



Ministère de l'Enseignement Supérieur et de la recherche scientifique
Direction Générale des Etudes Technologiques
Institut Supérieur des Etudes Technologiques de Sfax
Département Technologies de l'Informatique



STAGE DE PERFECTIONNEMENT

Nom et Prénom : **Guidara Najmeddine**

Groupe : **DSI21**

N° CIN / Passeport : **11162555**

Organisme d'accueil : **Société SIFAST**

Période du stage : **Du 09/01/2023 Au 04/02/2023**

Année Universitaire : **2022/2023**

Sommaire

Introduction générale.....	1
CHAPITRE 1 : Etude de l'organisme	3
1. Introduction	4
2. Présentation de l'organisme	4
3. Organigramme de l'organisme	5
4. Conclusion.....	5
CHAPITRE 2 : Gestion de version avec Git et GitHub.....	6
1. Introduction	7
2. Logiciels utilisés.....	7
2.1. Git.....	7
2.2. GitHub	11
CHAPITRE 3 : Modélisation conceptuelle	14
1. Introduction	15
2. Diagramme de cas d'utilisation.....	15
3. Diagramme de séquence.....	15
4. Conclusion.....	17
CHAPITRE 4 : Etude de cas	18
1. Introduction	19
2. Réalisation	19
2.1. Composant de création d'objet	21
2.2. Composant pour la liste des objets	23
2.3. Composant pour les détails de l'objet	24
2.4. Ajouter un CSS pour les composants React Typescript	26
3. La réalisation finale	27
4. Conclusion.....	28
Conclusion générale	29

Liste des figures

Figure 1- 1 : Logo de l'entreprise.....	4
Figure 2- 1 : Logo Git.....	7
Figure 2- 2 : Installation du Git	7
Figure 2- 3 : Git config.....	8
Figure 2- 4 : Git init, Git status, Git add.....	9
Figure 2- 5 : Git commit.....	10
Figure 2- 6 : Git log	10
Figure 2- 7 : Git tag	11
Figure 2- 8 : Logo de GitHub.....	11
Figure 2- 9 : Création compte sur GitHub.....	11
Figure 2- 10 : L'utilisation de Gist.....	12
Figure 2- 11 : Avant la création de notre repository	12
Figure 2- 12 : Après la création de notre repository.....	13
Figure 2- 13 : Création d'un dépôt sur GitHub	13
Figure 3- 1 : Diagramme de cas d'utilisation	15
Figure 3- 2 : Diagramme de séquence (Ajoutassions).....	16
Figure 3- 3 : Diagramme de séquence (Modification)	16
Figure 3- 4 : Diagramme de séquence (Suppression).....	17
Figure 3- 5 : Diagramme de séquence (Supprimez tout).....	17
Figure 4- 1 : La page App	19
Figure 4- 2 : Définie le type de données.....	20
Figure 4- 3 : Création un service de données	20
Figure 4- 4 : Intégration Firebase	21
Figure 4- 5 : Composant de création d'objet.....	21
Figure 4- 6 : Partie formulaire du composant de création d'objet.....	22
Figure 4- 7 : Composant pour la liste des objets	24
Figure 4- 8 : Composant pour les détails de l'objet.....	26
Figure 4- 9 : L'interface CSS	26
Figure 4- 10 : L'interface d'ajouter tutoriel	27
Figure 4- 11 : L'interface de la liste tutoriel.....	28

Remerciements

Tout d'abord, je tiens à exprimer mes sincères remerciements à Monsieur **Mohamed Bouaziz**, chef du projet au sein du département web et graphique de la société « **SIFAST** »

Qui m'a honoré en acceptant de m'encadrer durant toute la période de stage. Je souhaite également remercier chaleureusement tout le personnel pour leur accueil et leur patience, ainsi que pour leurs précieux conseils.

Ensuite, j'aimerais adresser mes profonds remerciements aux responsables et aux enseignants de l'**Institut Supérieur des Études Technologiques de Sfax** pour la qualité de l'enseignement offert et le soutien de l'équipe administrative.

Enfin, je tiens à exprimer ma reconnaissance envers tous ceux qui m'ont aidé à réaliser mon stage dans les meilleures conditions et à élaborer ce rapport modeste.



Introduction

générale



Introduction Générale

Dans le but de perfectionner mes connaissances et mes compétences, j'ai été accueilli dans les locaux de la société « **SIFAST** ». On m'a alors proposé de travailler sur la conception et la création d'une application web. J'ai ainsi développé une application React Typescript Firebase CRUD, en utilisant la bibliothèque Firebase. Cette application permet de gérer des tutoriels, chacun étant identifié par une clé, un titre et une description. Les opérations CRUD (création, mise à jour et suppression) sont effectuées via Firebase.

Ce stage a été une véritable opportunité pour mettre en pratique mes connaissances et compétences dans un environnement professionnel et concret.



CHAPITRE 1 :

Etude de l'organisme



1. Introduction

Dans ce chapitre, nous allons présenter la société dans laquelle nous avons effectué notre stage ainsi que son organigramme.

2. Présentation de l'organisme

SIFAST, créée en 2010 par des professionnels du service et de l'informatique, basée en **Tunisie** et implantée en **France**.

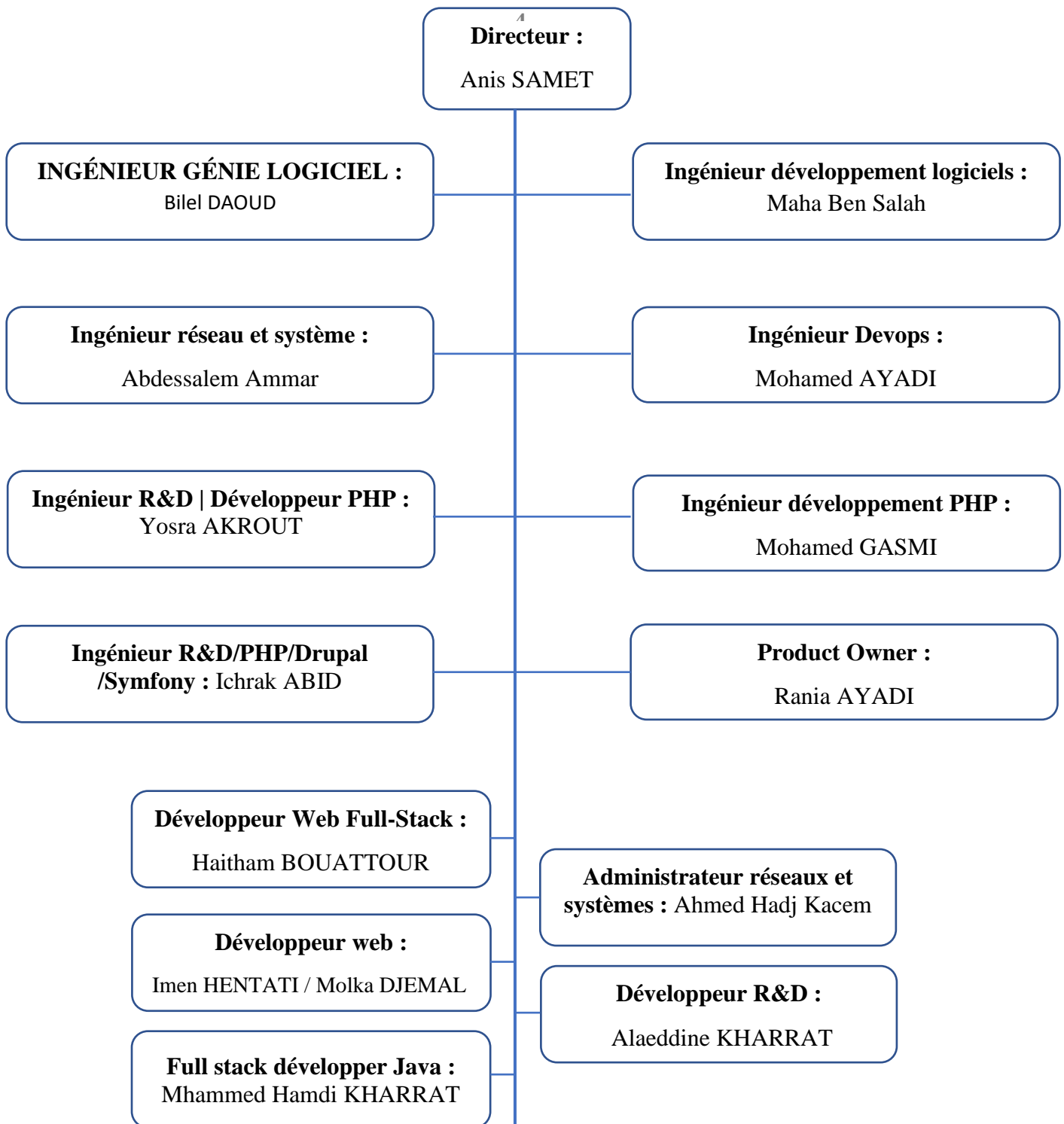
SIFAST compte une cinquantaine de collaborateurs avec **80% d'ingénieurs**, et est reconnue comme une entreprise **formatrice** ce qui lui a permis de tisser un réseau de recrutement fiable et de mettre en place plusieurs partenariats avec les écoles d'ingénieurs en Tunisie.

