Université de Montpellier

Faculté des Sciences

HAI405I - Projet de programmation

Groupe 16

Roman Chris
Garcia Charly
Bonnafé Rémi
Hafdane Léo

L2 Informatique

1 Introduction

Notre équipe, composée de Léo, Rémi, Chris et Charly, a développé au cours de ces deux dernières semaines un site de jeux de cartes en utilisant React, un framework JavaScript destiné à la création d'interfaces utilisateurs interactives.

Lors de la première semaine, notre objectif principal était de mettre en place le site avec un jeu : la bataille. Nous avons commencé par définir le schéma visuel de chaque page, puis nous avons procédé à la mise en place du serveur. Les tâches ont été réparties entre les membres de l'équipe.

Au cours de la deuxième semaine, nous avons accueilli Charly au sein de notre équipe. Ensemble, nous avons ajouté le jeu "six qui prend" et avons amélioré diverses fonctionnalités.

Dans ce rapport, nous présenterons l'architecture de notre projet, notre utilisation de React pour la création de composants réutilisables, ainsi que notre protocole de communication et d'échange de données. Enfin, nous dresserons un bilan de notre expérience et des accomplissements réalisés au cours de ces deux semaines de travail intense et collaboratif.

2 Organisation

2.1 Semaine 1

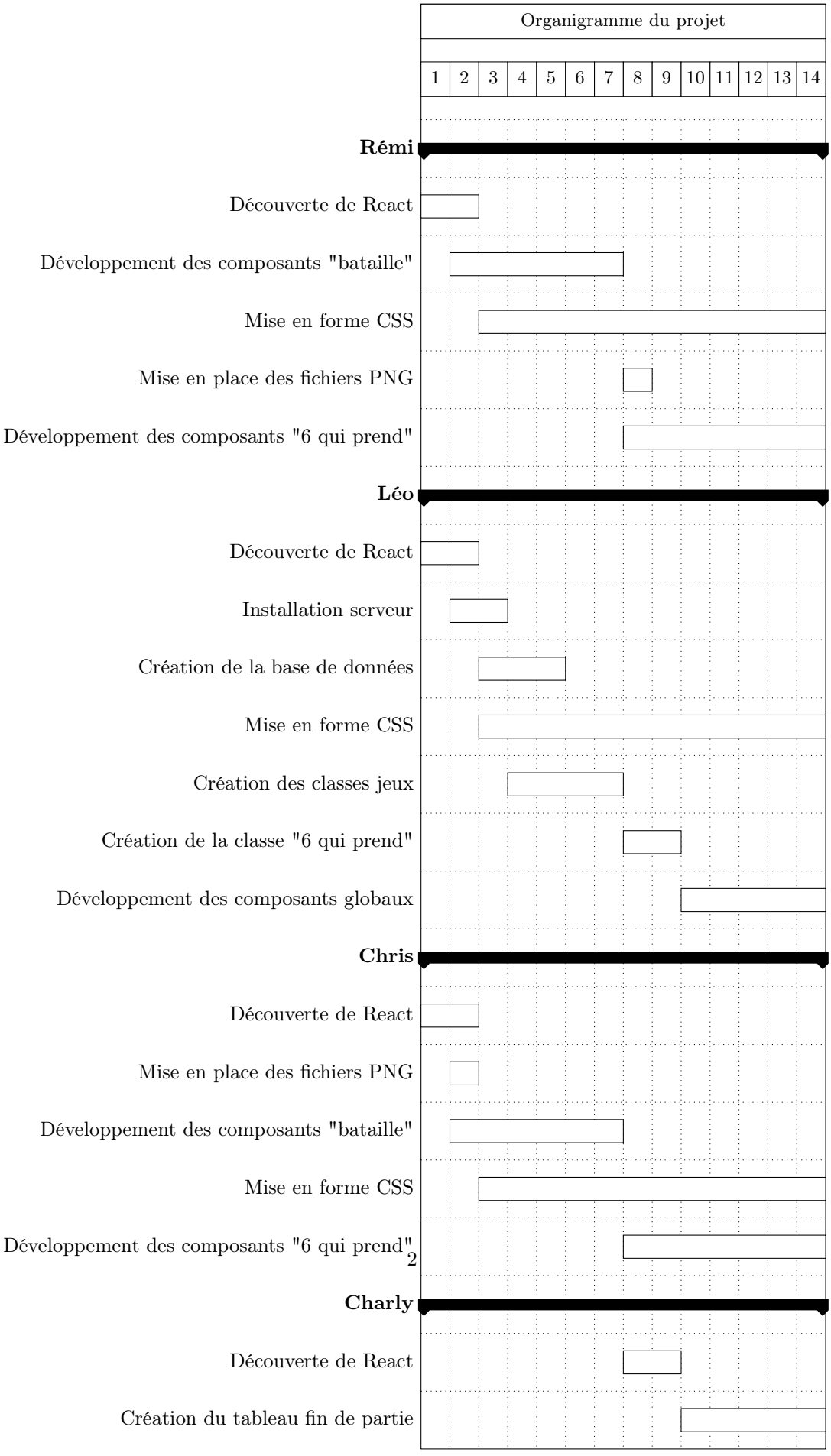
Au cours de la première semaine, le groupe a découvert React, a construit le schéma visuel de chaque page et a mis en place le serveur. Voici la répartition des tâches :

- Léo s'est principalement occupé du serveur, notamment la mise en place des sockets pour assurer la communication avec le client.
- Rémi s'occupait de la page « plateau bataille » et de ses composants : « Carte », « JoueurBataille », « Start » et « MonJeux ».
- Chris a travaillé sur la page « créer/rejoindre » et ses composants.
- Léo a mis en place la base de données pour gérer les comptes utilisateurs.
- Chris s'est occupé des fichiers PNG.
- Rémi et Chris se sont occupés de la mise en forme CSS et de la liaison avec le serveur des pages existantes et ont également créé les composants : « chat » et « sélection jeux ».
- Léo a créé les classes des jeux et a complété celle de la bataille.

