



TITRE PROFESSIONNEL :
CONCEPTEUR DEVELOPPEUR
D'APPLICATIONS

RNCP 31678

PRÉSENTATION DU PROJET

COSPLAY-MAKER

PAR CASSANDRA FORESTIER

LE BOCAL ACADEMY

Nice

31 Juillet 2023

Le cosplay en quelques mots

LE COSPLAY CONSISTE À RECRÉER UN COSTUME DE PERSONNAGE
DE POP CULTURE

Problématiques

- COMMENT AMÉLIORER L'EXPÉRIENCE DES PASSIONNÉS DU COSPLAY ?
- COMMENT FACILITER LA RECHERCHE DE MATERIAUX ?
- COMMENT INITIER LES DÉBUTANTS AU COSPLAY ?
- COMMENT PERMETTRE AUX ORGANISATEURS DE CONVENTION DE TROUVER DES COSPLAYEURS ?



Objectifs

- Développer une application web pour créer une communauté en ligne
- Faciliter le processus de création d'un cosplay
- Officialiser le lien entre les organisateurs et les cosplayeurs



Périmètre et limite du projet

Plateforme cible :

- Application web responsive
- Avantages: Accessibilité multiplateforme, pas de téléchargement requis, mises à jour simplifiées

Contraintes supplémentaires :

- Conformité RGPD pour collecte et stockage des données personnelles
- Mise à jour régulière d'un agenda de conventions
- Gestion spécifique pour les utilisateurs "gérants convention"

Présentation de l'Équipe



Cassandra FORESTIER

*Responsable du projet
"Cosplay-Maker"*

Compétences attendues :

Conception, développement front-end et back-end,
gestion de base de données,
conception graphique, CI/CD, DevOps,
sécurité, veille technologique, mise à jour du Kanban, déploiement de
l'application.

Livrable :

Sortir la première version du site en un an.

PARTIE CONCEPTION DU PROJET



Cas d'utilisation

ACTEURS :

- Cosplayeurs
- Organisateurs d'évènements
- Administrateurs

ACTEUR “COSPLAYEUR“ :

- Créer/modifier/supprimer un compte
- CRUD des cosplays
- CRUD des évènements
- Rechercher des cosplays
- Préciser un cosplay pour un événement
- Se connecter
- Se déconnecter
- Récupérer un mot de passe oublié
- Mettre en favoris des cosplays