Tous nos collaborateurs appliquent l'approche Agile (Scrum) pour mieux coller aux besoins de nos clients et DevOps pour **automatiser** et **sécuriser** notre chaîne de développement.



Figure 1- 1 : Logo de l'entreprise

3. Organigramme de l'organisme



4. Conclusion

A travers ce chapitre, nous avons présenté le cadre du travail.



CHAPITRE 2 :

Gestion de version avec

Git et GitHub



1. Introduction

Avant de commencer le travail, nous présentons la gestion de version avec **Git** et **GitHub** rencontrées durant la période du stage.

2. Logiciels utilisés

2.1. Git

Git est un logiciel VCS (Version Control System) local qui permet aux professionnels de la data et aux développeurs de sauvegarder l'historique de modifications de leurs projets.



Figure 2- 1 : Logo Git

- Installation du git pour utiliser quelque commande.

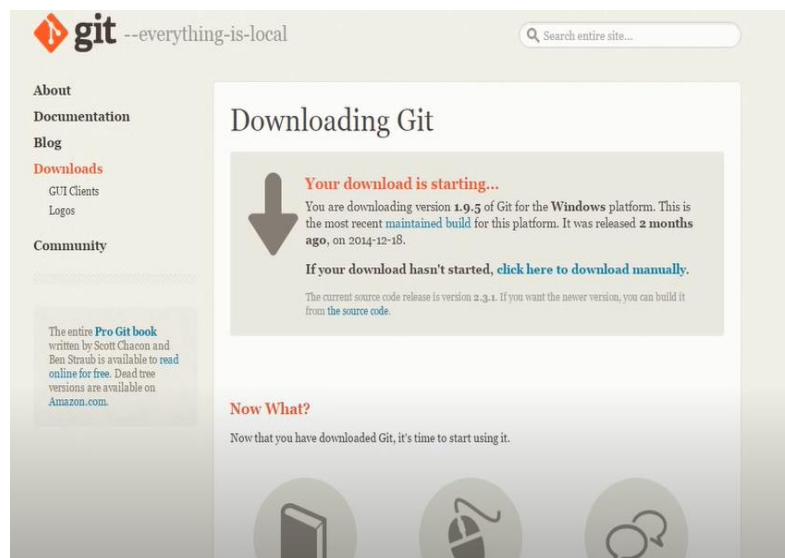


Figure 2- 2 : Installation du Git

2.1.1. Commandes GIT de base

❖ Git config

L'une des commandes git les plus utilisées est **git config**. On l'utilise pour configurer les préférences de l'utilisateur : son mail, l'algorithme utilisé pour diff, le nom d'utilisateur et le format de fichier.

Par exemple, la commande suivante peut être utilisée pour définir le mail et Name d'un Utilisateur :



```
C:\Users\DELL>cd C:\Users\DELL\3T
C:\Users\DELL\3T>cd .\mon-premier-projet-git
C:\Users\DELL\3T\mon-premier-projet-git>git help config
C:\Users\DELL\3T\mon-premier-projet-git>git config --global user.email "najmeddineguidara71@gmail.com"
C:\Users\DELL\3T\mon-premier-projet-git>git config --global user.name "ahmed"
C:\Users\DELL\3T\mon-premier-projet-git>git config --global color.ui true
```

Figure 2- 3 : Git config


❖ Git init, Git status, Git add

La **commande git init** est utilisée pour créer un nouveau dépôt.

La **commande git status** affiche la liste des fichiers modifiés ainsi que les fichiers qui doivent encore être ajoutés ou validés.

La **commande git add** peut être utilisée pour ajouter des fichiers à l'index.

Par exemple, la commande suivante ajoutera un fichier nommé **najm**.

 Invite de commandes

```
'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.

C:\Users\DELL>cd C:\Users\DELL\.3T\mon-premier-projet-git

C:\Users\DELL\.3T\mon-premier-projet-git>git init
Reinitialized existing Git repository in C:/Users/DELL/.3T/mon-premier-projet-git/.git/

C:\Users\DELL\.3T\mon-premier-projet-git>touch najm.html

C:\Users\DELL\.3T\mon-premier-projet-git>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   readme.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        najm.html

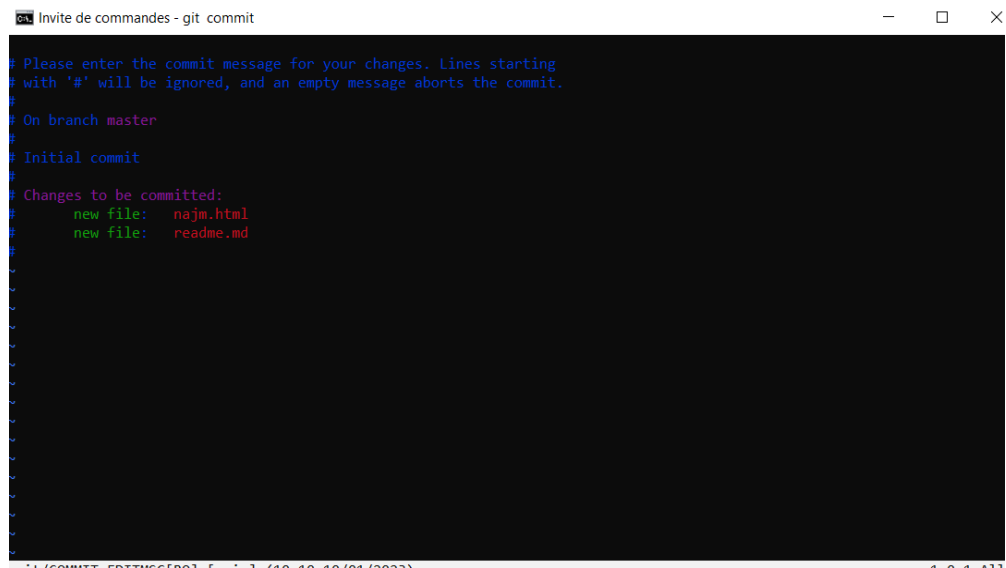
C:\Users\DELL\.3T\mon-premier-projet-git>git add najm.html

C:\Users\DELL\.3T\mon-premier-projet-git>
```

