



Lycée CCI – Nîmes

Dossier de stage

BTS Service informatique aux organisations (2^e année)

Rapport réalisé par :

M. DELAPORTE Thomas

Dans l'entreprise :

SARL ECF BOUSCAREN

Date du début de stage : 7 janvier 2019

Date de fin de stage : 15 février 2019



Table des matières

I.	Entreprise.....	3
A.	Forme juridique.....	3
B.	Activité.....	3
II.	Contexte informatique	4
A.	Matériels.....	4
B.	Méthodes, langages, environnements, systèmes utilisés.....	4
III.	Lettre de missions	5
A.	Objectifs de l'application	5
IV.	Organisation	5
A.	Planning (Gantt)	5
B.	Outils et environnements utilisés	6
V.	Etude préalable	6
A.	Etude documents et fichiers existants.....	6
VI.	Etude détaillée.....	7
A.	Base de données	7
VII.	Réalisation	9
A.	Maquettage de l'application.....	9
B.	Frontend.....	9
C.	Backend.....	Error! Bookmark not defined.
D.	Lien entre frontend et backend	14
E.	Règles de sécurité	15
F.	Objets ou composants spécifiques.....	15
VIII.	Mise en œuvre de la solution	17
A.	Tests	17
IIX.	Analyse, critique et conclusion	17
A.	Etat actuel du projet.....	17
B.	Conclusion personnelle	17

I. Entreprise

A. Forme juridique

La société **ECF Bouscaren** est aujourd'hui juridiquement définie comme une SARL au capital de 76 224,51€.

B. Activité

ECF Bouscaren est une structure créée en 1985 et spécialisée dans la formation des permis de conduire et propose également des formations spécifiques comme les stages d'éco-conduite, de récupération de points, l'handi'conduite ou encore le perfectionnement des séniors.

Cette entreprise est située à Montpellier (34000), ou encore à Nîmes (30000) et à Lunel-Viel (34400).

II. Contexte informatique

A. Matériels

Ordinateur personnel, sous système d'exploitation Windows 10.

Le réseau interne de l'entreprise est composé de 2 serveurs distincts : un serveur permettant le stockage et le partage entre les employés et un serveur de messagerie.

L'hébergement du/des sites internet est géré par une entreprise tierce (OVH).

B. Méthodes, langages, environnements, systèmes utilisés

Front end :

- HTML, CSS
- SPA (Single page application) :
 - ReactJS

Back end :

- NodeJS
 - ApolloServer
 - GraphQL
- MySQL
 - (ORM* : Sequelize)

* Un mapping objet-relationnel (en anglais object-relational mapping ou ORM) est un type de programme informatique qui se place en interface entre un programme applicatif et une base de données relationnelle pour simuler une base de données orientée objet.

III. Lettre de missions

A. Objectifs de l'application

L'objectif de la plateforme de gestion étant de pouvoir améliorer la vitesse et la simplification de gestion des élèves, des formations, ainsi que l'emploi du temps, plusieurs systèmes sont demandés :

- Pouvoir enregistrer/supprimer/modifier des moniteurs
- Pouvoir enregistrer/supprimer/modifier des secrétaires (Administratif)
- Pouvoir enregistrer/supprimer/modifier des élèves
- Gérer les formations
- Gérer les cours des élèves (Prévisualisation planning)
- Système de mot de passe oublié

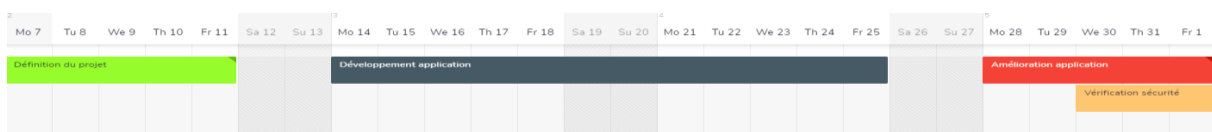
La demande porte également sur la mise en place d'une application mobile dans le futur qui sera développée plus tard après le développement complet et modification apportée à l'application développée lors de ce stage.

