Bienvenue

Nous sommes la maison
Pandargent

Notre jeu ⚡

- Une course de balais
- Transposition du mille bornes dans l'univers Harry Potter
- Ajout de cartes et de règles par rapport au jeu de base

Le mille bornes

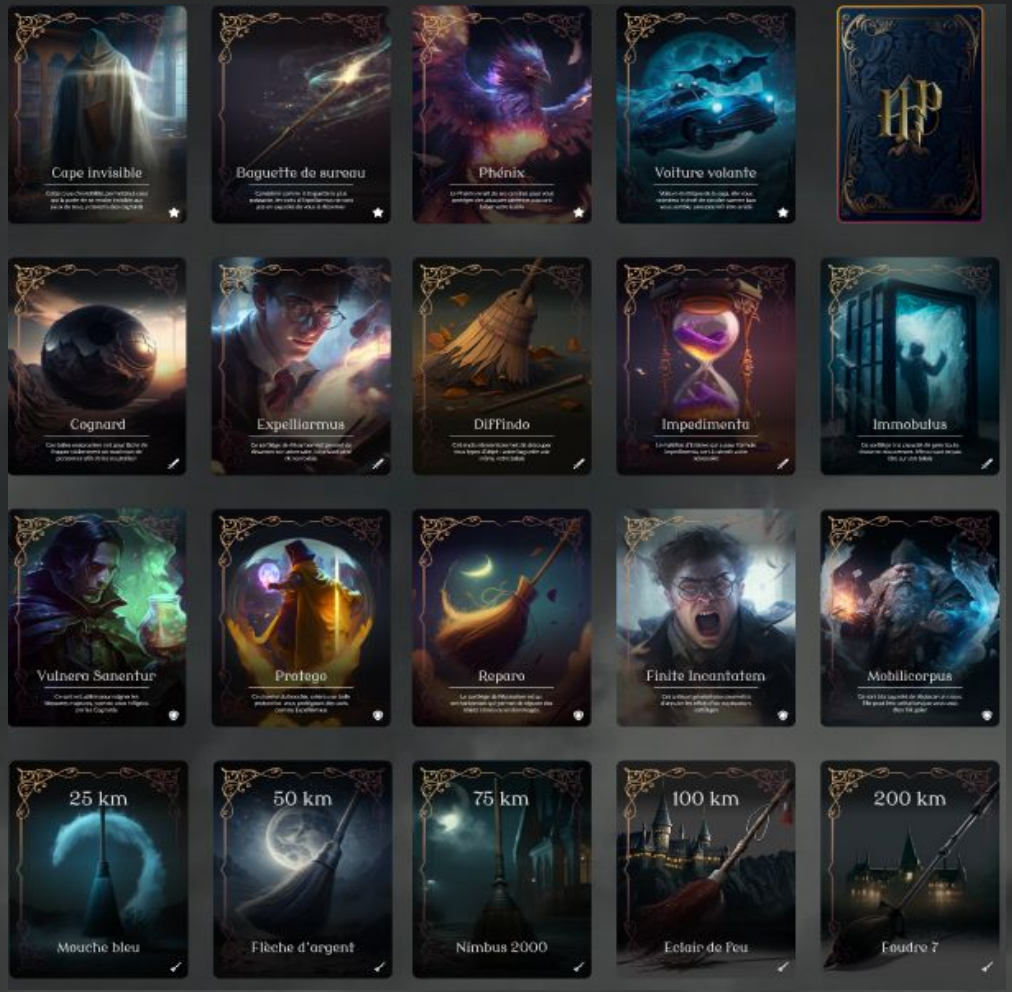
Jeu de société :

- ➔ 110 cartes
- ➔ 2 à 8 joueurs
- ➔ Objectif : atteindre 1000 bornes !