Figure 2- 4 : Git init, Git status, Git add

❖ **Git commit**

La commande **git commit** permet de **valider les modifications**.



```

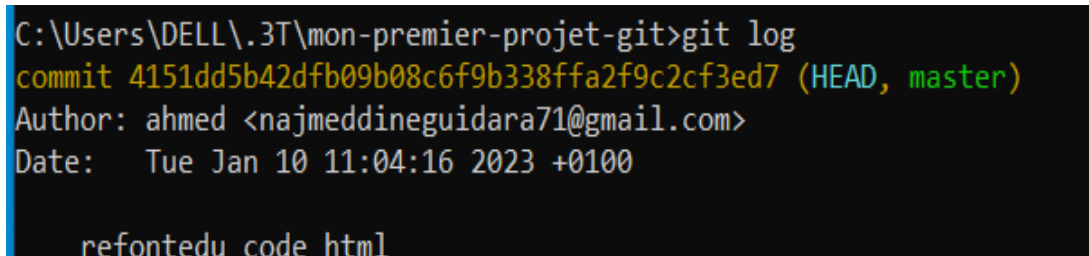
Invite de commandes - git commit
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   new file:   najm.html
#   new file:   readme.md
#
#
# Please enter the commit message for your changes (press Ctrl+C to abort):

```

Figure 2- 5 : Git commit

❖ **Git log**

Pour voir l'historique des commits, ont utilisé cette commande :



```

C:\Users\DELL\3T\mon-premier-projet-git>git log
commit 4151dd5b42dfb09b08c6f9b338ffa2f9c2cf3ed7 (HEAD, master)
Author: ahmed <najmeddineguidara71@gmail.com>
Date: Tue Jan 10 11:04:16 2023 +0100

    refontedu code html

```

Figure 2- 6 : Git log

Pour afficher un seul commit, utilisez la **commande git log -n**.

❖ **Git checkout**

La commande **git checkout** permet de passer d'une branche a une autre.

❖ **Git tag**

Les tags sont généralement utilisés pour capturer un point de l'historique utilisé pour une version marquée (first commit).

```

C:\Users\DELL\3T\mon-premier-projet-git>git tag first-commit

C:\Users\DELL\3T\mon-premier-projet-git>git log
commit 4151dd5b42dfb09b08c6f9b338ffa2f9c2cf3ed7 (HEAD, tag: first-commit, master)
Author: ahmed <najmeddineguidara71@gmail.com>
Date: Tue Jan 10 11:04:16 2023 +0100

    refontedu code html

```

Figure 2- 7 : Git tag

2.2. GitHub

GitHub est un service en ligne qui permet d'héberger des dépôts ou repot Git. C'est le plus grand hébergeur de dépôts Git du monde.



Figure 2- 8 : Logo de GitHub

- Création compte sur GitHub

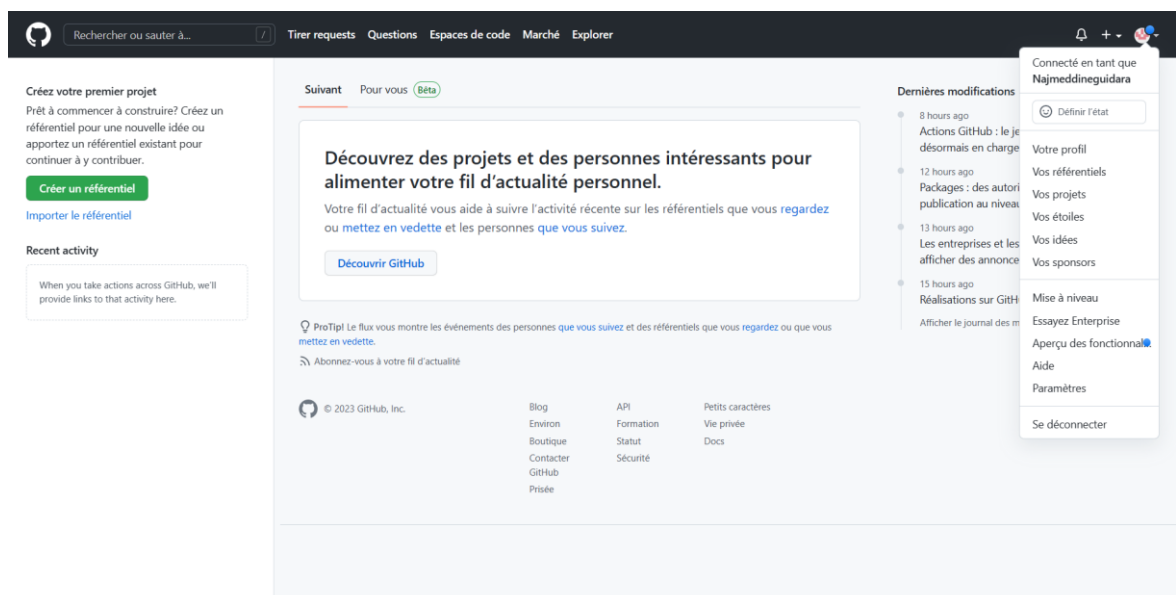


Figure 2- 9 : Création compte sur GitHub

Gist est un service exceptionnel fourni par GitHub. C'est un moyen simple **de partager** en public ou en privé des extraits de code, de notes, de listes de tâches et etc.

Comment utiliser Gist ?

Pour utiliser Gist, il faut se rendre sur la plateforme de GitHub <https://github.com>, après avoir été connecté, cliquez sur le bouton + et choisissez sur **New Gist** comme le montre l'image ci-dessous.

