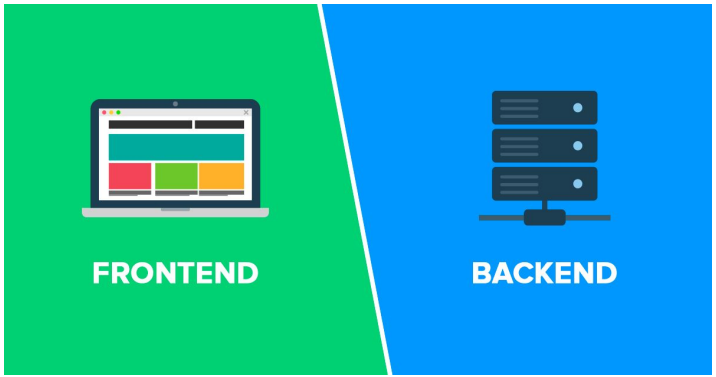
Projet
Web Services



Thibaud Cruzille - Rémy Viniacourt - Rémy Moulaire

Sommaire

Objectifs

Fournir une application web permettant d'imputer les temps passés sur des projets.

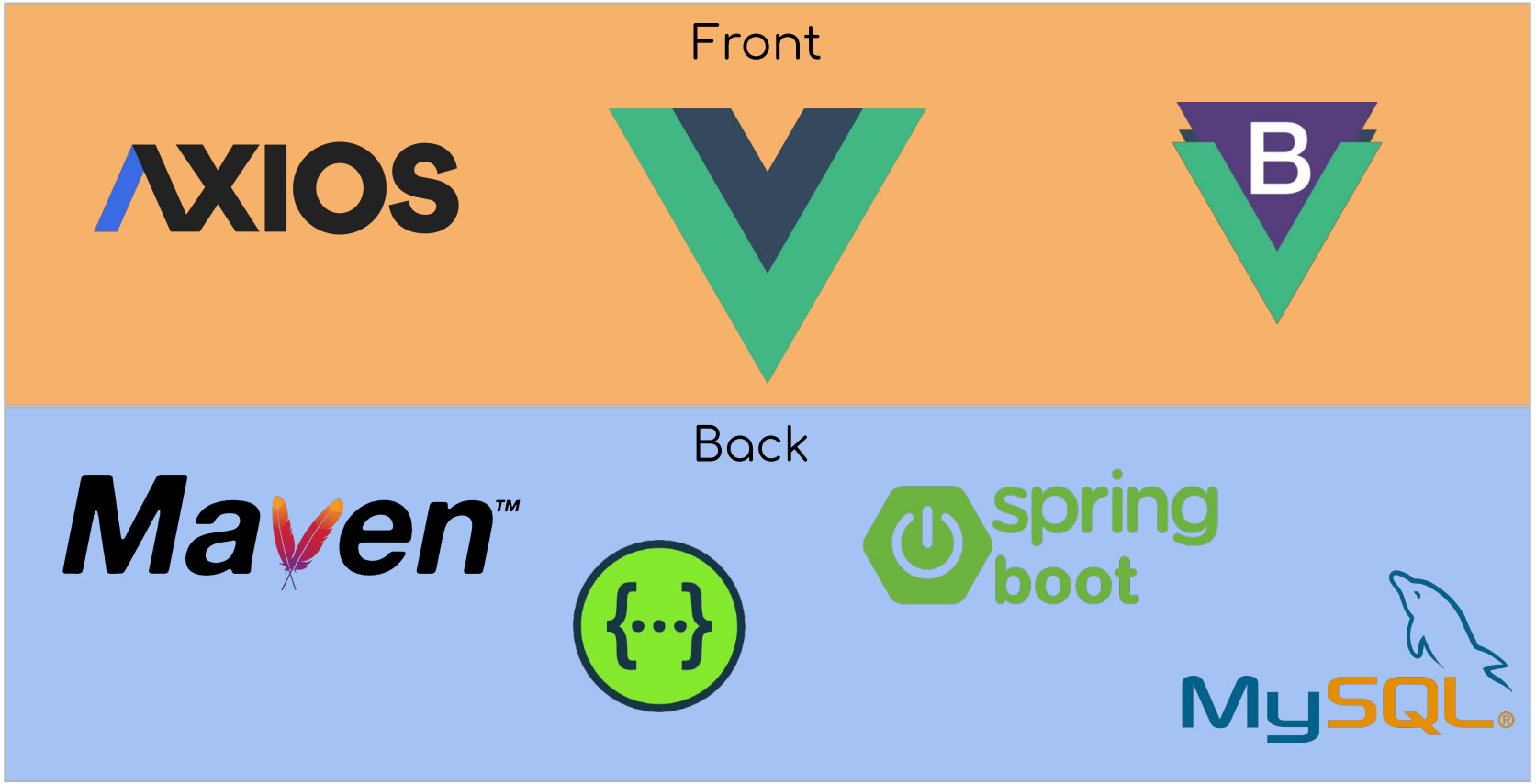
Features :