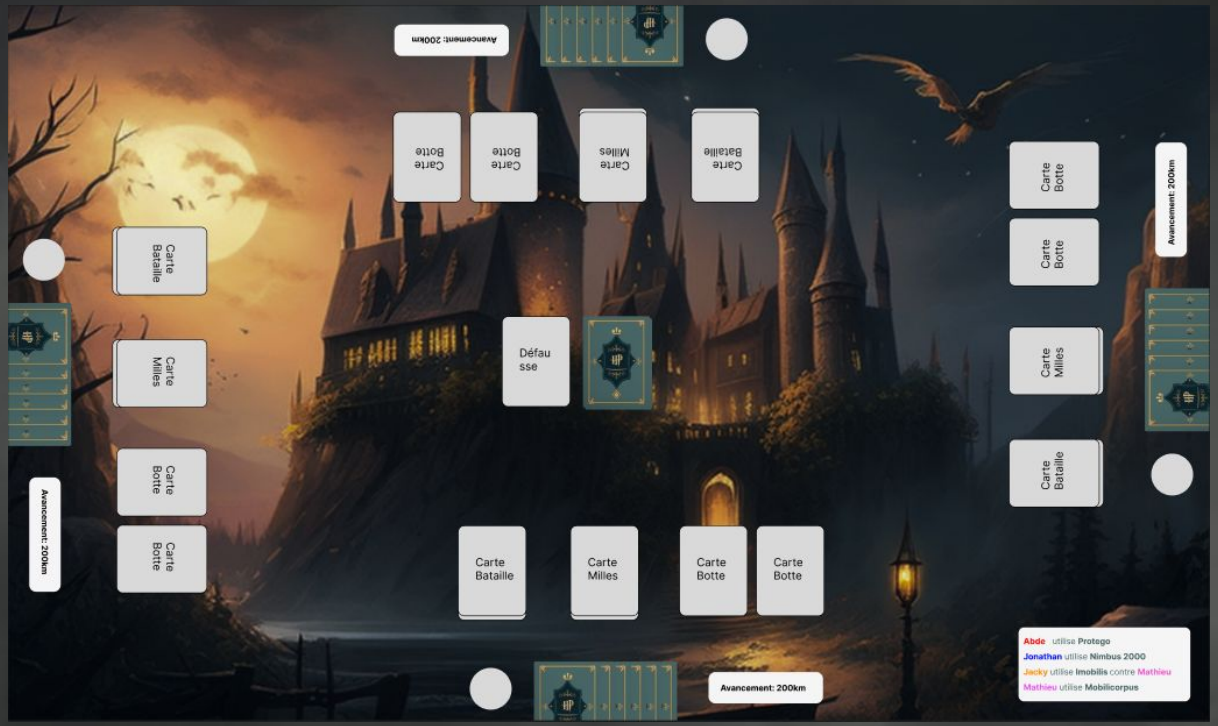
Liste des cartes 🃏

4 catégories :

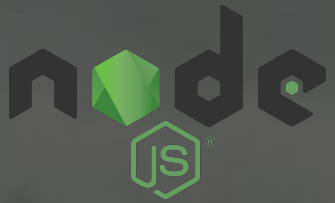
- ➔ Spéciale
- ➔ Attaque
- ➔ Défense
- ➔ Distance



La maquette magique



Technos utilisées 🧙



Difficultées 👽

- Manque de temps
- Utilisation de technos pour la première fois (Firebase, Svelte.js)
- Complexité du jeu à reproduire

Points forts 🏰

- Un back-end complet et fonctionnel
- Un design cohérent et enchanteur
- Un projet challengeant et ambitieux

Fonctionnement - exemple : Jouer une carte

```
{#if showActionCard && me && players && playCard !== undefined}
    <Action card={showActionCard} closeAction={closeAction} me={me}
    players={players} playCard={playCard} discardCard={discardCard}/>
{/if}
```

Hand.svelte

```
{#if canAddCardToMe}
<button on:click={()=>playCardAndClose(me,card)}>
    Jouer la carte sur moi</button>{/if}
{#if canAddCardToPlayer2}
<button on:click={()=>playCardAndClose(players[0],card)}>
    Jouer la carte sur le {players[0].pseudo}</button>{/if}
```

Action.svelte

1) Le joueur clique sur une carte dans sa main pour la jouer sur lui ou contre un autre joueur

Fonctionnement - exemple : Jouer une carte

page.svelte

```
const playCard = (target, card) => {
    let nPlayer2 = players.findIndex((player) => player.pseudo === target.pseudo)
    const cardIndex = me.hand.findIndex((cardPlayer) => cardPlayer.id === card.id)
    me.hand.splice(cardIndex, 1)
    me = me

    socket.emit('send_playCard', {
        nPlayer: nPlayer,
        nPlayer2: nPlayer2,
        card: card
    });
}
```

2) On enlève la carte de la main du joueur, et on envoie une socket avec le joueur, le joueur cible et la carte,

server.js

```
socket.on("send_playCard", ({nPlayer, nPlayer2, card}) => {
    const cardWithMethod = jsonGame.players[nPlayer].hand.find(cardInHand => cardInHand.id === card.id);
    jsonGame.players[nPlayer].playCard(cardWithMethod, jsonGame.players[nPlayer2])

    if (cardWithMethod.type === CardType.DIST) {
        sendAll("get_distance", {
            action: "distance",
            nPlayer: nPlayer2,
            distance: jsonGame.players[nPlayer2].progress,
        })
    }

    sendAll("get_playCard", {
        action: "playCard",
        nPlayer: nPlayer2,
        card: cardWithMethod,
    })
    sendAll("get_nextPlayer", {
        nextPlayer: jsonGame.passPlayer(),
        noDistCard: noDistCardInHands(),
        noDistCardInDeck: noDistCardInDeck(jsonGame.deck)
    })
});
```

3) Si c'est une carte Distance on envoie la socket d'incrément du km du joueur.