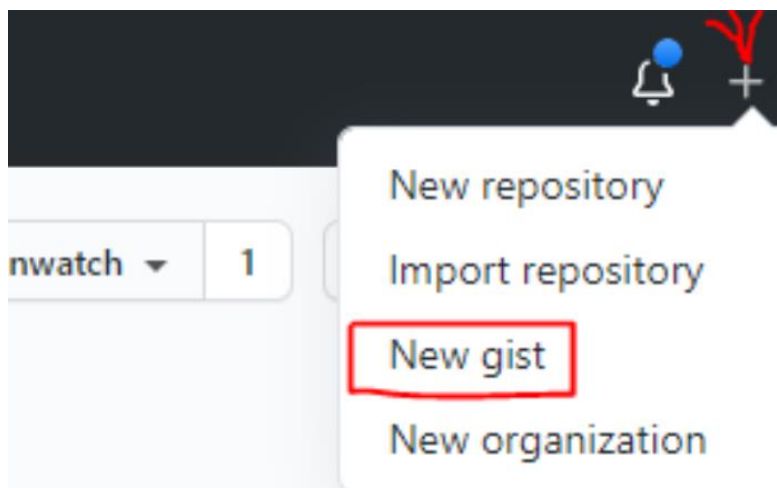


Figure 2- 10 : L'utilisation de Gist

- **Avant la création de notre repository**

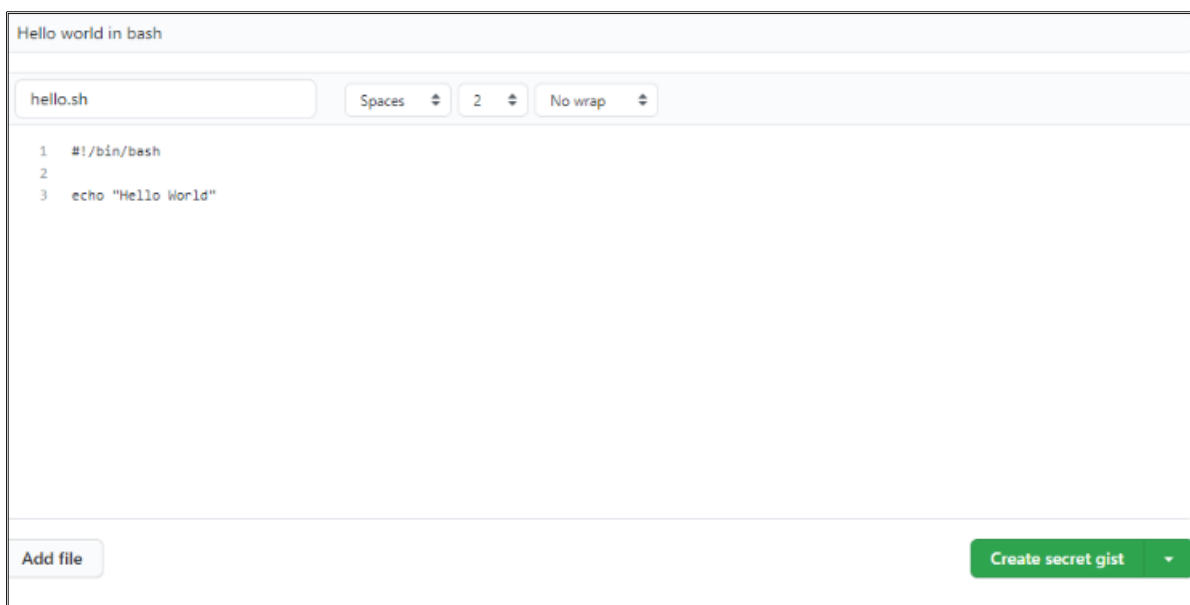


Figure 2- 11 : Avant la création de notre repository

- **Après la création de notre repository**



Figure 2- 12 : Après la création de notre repository

Créer un dépôt sur GitHub

Cliquer sur le bouton avec l'icône du plus sur la barre de navigation avant ta photo de profil puis sur New repository.



Figure 2- 13 : Création d'un dépôt sur GitHub

Question : Utilisez GitHub Issues pour suivre des idées, des commentaires, des tâches ou des bogues pour le travail sur GitHub.

Demande d'extraction : Proposer et vérifier les modifications avant d'effectuer une fusion sur GitHub.



CHAPITRE 3 :

Modélisation conceptuelle



1. Introduction

Pendant la période de stage j'ai effectué une étude du système existant afin de créer un site web désiré.

2. Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation est un outil important dans le domaine de l'analyse et de la conception des systèmes informatiques. Il permet de représenter graphiquement les interactions entre les différents acteurs et le système, en décrivant les services offerts par ce dernier.

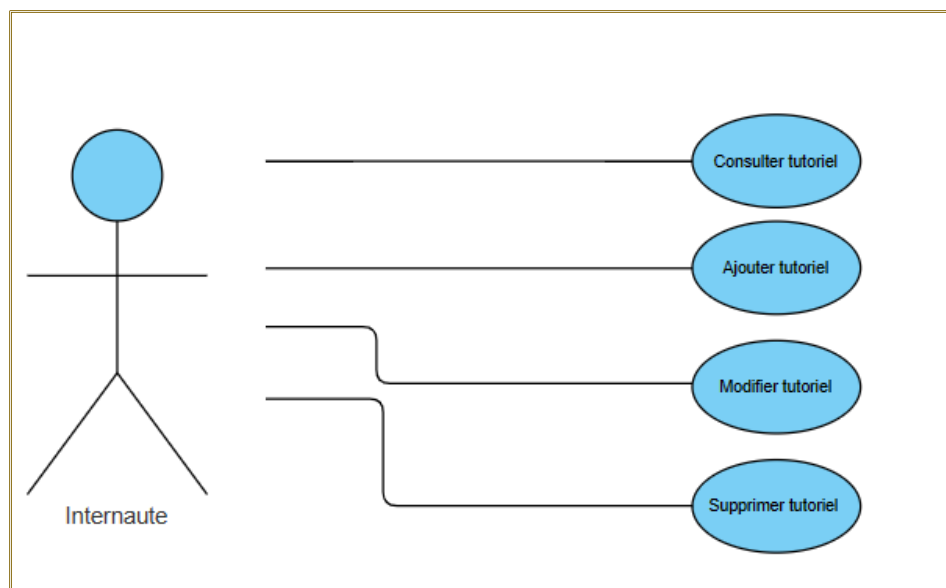


Figure 3- 1 : Diagramme de cas d'utilisation

3. Diagramme de séquence

Un diagramme de séquence est une représentation graphique qui décrit l'interaction entre différents objets ou composants d'un système logiciel, notamment pour les opérations CRUD (Create, Read, Update, Delete) qui permettent la manipulation de données.