- Connexion
- Lire ses temps
- Ajouter un temps
- Supprimer un temps
- Imprimer la liste des temps par mois

Rôles

- Developer : rôle de base
- Manager : rôle qui donne accès à un écran supplémentaire qui liste son équipe et lui donne accès à leurs temps.

Technologies



Listes fonctionnalités

API SOAP Users :

Connexion à l'application :
- Input : Email et mot de passe
- Output : Objet user qui correspond

Les utilisateurs par équipe :
- Input : Le numéro de l'équipe
- Output : La liste des utilisateurs de cette équipe

API SOAP Projects :

Récupérer tous les projets :
- Input : Aucun
- Output : Tous les projets

Usages Back end

API SOAP Users :

Définition du contenu de l’API :

L’entité

users.xsd

```
<xs:complexType name="user">
    <xs:sequence>
        <xs:element name="id_user" type="xs:int"/>
        <xs:element name="firstname" type="xs:string"/>
        <xs:element name="lastname" type="xs:string"/>
        <xs:element name="mail" type="xs:string"/>
        <xs:element name="password" type="xs:string"/>
        <xs:element name="role" type="tns:role"/>
        <xs:element name="team" type="xs:int"/>
    </xs:sequence>
</xs:complexType>

<xs:simpleType name="role">
    <xs:restriction base="xs:string">
        <xs:enumeration value="manager"/>
        <xs:enumeration value="developer"/>
    </xs:restriction>
</xs:simpleType>
```

Usages Back end

API SOAP Utilisateur :

Définition du contenu de l’API

La fonction

```
<xs:element name="getUserByTeamRequest">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="team" type="xs:int"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:element name="getUserByTeamResponse">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="user" type="tns:user" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

Usages Back end

endpoint.java

```java
@PayloadRoot(namespace = NAMESPACE_URI, localPart = "getUserByTeamRequest")
    @ResponsePayload
    public GetUserByTeamResponse getUser(@RequestPayload GetUserByTeamRequest request) {
        GetUserByTeamResponse response = new GetUserByTeamResponse();
        response.getUser().addAll(userRepository.findUserByTeam(request.getTeam()));

        return response;
    }
```

repository.java

```java
public List<User> findUserByTeam(Integer team) {
    Assert.notNull(team, "The user's team must not be null");
    List<User> teamUsers = new ArrayList<User>(10);
    for(Map.Entry<Integer, User> entry : users.entrySet()) {
        if(entry.getValue().getTeam()==team) {
            teamUsers.add(entry.getValue());
        }
    }
    return teamUsers; //retourne la liste des users
}
```

Listes fonctionnalités

API REST Time :

Récupération temps par utilisateur :
- Input : Utilisateur, année et mois/semaine
- Output : Objet time qui correspond

Création d’un temps pour un utilisateur :
- Input : Objet time complet
- Output : Objet créé

Suppression d’un temps pour un utilisateur :
- Input : Id de l’objet time
- Output : Objet supprimé

Usages Back end

API REST Gestion des temps :

Définition de la classe Time

Time.java