IV. Organisation

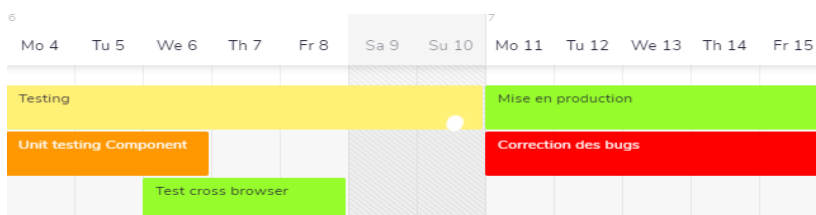
A. Planning (Gantt)

La durée du stage est de 6 semaines. (7 janvier au 15 février)

Janvier



Février



B. Outils et environnements utilisés

Environnement de développement (ou IDE) laissé au choix du développeur, soit utilisation du logiciel « **Visual studio Code** ».

- *Environnement* : Webpack dev server (Serveur simple http)
- *Outils* : phpmyadmin

V. Etude préalable

A. Etude documents et fichiers existants

Les documents fournis sont :

- Le logo
- Liste des formations sous format .doc
- Liste des moniteurs

Le format de la table « Formations » a été fourni à l'avance par l'administration de l'ECF :

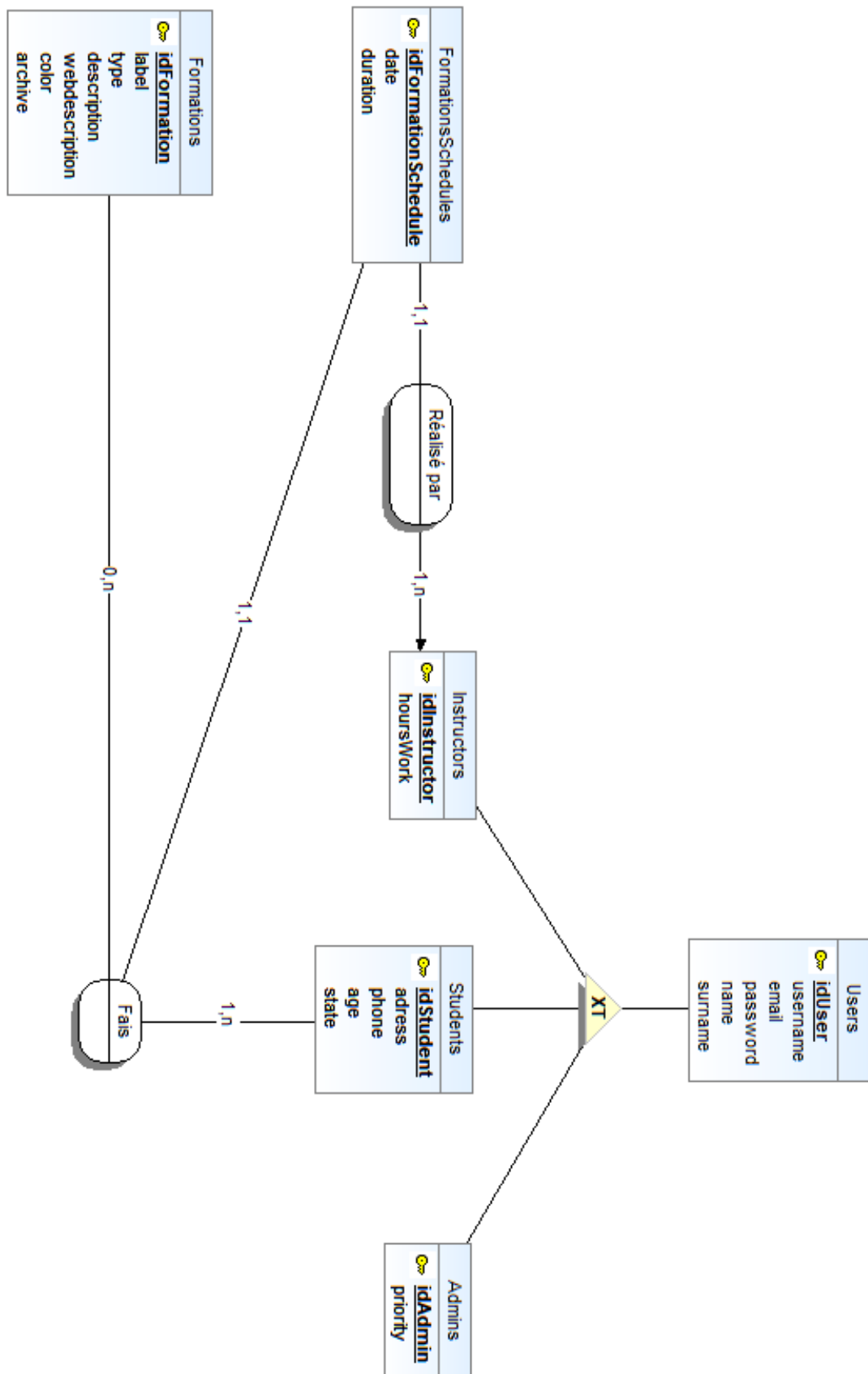
- Label : (Permis B, Permis B conduite accompagnée).
- Type : (Voiture, Code, Deux-roues et plus).
- Description
- Web présentation : Présentation en HTML pour le site principal
- Couleur : Couleur représentant la formation
- Archivé : Etat d'archivage (0 ou 1 si archivé)