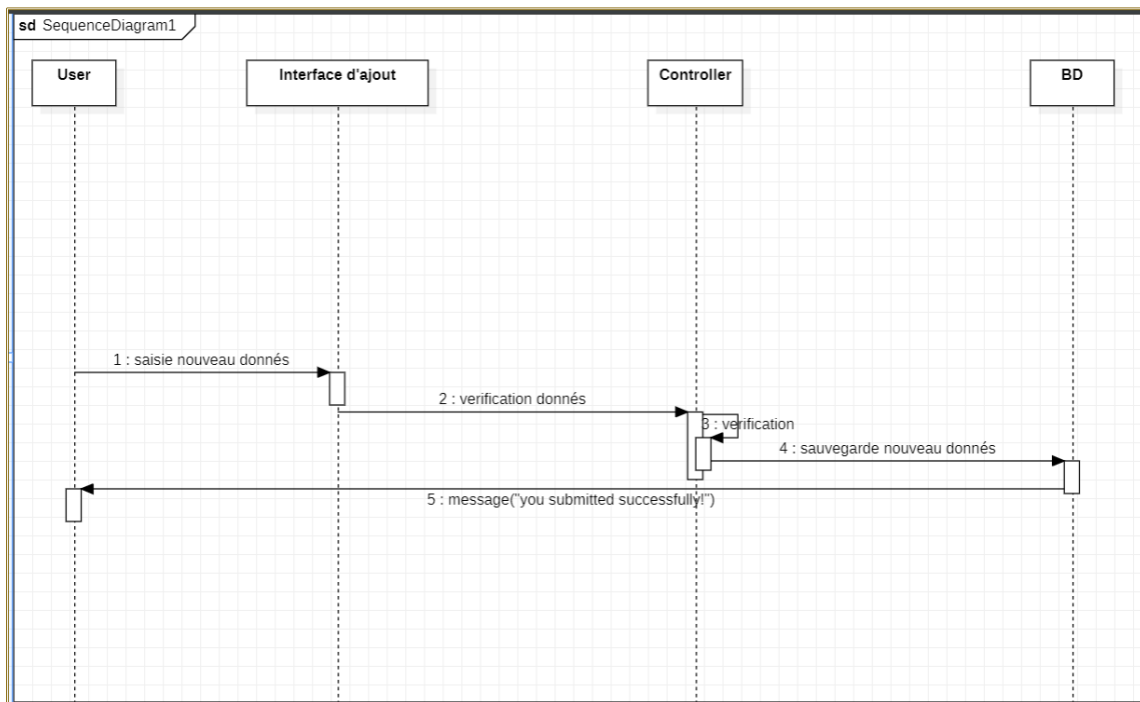


Figure 3- 2 : Diagramme de séquence (Ajoutassions)

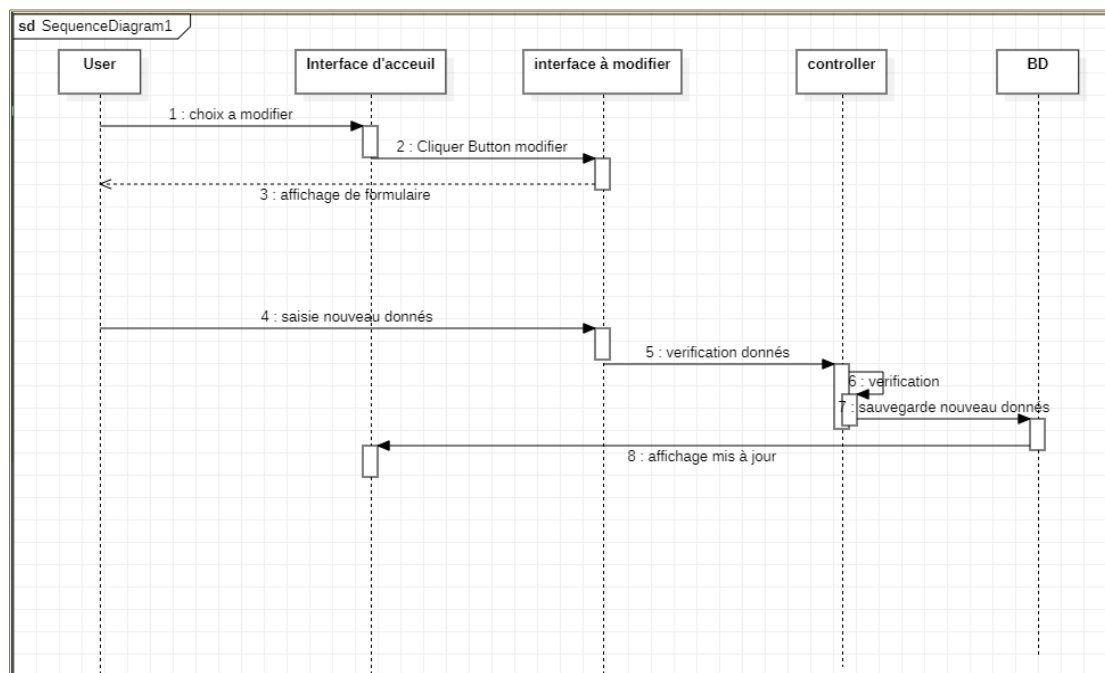


Figure 3- 3 : Diagramme de séquence (Modification)

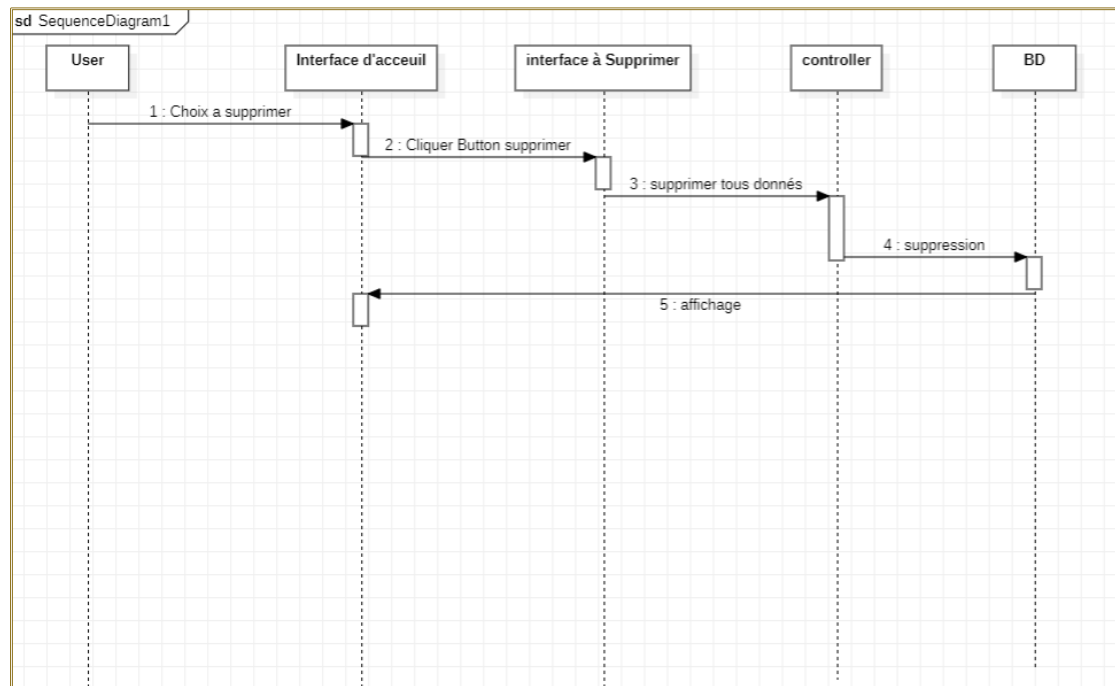


Figure 3- 4 : Diagramme de séquence (Suppression)