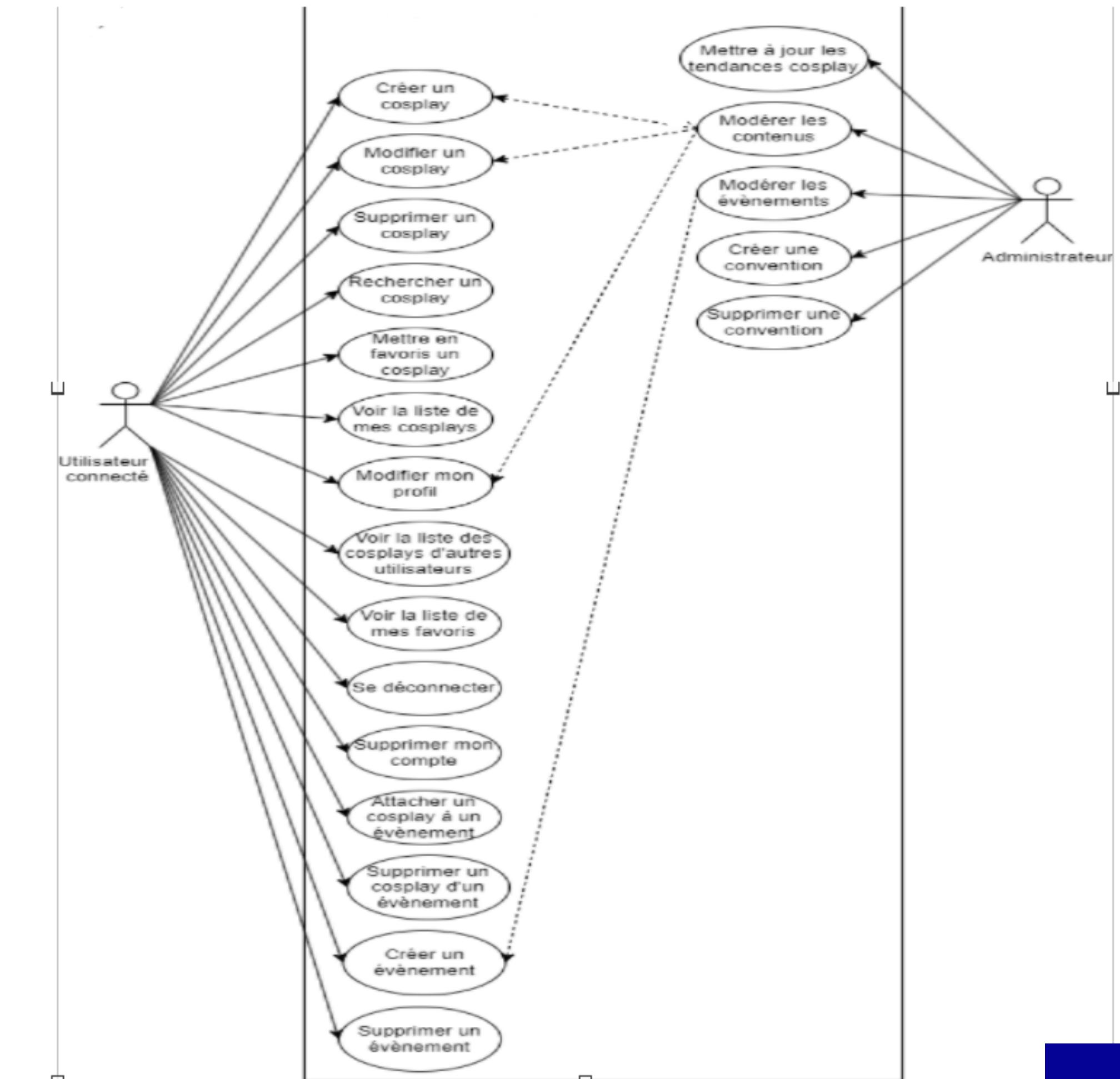


Diagramme de Contexte - Niveau 0

Vue d'ensemble des flux de données et interactions au sein de l'application Cosplay-Maker.