```java
@Entity
public class Time {

@Id
@GeneratedValue
private Integer id_time;
private Integer id_user;
private Integer id_project;
private Integer nb_time;
private Integer week;
private Integer month;
private Integer year;
```

Usages Back end

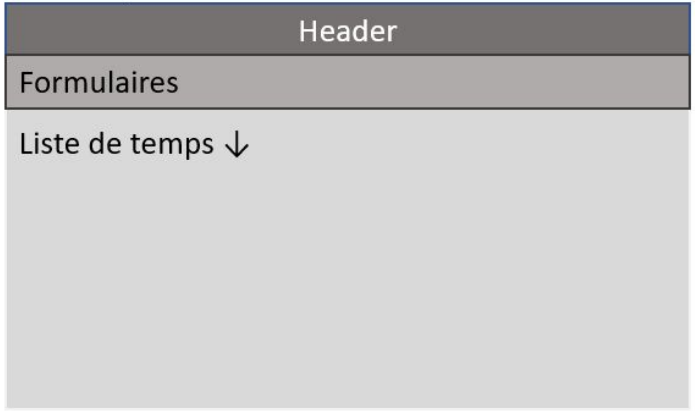
Controller.java

```
@GetMapping("/time/{id_time}")
public Time findTime(@PathVariable Integer id_time) {
    return this.timeService.findTime(id_time);
}

@PostMapping("/time")
public Time createTime(@RequestBody Time time) {
    return this.timeService.createTime(time);
}