VI. Etude détaillée

A. Base de données

Règles de gestion

- Un utilisateur est défini par un nom, un prénom, un login, un mot de passe et un mail.
- Un étudiant, administrateur, moniteur hérite d'utilisateur
- Un étudiant possède en plus une adresse, un numéro de téléphone, un âge.
- Un étudiant peut être inscrit à plusieurs formations
- Une formation est définie par :
 - Libelle
 - Type (Voiture, Code, Deux-roues et plus, Stage accéléré, Conduire dès 14 ans, Remorque, Caravane et Camping-Car, Bateau, Stage récupération de points, Formations professionnelles)
 - Description
 - Web Description
 - Couleur
 - Etat d'archivage
- Un étudiant peut réaliser des cours avec un moniteur pour chacune de ces formations.
 - Un cours possède une date de début, le total d'heures

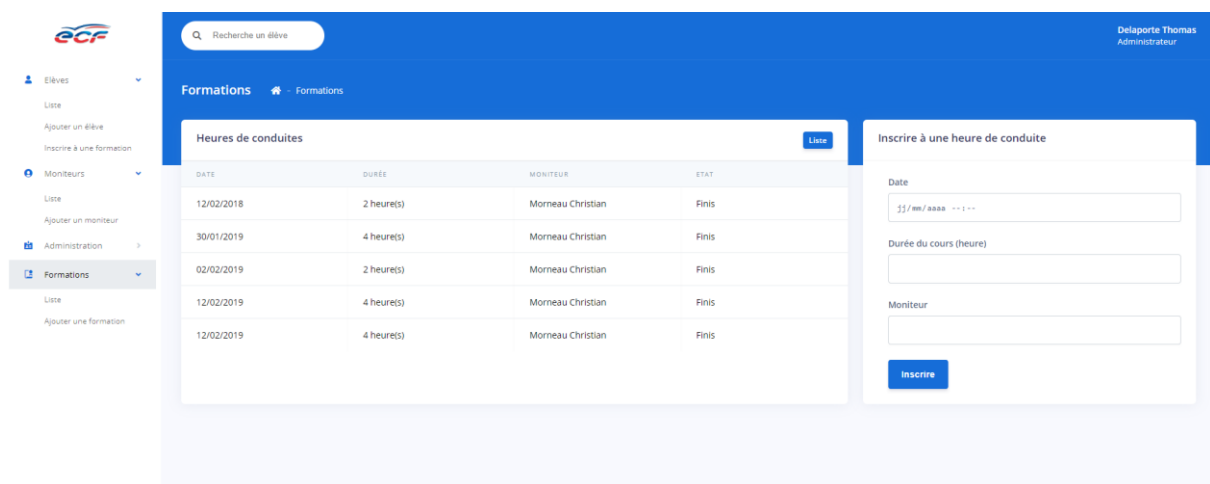


VII. Réalisation

A. Maquettage de l'application



Design de l'application



B. Frontend

« **ReactJS** est une librairie JavaScript, qui permet véritablement de révolutionner le développement des interfaces pour vos applications web »

Organisation des fichiers

- **Assets** (*image, style*)

- Components
 - Les composants permettent de diviser l'interface en éléments indépendants réutilisables et de considérer chaque élément de manière isolée.
- Core
 - Il s'agit de l'ensemble de méthodes utilitaires, de fichiers de configuration (constantes)
- Pages
 - Il s'agit de composants représentant chaque page de l'application, avec toute la partie gestion des données avec le serveur.

Exemple de composant général

```
1. class Composant extends React.Component {
2.   render() {
3.     return <h1>Bonjour, {this.props.name}</h1>;
4.   }
5. }
```

```
1. function App() {
2.   return (
3.     <div>
4.       <Welcome name="Thomas" />
5.       <Welcome name="Leo" />
6.       <Welcome name="Jose" />
7.     </div>
8.   );
9. }
10.
11. ReactDOM.render(
12.   <App />, document.getElementById('root')
13. );
```

Code disponible au lien : <https://codesandbox.io/s/03wr2346vv>

Résultat

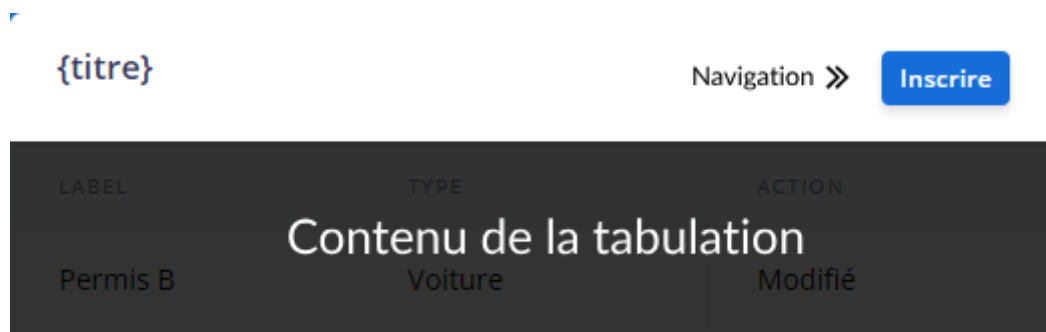
Bonjour, Thomas

Bonjour, Leo

Bonjour, Jose

Exemple composant dans l'application

Une **card** est un composant permettant un système de tabulation.