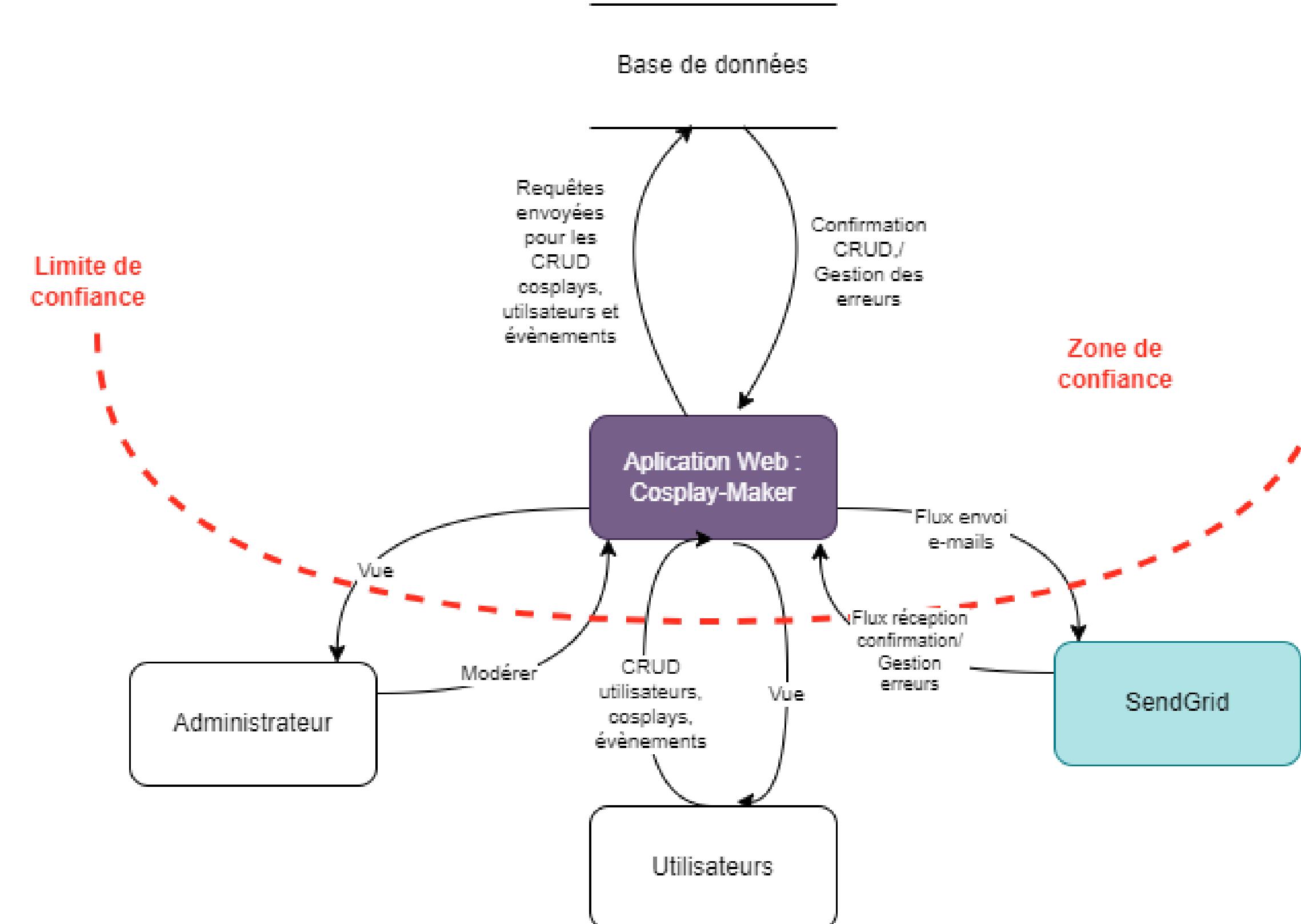