4) On envoie les infos de la carte jouée

5) Et enfin on passe au joueur suivant en envoyant les infos de s'il reste des cartes Distance dans la pioche ou les mains (sinon c'est la fin de la partie)

page.svelte

```
socket.on('get_distance', ({nPlayer, distance}) => {
    players[nPlayer].progress = distance
    players = players
});
```

6) On incrémente la progression(km) du joueur

```
socket.on('get_playCard', ({nPlayer, card}) => {
    if (card.type === CardType.SPEC) {
        players[nPlayer].specialCards.push(card)
        if(checkSpecialCardAgainstMalus(card.name, players[nPlayer].state.name)){
            players[nPlayer].state = undefined
        }
    } else if (card.type === CardType.BON || card.type === CardType.MAL) {
        players[nPlayer].state = card
    } else if (card.type === CardType.DIST) {
        players[nPlayer].distanceCard = card
    }

    players = players
});
```

7) Selon le type de la carte jouée, on la place au bon endroit avec les bons effets. Par exemple, le joueur peut faire disparaître une carte Malus placée par un autre joueur dans son plateau en utilisant une carte Spéciale.

page.svelte

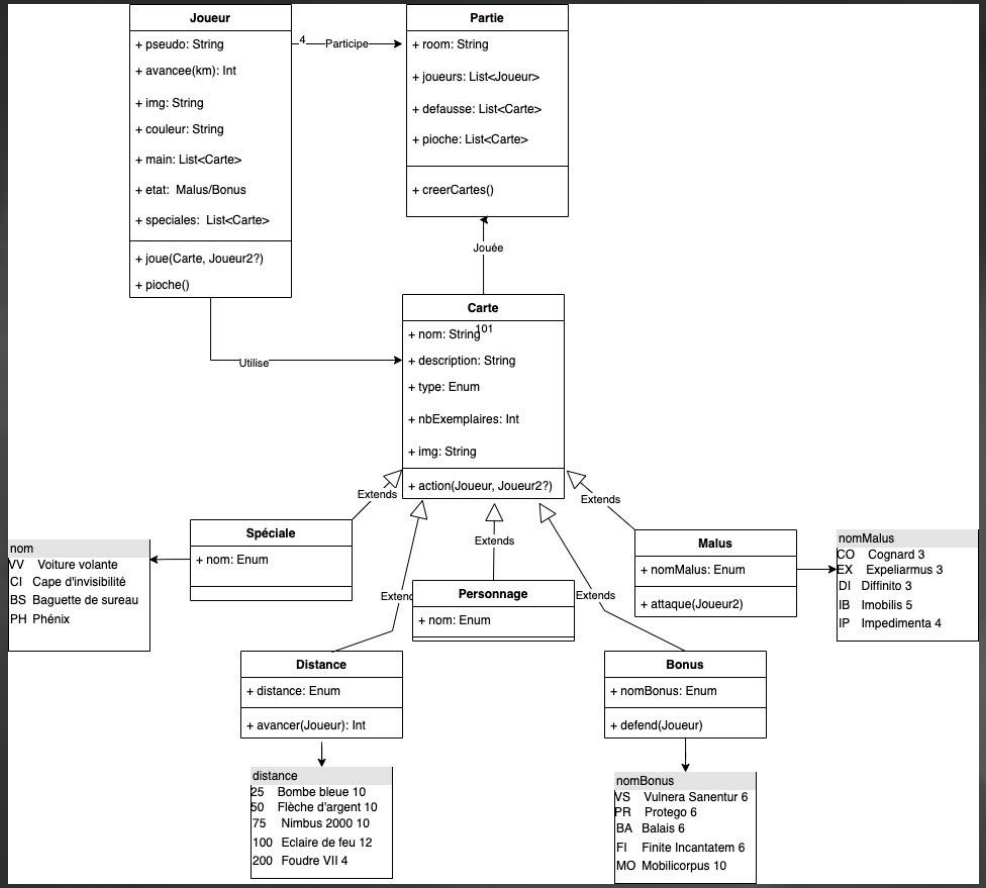
```
socket.on('get_nextPlayer', ({nextPlayer, noDistCard,noDistCardInDeck}) => {
    //game finish by 1000km reached
    if (isDistanceReached()!==false){
        blockActions(true)
        noteInfo = {
            text:`Partie terminée. ${isDistanceReached()} a remporté la course !`,
            isActionNeeded: false
        }
    } else if (noDistCard && noDistCardInDeck){
        blockActions(true)
        noteInfo = {
            text:`Il n'y a plus de carte distance. ${players[nextPlayer].pseudo} a remporté la course !`,
            isActionNeeded: false
        }
    } else {
        if (nPlayer === nextPlayer) {
            blockActions(false)
            noteInfo = {
                text: "C'est votre tour. Piochez une carte.",
                isActionNeeded: true
            }
        }
        else {
            blockActions(true)
            noteInfo = {
                text: `Tour de ${players[nextPlayer].pseudo}`,
                isActionNeeded: false
            }
        }
        currentPlayer = nextPlayer
```

8) Avant de passer au joueur suivant on vérifie si la partie n'est pas terminée car :

- 1000km atteint par le joueur
  OU
- Plus de cartes distances dispo

On passe la main au prochain joueur en bloquant les actions de celui qui a joué

Diagramme de classes



Merci

de votre attention