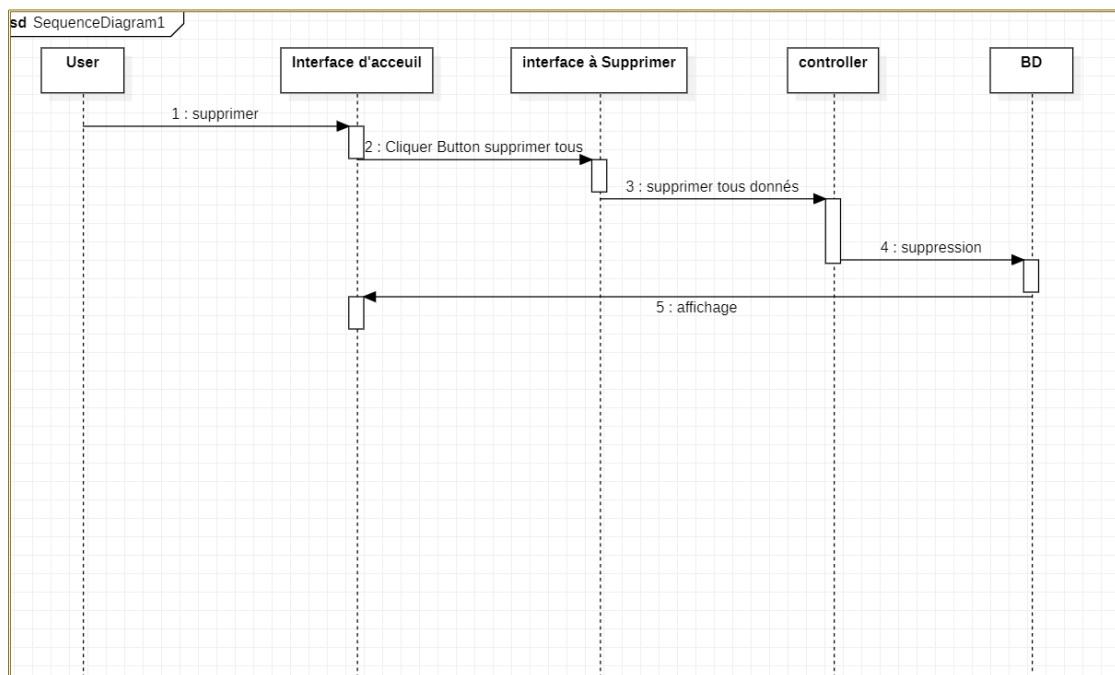


Figure 3- 5 : Diagramme de séquence (Supprimez tout)

4. Conclusion

Dans ce chapitre, nous avons présenté une étude du système existant, incluant le diagramme de cas d'utilisation et le diagramme de séquence. Ces outils sont essentiels pour faciliter le travail à réaliser.



CHAPITRE 4 :

Etude de cas



1. Introduction

Après avoir placé notre projet dans son cadre générale, nous allons présenter le processus de sa réalisation.

2. Réalisation

La partie Frontend :

Ce composant est le conteneur racine de notre application **App.js** :

```
import { Component } from "react";
import { Switch, Route, Link } from "react-router-dom";
import "bootstrap/dist/css/bootstrap.min.css";
import './App.css';
import AddTutorial from "../components/add-tutorial.component";
import TutorialList from "../components/tutorials-list.component";

class App extends Component {
  render() {
    return (
      <div>
        <nav className="navbar navbar-expand navbar-dark bg-dark">
          <div className="navbar-nav mr-auto">
            <li className="nav-item">
              <Link to={"/tutorials"} className="nav-link">Tutorial</Link>
            </li>
            <li className="nav-item">
              <Link to={"/add"} className="nav-link">
                Ajouter
              </Link>
            </li>
          </div>
        </nav>
        <div className="container mt-3">
          <h2>React Typescript Firebase example</h2>
          <Switch>
            <Route exact path={["/", "/tutorials"]} component={TutorialList} />
            <Route exact path="/add" component={AddTutorial} />
          </Switch>
        </div>
      </div>
    );
  }
}
export default App;
```

Figure 4- 1 : La page App

Nous devons maintenant définir le type de données :

```
// définir type base donnée
export default interface ITutorialData {
  key?: string | null,
  title: string,
  description: string,
}
```

Figure 4- 2 : Définie le type de données

- Ce service utilisera le service Firebase pour interagir avec Firebase.
- Il contient les fonctions nécessaires aux opérations CRUD.
- On importe la bibliothèque et on ajoute la configuration que nous avons enregistrée lorsque la fenêtre contextuelle s'est affichée.

```
import firebase from "../firebase";
import ITutorialData from "../types/tutorial.type";

const db = firebase.ref("/tutorials");

class TutorialDataService {
  getAll() {
    return db;
  }

  create(tutorial: ITutorialData) {
    return db.push(tutorial);
  }

  update(key: string, value: any) {
    return db.child(key).update(value);
  }

  delete(key: string) {
    return db.child(key).remove();
  }

  deleteAll() {
    return db.remove();
  }
}

export default new TutorialDataService();
```

Figure 4- 3 : Création un service de données


```
import firebase from "firebase/app";
import "firebase/database";

let config = {
  apiKey: "AIzaSyAtneVZln--6rnR1vtl44MwnnRJPaTZ0t4",
  authDomain: "crud-user-a681d.firebaseio.com",
  databaseURL: "https://crud-user-a681d-default-rtdb.firebaseio.com",
  projectId: "crud-user-a681d",
  storageBucket: "crud-user-a681d.appspot.com",
  messagingSenderId: "86628520675",
  appId: "1:86628520675:web:c8e21accde5ea85fc42bd6"
};

firebase.initializeApp(config);

export default firebase.database();
```

Figure 4- 4 : Intégration Firebase

2.1.Composant de création d'objet

```
import { Component, ChangeEvent } from "react";
import TutorialDataService from "../services/tutorial.service";
import ITutorialData from "../types/tutorial.type";

type Props = {};

type State = ITutorialData ;

export default class AddTutorial extends Component<Props, State> {
  constructor(props: Props) {
    super(props);
    this.onChangeTitle = this.onChangeTitle.bind(this);
    this.onChangeDescription = this.onChangeDescription.bind(this);
    this.saveTutorial = this.saveTutorial.bind(this);
    this.newTutorial = this.newTutorial.bind(this);

    this.state = {
      title: "",
      description: "",
    };
  }

  onChangeTitle(e: ChangeEvent<HTMLInputElement>) {
    this.setState({
      title: e.target.value,
    });
  }

  onChangeDescription(e: ChangeEvent<HTMLInputElement>) {
    this.setState({
      description: e.target.value,
    });
  }
}
```

```
saveTutorial() {
  let data = {
    title: this.state.title,
    description: this.state.description,
  };

  TutorialDataService.create(data)
    .then(() => {
      console.log("Created new item successfully");
      this.setState({
        title: "",
        description: "",
      });
    })
    .catch((e: Error) => {
      console.log(e);
    });
}

newTutorial() {
  this.setState({
    title: "",
    description: "",
  });
}
```

Figure 4- 5 : Composant de création d'objet

La partie formulaire

```

render() {
  return (
    <div className="submit-form">
      {this.state.submitted ? (
        <div>
          <h4>You submitted successfully!</h4>
          <button className="btn btn-success" onClick={this.newTutorial}>
            Ajouter
          </button>
        </div>
      ) : (
        <div>
          <div className="form-group">
            <label htmlFor="title">Title</label>
            <input
              type="text"
              className="form-control"
              id="title"
              required
              value={this.state.title}
              onChange={this.onChangeTitle}
              name="title"
            />
          </div>
          <div className="form-group">
            <label htmlFor="description">Description</label>
            <input type="text"
              className="form-control"
              id="description"
              required
              value={this.state.description}
              onChange={this.onChangeDescription}
              name="description"
            />
          </div>
          <button onClick={this.saveTutorial} className="btn btn-success">
            Envoyer
          </button>
        </div>
      )}
    </div>
  );
}

```

Figure 4- 6 : Partie formulaire du composant de création d'objet

2.2. Composant pour la liste des objets

Cette composante comporte :