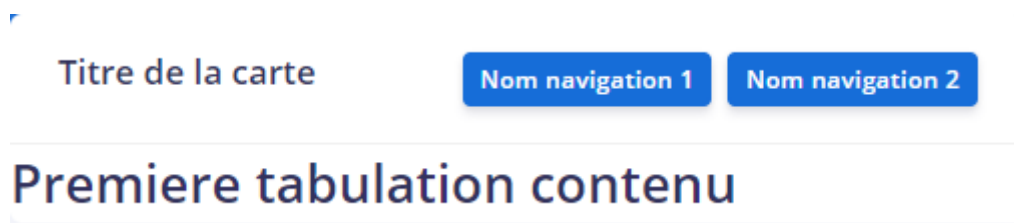


```

1. <Card
2.   title="Titre de la carte"
3.   tabs={{
4.     premieretab: {
5.       name: 'Nom navigation 1',
6.       component: (props) => (
7.         <h1>Premiere tabulation contenu</h1>
8.       )
9.     },
10.
11.     secondtab: {
12.       name: 'Nom navigation 2',
13.       component: (props) => (
14.         <h2>Seconde tabulation contenu</h2>
15.       )
16.     }
17.   }} />

```

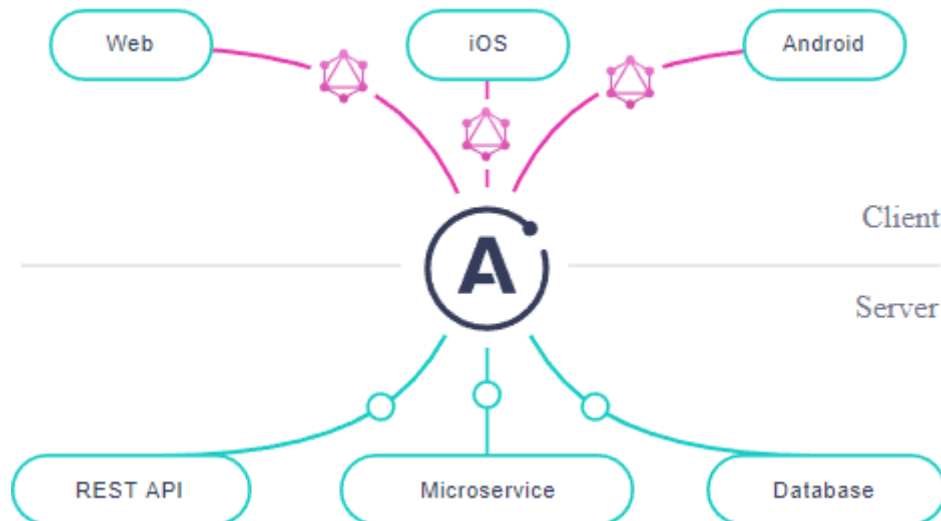
Résultat



C. Back end

Le serveur est sous **NodeJS**. J'utilise une librairie « **ApolloServer** » permettant la création simplifiée d'un serveur utilisant **GraphQL** ainsi que d'outils pour faciliter le développement.

Le serveur est en soit une API faisant le lien entre le client et la/les bases de données.



La base de données est sous **MySQL** et j'utilise un ORM nommé Sequelize permettant la simplification des requêtes SQL sous forme de fonction ou méthode.

Exemple de création de table (sous Sequelize)

```
1. export default (database, Sequelize) => {
2.
3.   const formationSchema = database.define('formation', {
4.     idFormation: {
5.       type: Sequelize.INTEGER,
6.       primaryKey: true,
7.       autoIncrement: true
8.     },
9.     label: Sequelize.STRING,
10.    type: Sequelize.STRING,
11.    description: {
12.      type: Sequelize.STRING,
13.      defaultValue: ''
14.    },
15.    webdescription: {
16.      type: Sequelize.STRING,
17.      defaultValue: ''
18.    },
19.    color: Sequelize.STRING,
20.    archive: {
21.      type: Sequelize.BOOLEAN,
22.      defaultValue: false
23.    }
24.  });
25.
26.  formationSchema.associate = (models) => {
27.
```

```

28.     formationSchema.hasMany(models.formationStudent, {
29.         foreignKey: 'idFormation'
30.     });
31. }
32.
33. return formationSchema;
34. }