@DeleteMapping("/time/{id_time}")
public void deleteTime(@PathVariable Integer id_time) {
    this.timeService.deleteTime(id_time);
}
```

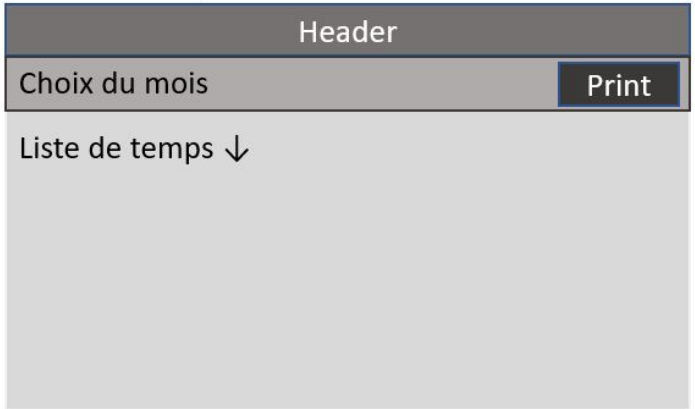
Agencement Front end

Vue Router

Mes temps



Editer les temps



Gérer mon équipe

Exemples de code

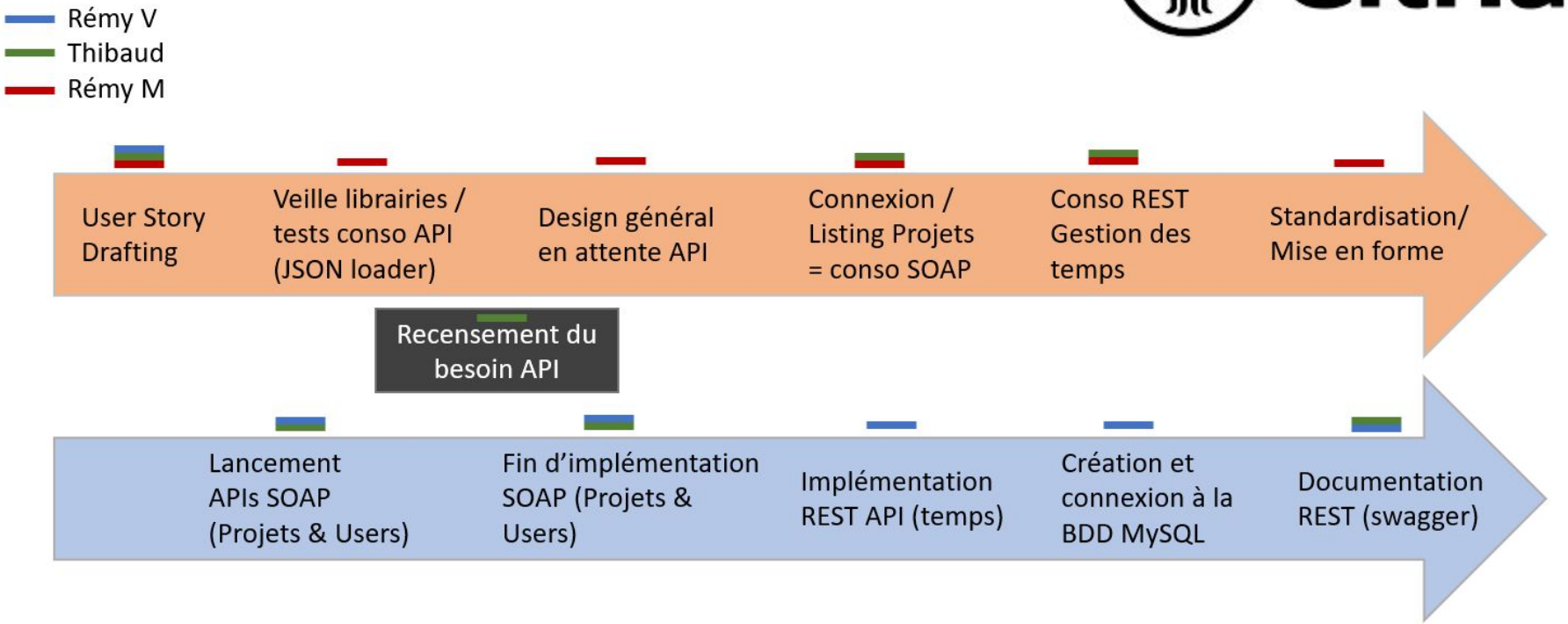
- Vue router pour lien direct entre les pages
- Axios pour toutes les requêtes API
- Données API SOAP gérées avec DomParser

```
let queryFetchTimes = '{'+
                      '"id_user":'+this.$userForm.user_id+','+
                      '"week":'+this.filterForm.week+','+
                      '"year":'+this.filterForm.year+
                      '}';
                    console.log(queryFetchTimes)

    let urlTimes = this.$urlREST_fetchTimes;

    axios({
      method: 'post',
      url: urlTimes,
      data: queryFetchTimes,
      headers: { "Content-Type": "application/json" }
              })
    .then(timeRes=>{ this.timeList = timeRes.data;
```

```
<b-collapse id="nav-collapse" is-nav>
  <b-navbar-nav>
    <b-nav-item> <router-link to="/" class="headerMenuText">Homepage</router-link> </b-nav-item
    <b-nav-item> <router-link to="/Edition" class="headerMenuText">Édition</router-link> </b-na
  </b-navbar-nav>
</b-collapse>
```

```
var parser = new DOMParser();
var xmlDoc = parser.parseFromString(res.data,"text/xml");
var projects_id = xmlDoc.getElementsByTagName("ns2:id_project");
var projects_name = xmlDoc.getElementsByTagName("ns2:name");
```

Méthodologie projet



tests unitaires - tests de non régressions - tests fonctionnels

Démonstration...

Points d’évolution

- Ajouter le fonctionnement des temps de l’équipe pour le manager.
- Permettre le rafraîchissement automatique.
- Ajouter la possibilité de modifier un temps.
- Consolider les temps communs.
- Gérer les alertes

Conclusion

Lien vers github :
https://github.com/remyvin/tse_de3_webservices