2.2 Semaine 2

Charly a rejoint le groupe au cours de la deuxième semaine. Il a découvert React et le projet. Voici la répartition des tâches :

- Léo a créé la classe "6 qui prend" pendant que Chris et Rémi ont développé la page associée et ses composants, en modifiant : « carte » et « chat ».
- Charly a créé le tableau des scores en fin de partie.
- Rémi s'est occupé d'enregistrer les scores et les parties dans la base de données pour les utiliser dans le « profil » et le « leaderboard » créé avec Chris.
- Chris a créé l'audio et Léo a réglé des problèmes d'historique, fait une navbar et créé des SVG très jolis pour le chat, la navbar et l'audio pour une meilleure expérience utilisateur.
- Léo et Rémi ont créé un timer pour le choix de la carte ou de la colonne.
- Chris a créé un rang dans le profil.


Organigramme du projet
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Rémi
Découverte de React
Développement des composants "bataille"
Mise en forme CSS
Mise en place des fichiers PNG
Développement des composants "6 qui prend"
Léo
Découverte de React
Installation serveur
Création de la base de données
Mise en forme CSS
Création des classes jeux
Création de la classe "6 qui prend"
Développement des composants globaux
Chris
Découverte de React
Mise en place des fichiers PNG
Développement des composants "bataille"
Mise en forme CSS
Développement des composants "6 qui prend"2
Charly
Découverte de React
Création du tableau fin de partie

3 Architecture, protocole de communication et échange de données

Dans notre projet, trois grandes entités communiquent : le client, le serveur React et le serveur Node.

Le serveur React et le serveur Node communiquent avec le protocole WebSocket (nous n'utilisons pas HTTP). WebSocket permet d'échanger des variables et des listes de variables via une requête. Le protocole HTTP pourrait être utilisé, par exemple, pour les "room", où le lien de la partie pourrait être l'objet d'une requête HTTP.

Voici comment nous avons mis en place les sockets :

```
const express = require("express");
const app = express();
const http = require("http");
const { Server } = require("socket.io");
const cors = require("cors");
app.use(cors());
const httpServer = http.createServer(app);
const io = new Server(httpServer, {
    cors: {
        origin: "*",
        methods: ["GET", "POST"],
    },
});
const port = process.env.HAI405I_PORT || 3001;
httpServer.listen(port, () => {
    console.log(`server running on port http://localhost:${port}/`);
});

const sockets = {}; // clef: socket.id         valeur: {compte, partie}
const parties = {}; // clef: code de la partie valeur: instance de jeu
```

Voici un exemple d'utilisation des sockets :

Émission de la requête :

```
    function carteClick() {
        socket.emit("reqCoup", { carte: { valeur: valeur, type: type }, index: props.index });
    }
```

Réception de la requête :

```
    socket.on("reqCoup", async (json) => {
        if (!sockets[socket.id]) {
            return;
        }
        const carte = json.carte;
        const index = json.index;
        const code = sockets[socket.id].partie;
        const jeux = parties[code];
        if (!jeux || !jeux.coup(socket.id, carte, index)) {
            return;
        }
        resPlayers(code);
        resPlateau(code);
        if (jeux.everyonePlayed()) {
```

```
            if (jeux.nextRound()) {
                await new Promise((r) => setTimeout(r, 1000));
                resPlayers(code);
                resPlateau(code);
            }
        }
    });
```

4 Utilisation de React

React est un framework qui permet la création de composants autonomes et réutilisables retournant du HTML. Ces composants peuvent être associés pour créer des pages complexes.

Cette architecture nous a permis, par exemple, de créer un composant "carte" utilisé dans toutes les pages "in-game". Nous pouvons également créer des composants avec des paramètres :

```
import "./Start.css";

function Start(props) {
    return (
        <div id="divStart">
            <label className="code">code de la partie:</label>
            <label className="code">{props.code}</label>
            <button
                hidden={!props.afficheStart}
                onClick={props.start}
            >
                commencer
            </button>
            <button
                hidden={!props.afficheSave}
                onClick={props.save}
            >
                save
            </button>
        </div>
    );
}
export default Start;
```

Ce composant peut être utilisé dans n'importe quel autre composant ou page avec la balise :

```
import Carte from "../../../component/Carte/Carte";

<Carte visible={...}>
```

5 Bilan

Au terme de ces deux semaines de travail sur notre projet, nous pouvons constater quelques difficultés rencontrées ainsi qu'un apprentissage en React.

5.1 Difficultés rencontrées

— Sauvegarde des parties : L'une des principales difficultés auxquelles nous avons été confrontés a été la mise en œuvre de la fonction de sauvegarde des parties. Nous avons rencontré des obstacles techniques.
— Mise en place du serveur : Un autre défi majeur a été la configuration et la mise en place du serveur, en particulier dans le contexte de la communication entre le serveur React et le serveur Node.

5.2 Analyse rétrospective

Sur le plan organisationnel, nous avons appris l'importance de GitHub lors d'un projet complexe. La répartition claire des tâches, l'entraide et la mise en place de points de contrôle réguliers ont contribué à maintenir notre équipe sur la bonne voie malgré les défis rencontrés.

Du point de vue technique, nous avons appris à utiliser React et à configurer des serveurs.

En fin de compte, ce projet nous a appris à travailler en groupe et à développer en React.