```

Définition du type (GraphQL)

```

1. graphql`
2.
3.     type Formation {
4.         idFormation: ID!
5.         label: String!
6.         type: String!
7.         description: String
8.         webdescription: String
9.         color: String
10.    }
11.
12.    extend type Query {
13.        getFormationTypes: [String!]
14.        getFormation(id: ID!): Formation
15.        formations: [Formation!]!
16.    }
17.
18.    extend type Mutation {
19.        createFormation(label: String!, type: String!, description: String!, webdescription: String, color: String!): Formation
20.        updateFormation(id: ID!, label: String, type: String, description: String, webdescription: String): Formation
21.        destroyFormation(id: ID!): ID
22.    }
23.
24. `;

```

Récupération de données (GraphQL) : requête SELECT

```

1. graphql`
2.     query getFormation($id: ID!){
3.         getFormation(id: $id){
4.             idFormation
5.             label
6.             type
7.             color
8.             description
9.             webdescription
10.        }
11.    }
12. `;

```

```

1. graphql`
2.     {
3.         formations {
4.             idFormation
5.             label
6.             type
7.             color
8.             description
9.             webdescription
10.        }
11.    }
12. `;

```

Requête création (via GraphQL) : mutation (INSERT, UPDATE, DELETE)

```
1. gql`
2.   mutation createFormation($label: String!, $type: String!, $description: String!,
3.     $webdescription: String!, $color: String!){
4.     createFormation(label: $label, type: $type, description: $description, webde
5.       scription: $webdescription, color: $color){
6.         idFormation
7.         label
8.         type
9.         color
10.        description
11.        webdescription
12.      }
13.    }
14. `;
```

D. Lien entre frontend et backend

Apollo propose une librairie pour le côté client nommé « **ApolloClient** » spécialement pour ReactJS permettant d'exécuter des requêtes spécifiques simplement, avec une gestion d'erreur.

Pour récupérer des données :

```
1. <Query query={FORMATIONS}>
2.   {({ loading, error, data }) => {
3.
4.     if (loading) return <div>Chargement des données</div>
5.     if (error) return <div>Erreur</div>
6.
7.     return (...);
8.   }}
9. </Query>
```

Le composant *Query* retourne une variable « data » qui est un tableau contenant toutes les données envoyées par le serveur.

Pour exécuter une fonction/méthode :

```
1. <Mutation mutation={CREATE_FORMATION}>
2.   {(createFormation) => {
3.
4.     return (...);
5.   }}
6. </Mutation>
```

Le composant *Mutation* retourne une variable « createFormation » dans ce cas qui représente une fonction qui lors de l'exécution de celle-ci va envoyer au serveur la requête de création de formation avec les données adéquates.

Par exemple :

```
1. createFormation({
2.   variables: {
3.     label: "Permis B"
4.     type: "Voiture",
5.     description: "Description permis B",
6.     webdescription: "<b>Description</b>",
7.     color: "#fff"
8.   }
9. });
```

E. Règles de sécurité

L'administration proposée permet aux administrateurs de seulement se connecter. Chaque administrateur possède un champ « *priority* » pour permettre à l'avenir de mettre des droits plus précis pour chaque module de l'application.

Les mots de passe dans la base de données sont hachés à l'aide de la librairie « bcrypt ».

Lors de la connexion, le serveur vérifie les éléments transmis (login et mot de passe) puis si les données sont correctes, le serveur retourne un « token » qui est stocké dans les données locales du navigateur pour le site : « localStorage ».

A chaque chargement de page et à chaque requête, le serveur vérifie que le token est toujours valide.

Le mot de passe de chaque utilisateur n'est pas choisi à l'avance par ceux-ci. Ils sont générés par le serveur puis envoyés par mail à l'utilisateur.