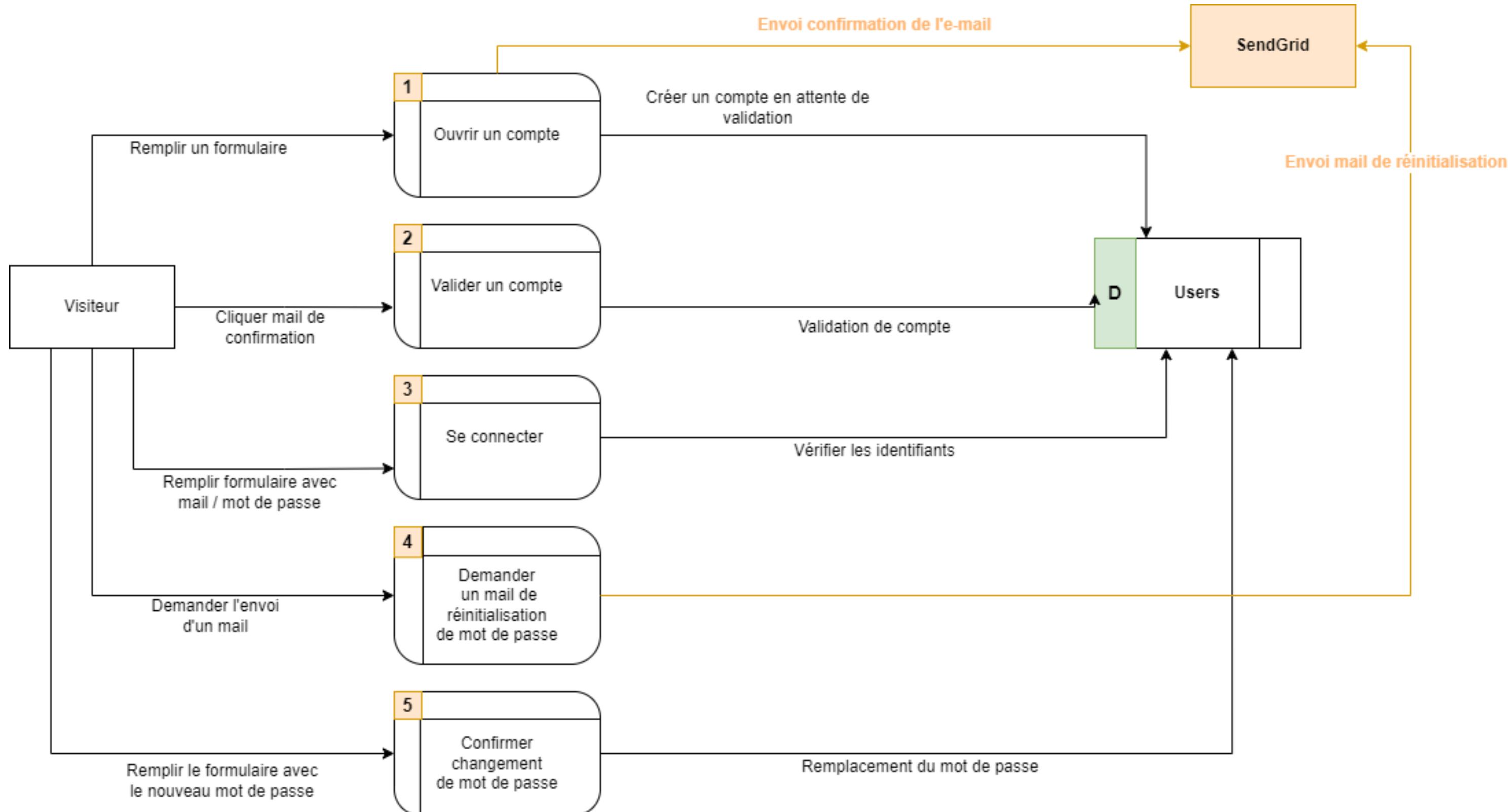