- Un tableau de tutoriels affiché sous la forme d'une liste sur la gauche.
- Un tutoriel sélectionné qui est affiché à droite.

```
import { Component } from "react";
import TutorialDataService from "../services/tutorial.service";
import Tutorial from "../tutorial.component";
import ITutorialData from "../types/tutorial.type";

type Props = {};

type State = {
  tutorials: Array<ITutorialData>,
  currentTutorial: ITutorialData,
  currentIndex: number,
};

export default class TutorialsList extends Component<Props, State> {
  constructor(props: Props) {
    super(props);
    this.refreshList = this.refreshList.bind(this);
    this.setActiveTutorial = this.setActiveTutorial.bind(this);
    this.removeAllTutorials = this.removeAllTutorials.bind(this);
    this.onDataChange = this.onDataChange.bind(this);

    this.state = {
      tutorials: [],
      currentTutorial: null,
      currentIndex: -1,
    };
  }

  //cycle de vue de composants
  componentDidMount() {
    TutorialDataService.getAll().on("value", this.onDataChange);
  }

  componentWillUnmount() {
    TutorialDataService.getAll().off("value", this.onDataChange);
  }
}
```

```
refreshList() {
  this.setState({
    currentTutorial: null,
    currentIndex: -1,
  });
}

setActiveTutorial(tutorial: ITutorialData, index: number) {
  this.setState({
    currentTutorial: tutorial,
    currentIndex: index,
  });
}

removeAllTutorials() {
  TutorialDataService.deleteAll()
    .then(() => {
      this.refreshList();
    })
    .catch((e: Error) => {
      console.log(e);
    });
}
```

```
render() {
  const { tutorials, currentTutorial, currentIndex } = this.state;

  return (
    <div className="list row">
      <div className="col-md-6">
        <h4> la Liste Tutorials </h4>

        <ul className="list-group">
          {tutorials &&
            tutorials.map((tutorial, index) => (
              <li
                className={
                  "list-group-item " +
                  (index === currentIndex ? "active" : "")
                }
                onClick={() => this.setActiveTutorial(tutorial, index)}
                key={index}>
                {tutorial.title}
              </li>
            ))}
        </ul>

        <button
          className="m-3 btn btn-sm btn-danger"
          onClick={this.removeAllTutorials}>
          Remove All
        </button>
      </div>
    </div>
  );
}
```

```
<div className="col-md-6">
  {currentTutorial ? (
    <Tutorial
      tutorial={currentTutorial}
      refreshList={this.refreshList}
    />
  ) : (
    <div>
      <br />
      <p>Please click on a Tutorial...</p>
    </div>
  )}
</div>
</div>
);
}
```

Figure 4- 7 : Composant pour la liste des objets

2.3. Composant pour les détails de l'objet

Ce composant obtenir la mise à jour, supprimez le tutorial, nous allons utiliser deux méthodes :

- Update ()
- Delete ()

```
import { Component, ChangeEvent } from "react";
import TutorialDataService from "../services/tutorial.service";
import ITutorialData from "../types/tutorial.type";

type Props = {
  tutorial: ITutorialData;
  refreshList: Function;
};

type State = {
  currentTutorial: ITutorialData;
  message: string;
};

export default class Tutorial extends Component<Props, State> {
  constructor(props: Props) {
    super(props);
    this.onChangeTitle = this.onChangeTitle.bind(this);
    this.onChangeDescription = this.onChangeDescription.bind(this);
    this.updateTutorial = this.updateTutorial.bind(this);
    this.deleteTutorial = this.deleteTutorial.bind(this);

    this.state = {
      currentTutorial: {
        key: null,
        title: "",
        description: "",
      },
    };
  }

  static getDerivedStateFromProps(nextProps: Props, prevState: State) {
    const { tutorial } = nextProps;
    if (prevState.currentTutorial.key !== tutorial.key) {
      return {
        currentTutorial: tutorial,
        message: "",
      };
    }
    return prevState.currentTutorial;
  }

  componentDidMount() {
    this.setState({
      currentTutorial: this.props.tutorial,
    });
  }

  onChangeTitle(e: ChangeEvent<HTMLInputElement>) {
    const title = e.target.value;

    this.setState(function (prevState: State) {
      return {
        currentTutorial: {
          ...prevState.currentTutorial,
          title: title,
        },
      };
    });
  }

  updateTutorial() {
    TutorialDataService.update(this.state.currentTutorial);
    this.setState({ message: "Tutorial updated" });
  }

  deleteTutorial() {
    TutorialDataService.delete(this.state.currentTutorial.key);
    this.setState({ message: "Tutorial deleted" });
  }
}
```

```
static getDerivedStateFromProps(nextProps: Props, prevState: State) {
  const { tutorial } = nextProps;
  if (prevState.currentTutorial.key !== tutorial.key) {
    return {
      currentTutorial: tutorial,
      message: "",
    };
  }
  return prevState.currentTutorial;
}

componentDidMount() {
  this.setState({
    currentTutorial: this.props.tutorial,
  });
}

onChangeTitle(e: ChangeEvent<HTMLInputElement>) {
  const title = e.target.value;

  this.setState(function (prevState: State) {
    return {
      currentTutorial: {
        ...prevState.currentTutorial,
        title: title,
      },
    };
  });
}

updateTutorial() {
  TutorialDataService.update(this.state.currentTutorial);
  this.setState({ message: "Tutorial updated" });
}

deleteTutorial() {
  TutorialDataService.delete(this.state.currentTutorial.key);
  this.setState({ message: "Tutorial deleted" });
}
```

```
onChangeDescription(e: ChangeEvent<HTMLInputElement>) {  
  const description = e.target.value;  
  
  this.setState((prevState) => ({  
    currentTutorial: {  
      ...prevState.currentTutorial,  
      description: description,  
    },  
  }));  
}
```

```
updateTutorial() {  
  if (this.state.currentTutorial.key) {  
    const data = {  
      title: this.state.currentTutorial.title,  
      description: this.state.currentTutorial.description,  
    };  
  
    TutorialDataService.update(this.state.currentTutorial.key, data)  
      .then(() => {  
        this.setState({  
          message: "The tutorial was updated successfully!",  
        });  
      })  
      .catch((e: Error) => {  
        console.log(e);  
      });  
  }  
}  
  
deleteTutorial() {  
  if (this.state.currentTutorial.key) {  
    TutorialDataService.delete(this.state.currentTutorial.key)  
      .then(() => {  
        this.props.refreshList();  
      })  
      .catch((e: Error) => {  
        console.log(e);  
      });  
  }  
}
```

```

render() {
  const { currentTutorial } = this.state;
  return (
    <div>
      <h4>Tutorial</h4>
      {currentTutorial ? (
        <div className="edit-form">
          <form>
            <div className="form-group">
              <label htmlFor="title">Title</label>
              <input
                type="text"
                className="form-control"
                id="title"
                value={currentTutorial.title}
                onChange={this.onChangeTitle}
              />
            </div>
            <div className="form-group">
              <label htmlFor="description">Description</label>
              <input
                type="text"
                className="form-control"
                id="description"
                value={currentTutorial.description}
                onChange={this.onChangeDescription}
              />
            </div>
          </form>

```

```

      <button
        type="submit"
        className="badge badge-success"
        onClick={this.updateTutorial}
      >
        update
      </button>
      <button
        className="badge badge-danger mr-2"
        onClick={this.deleteTutorial}
      >Delete
      </button>
      <p>{this.state.message}</p>
    </div>
  ) : (
    <div>
      <br />
      <p>Please click on a Tutorial...</p>
    </div>
  )
}
</div>
);
}