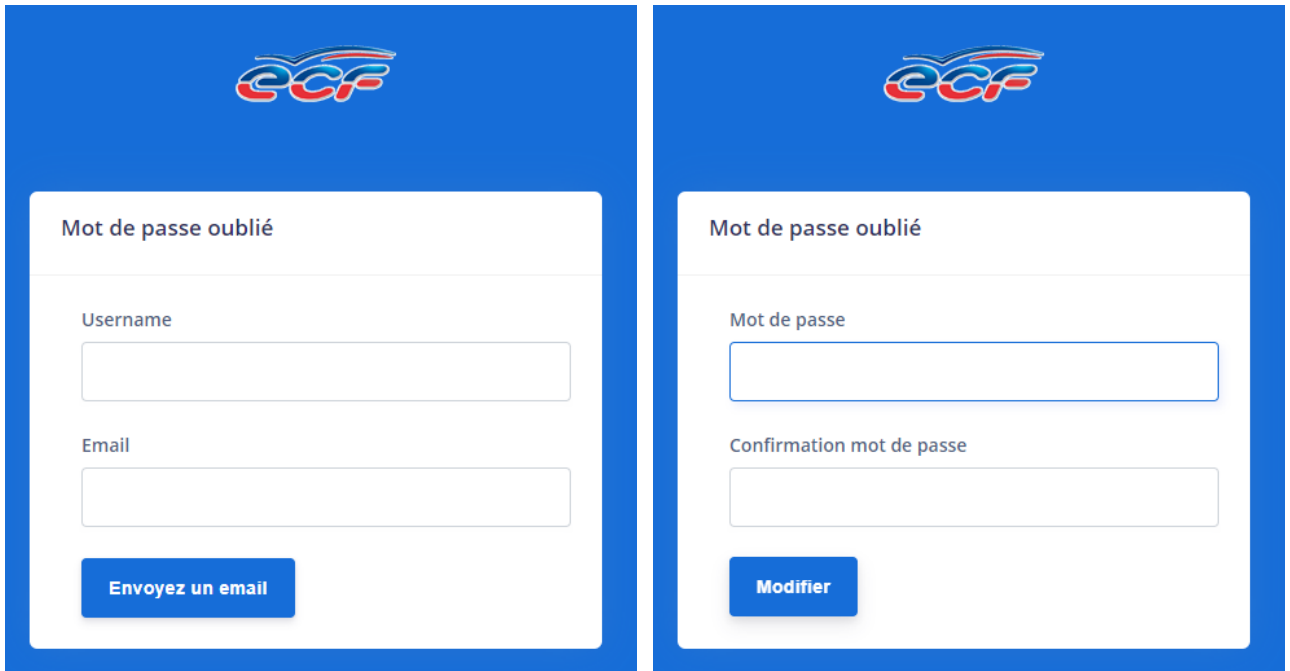
Exemple de mot de passe : « 2wzkfclh », « 88e26xvg »

F. Objets ou composants spécifiques

Un système de changement de mot de passe a été mis en place au cas où.

L'administrateur rentre son login et son mail, s'ils correspondent le serveur envoie un mail avec un lien spécifique pour modifier son mot de passe.

Interface



The image displays two side-by-side screenshots of a web interface for password management, featuring the ECF logo at the top of each panel.

Left Panel: Mot de passe oublié

- Form title: Mot de passe oublié
- Fields: Username, Email
- Button: Envoyez un email

Right Panel: Mot de passe oublié

- Form title: Mot de passe oublié
- Fields: Mot de passe, Confirmation mot de passe
- Button: Modifier