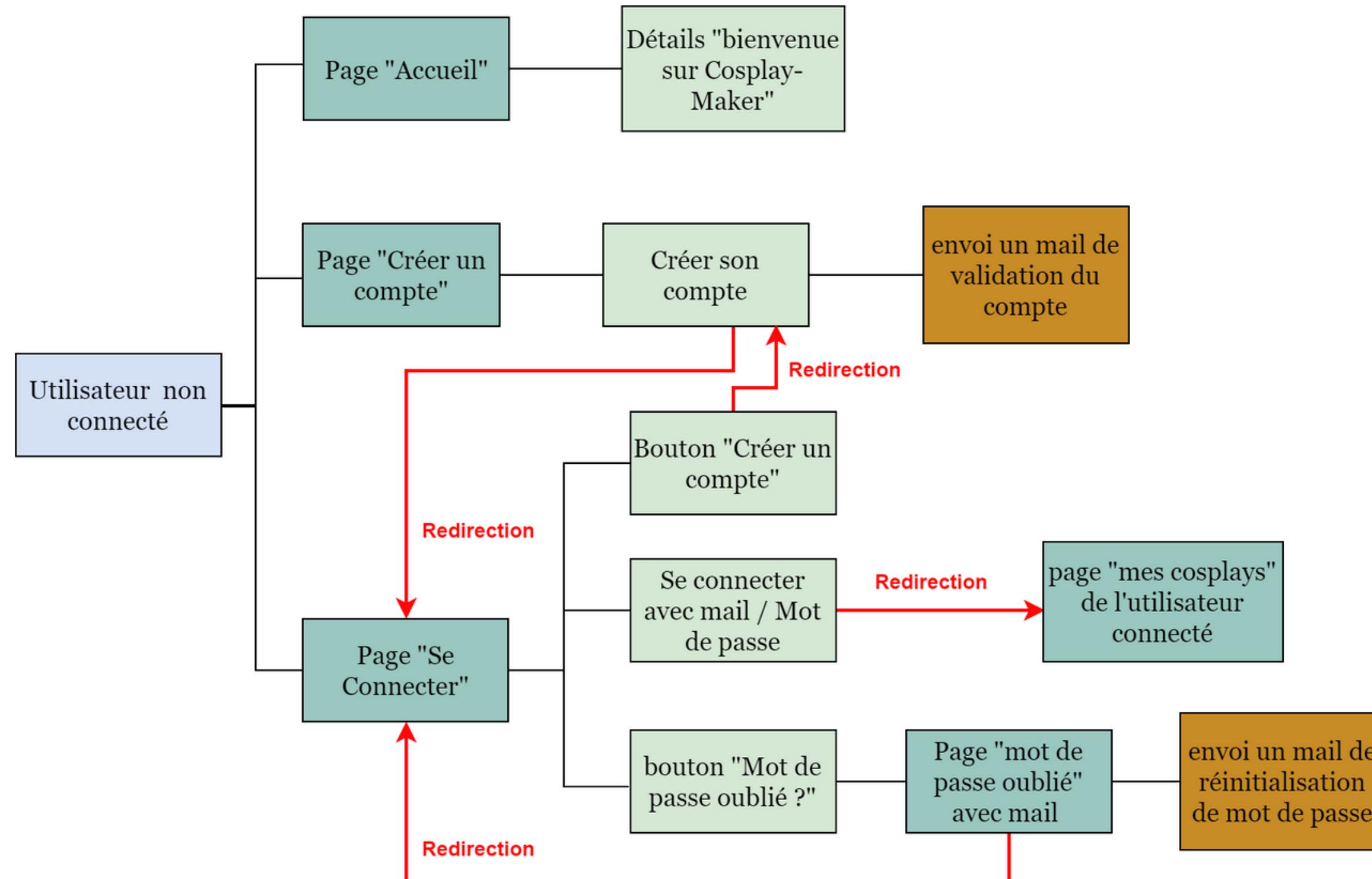


