ont besoin ;

que les données ne seront pas conservées au-delà de ce qui est nécessaire.

#### 3-respecter les droits des personnes

informant les utilisateurs du site sur la finalité des données collectées. l'idée ici c'est donc de créer une page contenant la politique de traitement de données afin que tout le monde puisse consulter



en respectant les volontés des utilisateurs (si jamais un parent voudrais par exemple que les photos de son enfant soient supprimées du système)

4-sécuriser nos données

étant donné que le site disposera d'un espace utilisateur, on devra donc définir une politique de mots de passe robuste

on s'assurera également d'utiliser HTTPS afin de sécuriser les échanges de données au travers le réseau

limiter les droits d'accès des employés au niveau de certains types de données telles que les fiches de paie par exemple

source:<http://www.resonancecommunication.com/blog/la-rgpd-quel-est-limpact-sur-vos-sites>

[https://fr.wikipedia.org/wiki/R%C3%A8glement\\_g%C3%A9n%C3%A9ral\\_sur\\_la\\_protection\\_des\\_donn%C3%A9es](https://fr.wikipedia.org/wiki/R%C3%A8glement_g%C3%A9n%C3%A9ral_sur_la_protection_des_donn%C3%A9es)

## Conclusion

Ce projet a été très intéressant et enrichissant. Il nous a permis d'appréhender plusieurs technologies, d'aiguiser nos techniques de recherches.

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.

Néanmoins pour cette seconde session, la coordination entre nous, membres qui avons continué a été bien respectée. Nous pouvons ainsi conclure que le projet a été réalisé à 98%. On sort de ce projet avec une bonne expérience professionnelle