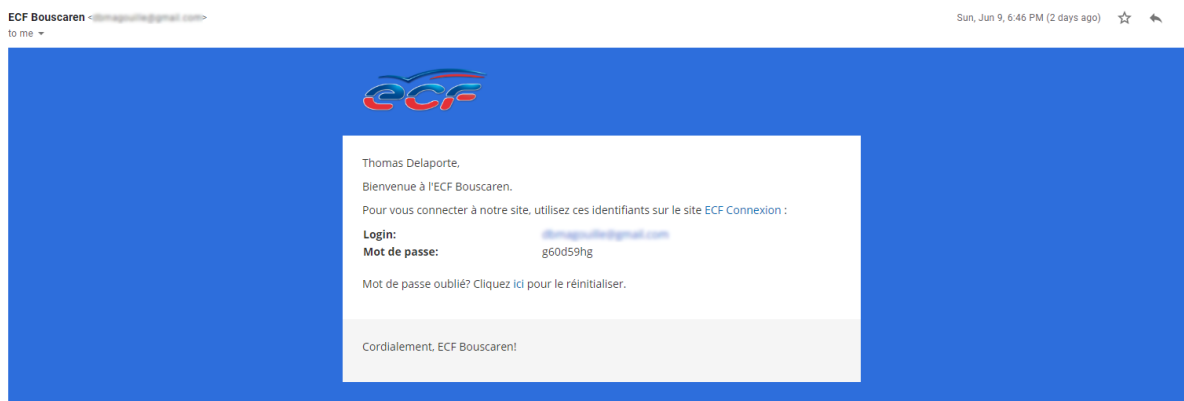
Un lien par exemple ressemble à « [http://\[site\].fr/forgot-password/eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9#eyJ0b2tIbWl6Imo4Zmk0ZzNoZ3kiLCJpZCI6NywiaWF0IjoxNTU4OTU4NDc4LCJleHAiOjE1NTg5NTkzNzh9%23cR3LQy_JvE_R50bUSNArbsplhn0cpstdLSeQSKcRdNM](http://[site].fr/forgot-password/eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9#eyJ0b2tIbWl6Imo4Zmk0ZzNoZ3kiLCJpZCI6NywiaWF0IjoxNTU4OTU4NDc4LCJleHAiOjE1NTg5NTkzNzh9%23cR3LQy_JvE_R50bUSNArbsplhn0cpstdLSeQSKcRdNM) »

Le lien pour le changement de mot de passe n'est valable qu'une seule fois et ne fonctionne que pendant les 15 minutes après la demande.

Système d'email

Lors de la création d'un compte élève, moniteur, ou administrateur, le système envoie un mail de confirmation de création de compte avec les paramètres tels que le login et le mot de passe.

Le système de mailing se base sur nodemailer qui permet de simplement envoyer des mails aux utilisateurs. Une template a été créée pour l'ensemble des mails de l'ECF Bouscaren respectant les normes.



VIII. Mise en œuvre de la solution

A. Tests

Chaque composant a été testé individuellement pour mettre en évidence des erreurs.

Nous avons procédé à une mise en production pendant une semaine pour présenter la nouvelle application à l'administration qui ont pu mettre en valeur quelques problèmes qui ont pu être corrigés directement.

IIIX. Analyse, critique et conclusion

A. Etat actuel du projet

Le projet est fonctionnel et répond bien aux demandes du cahier des charges, disponible au niveau local.

Il reste cependant des modules à développer pour améliorer l'application, et développer une application Android et/ou iOS pour les élèves afin d'accéder à leur emploi du temps simplement ainsi qu'à la liste des formations.

B. Conclusion personnelle

Ce stage m'a permis d'acquérir de nombreuses connaissances quant à la création d'un site web avec de nouvelles technologies ainsi qu'à développer mes compétences en mise en production du site internet avec la mise en place du site internet dans les locaux.

Puis dans un second temps, j'ai pu contribuer au bon fonctionnement de l'entreprise avec la gestion des postes informatiques.

Je tiens à remercier mon maitre de stage Mr BOUSCAREN, pour son accueil, son aide à l'intégration dans l'entreprise et l'aide pour la gestion du projet. Grâce à sa confiance, j'ai pu accomplir parfaitement mes missions.

Je remercie également toute l'équipe secrétariat pour leurs opinions et leur aide pour améliorer chaque module afin de répondre plus simplement aux besoins de l'entreprise.