```

Figure 4- 8 : Composant pour les détails de l'objet

2.4. Ajouter un CSS pour les composants React Typescript

```

.container h2 {
  text-align: center;
  margin: 25px auto;
}

.list {
  text-align: left;
  max-width: 750px;
  margin: auto;
}

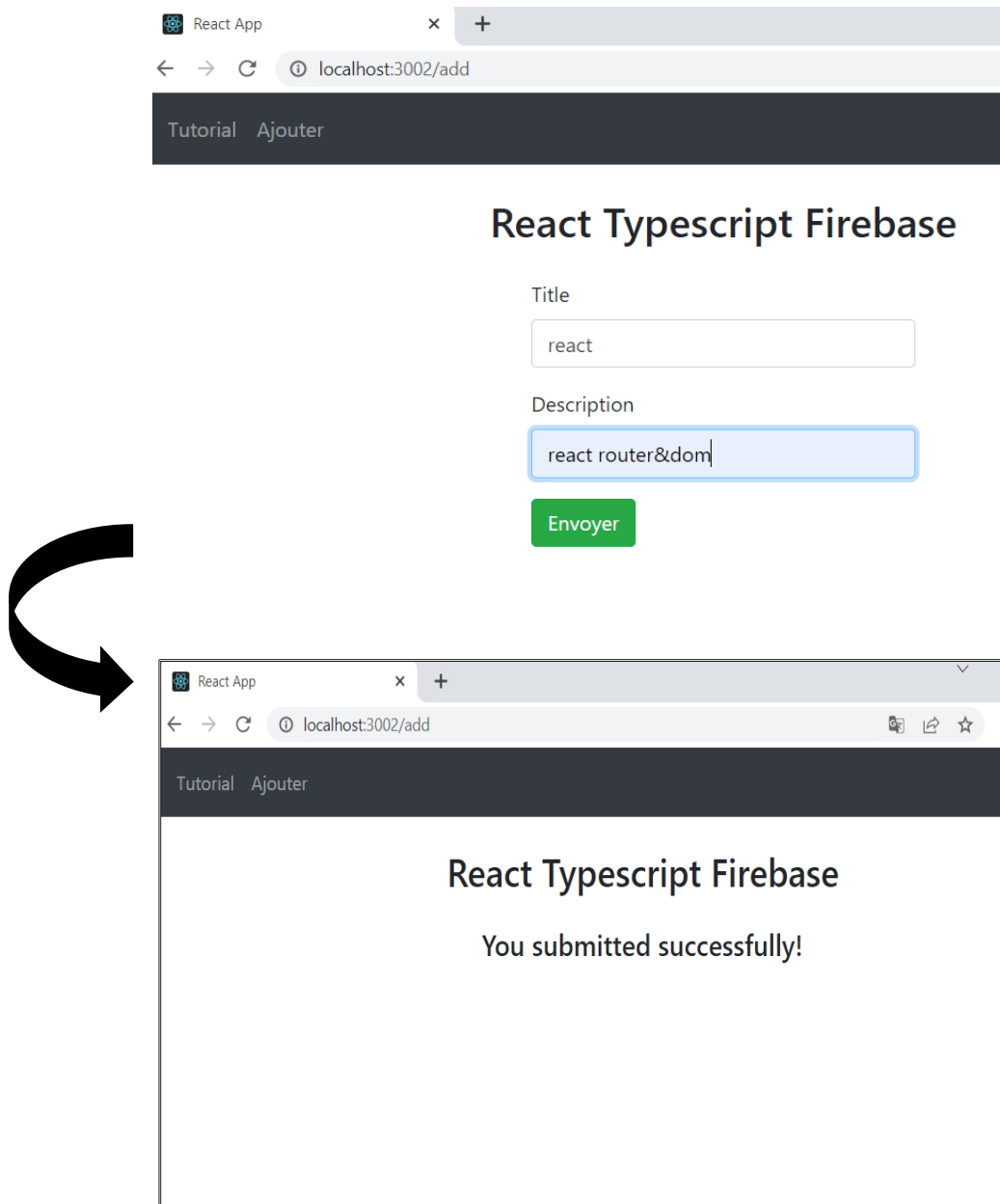
.submit-form {
  max-width: 300px;
  margin: auto;
}

.edit-form {
  max-width: 300px;
  margin: auto;
}

```

Figure 4- 9 : L'interface CSS

3. La réalisation finale



React App x +

localhost:3002/add

Tutorial Ajouter

React Typescript Firebase

Title

Description

Envoyer

Tutorial Ajouter

React Typescript Firebase

You submitted successfully!

Figure 4- 10 : L'interface d'ajouter tutoriel

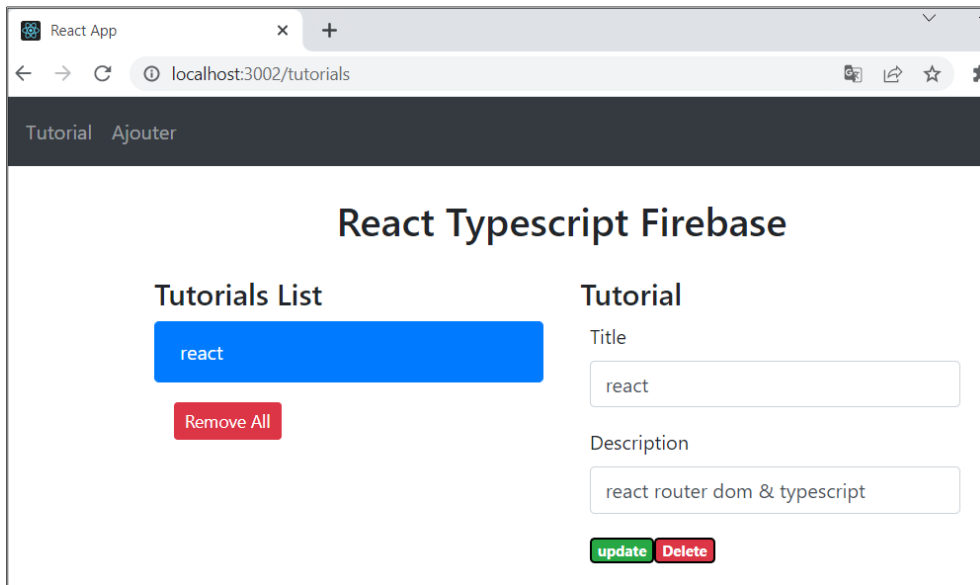


Figure 4- 11 : L'interface de la liste tutoriel

4. Conclusion

Dans ce chapitre, nous avons construit l'application React Typescript Firebase **CRUD** en travaillant avec succès avec la base de données en temps réel à l'aide de la bibliothèque **Firebase**.



Conclusion générale



Conclusion

Ce stage était donc l'occasion et le moyen de découvrir un milieu professionnel totalement inconnu et de reprendre contact avec les réalités du monde de travail.

Pour conclure, ce stage a été une bonne occasion non seulement pour élargir mes connaissances pratiques notamment dans le domaine développement une application web, mais aussi pour s'adapter aux nouvelles technologies qui ne cessent de s'améliorer. En effet il s'agit d'un secteur qui s'éveille chaque jour sur des nouvelles innovations.