Diagramme de flux de données - niveau 1



Vue de la navigation entre les pages

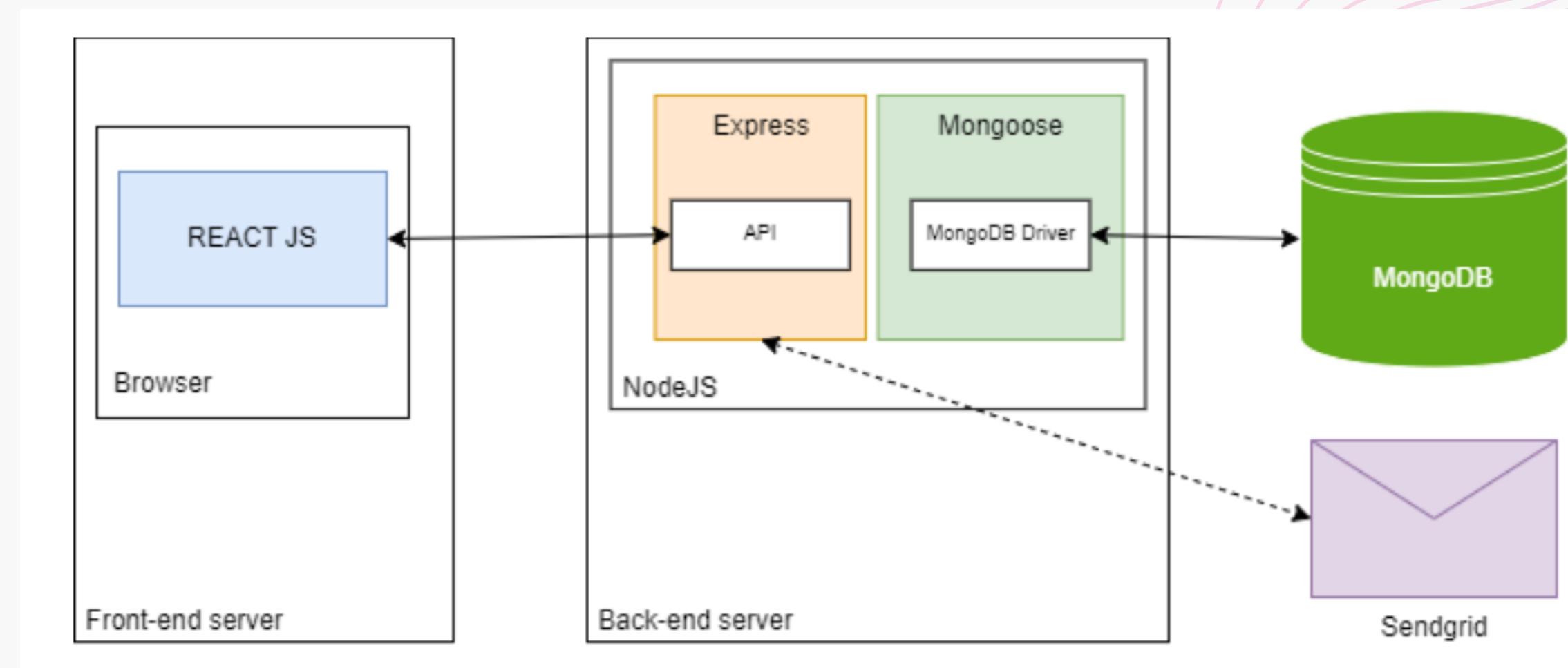


PARTIE CONCEPTION ET ARCHITECTURE DE L'APPLICATION

Architecture du projet

Partie “front“ :

- "pages": fichiers ".tsx" et ".css"
- "icons": les icônes SVG utilisées
- "components"
- "services"



Partie “back“ :

- "features": avec : ".controller.js", ".model.js" et ".route.js".
- "middlewares": traitement des requêtes.
- "utils" et "config"

Choix techniques

Base de données : MongoDB

- NoSQL
- Théorème de Brewer (AP)
- Flexibilité des schémas MongoDB

Back-end : NodeJS

- JavaScript sur client et serveur
- Moteur d'exécution V8
- Nature asynchrone

Back-end : Express

- API RESTFUL de l'application
- Facilite création de routes, gestion des requêtes et réponses.
- Middlewares

Front-end : React avec TypeScript

- Découpage de l'interface en composants réutilisables
- Typage statique grâce à TypeScript
- Autocomplétions avec VS Code
- Facilite le débogage
- Améliore la maintenabilité du code

Quelques librairies et logiciels

Ant Design

- Documentation claire avec exemples
- Bibliothèque de composants
- Accessibilité très bonne
- Librairie régulièrement mise à jour

SendGrid

- Fiabilité (taux de disponibilité de 99.99%).
- E-mails transactionnels
- Facilité d'intégration
- Offre gratuite avec l'envoi de 100 e-mails par jour

Masonry.js

- Mise en page en grille avec des éléments de taille variable
- Disposition dynamique et réactive des éléments
- CSS 'position:absolute', optimisation de l'espace disponible

Conception de la base de données

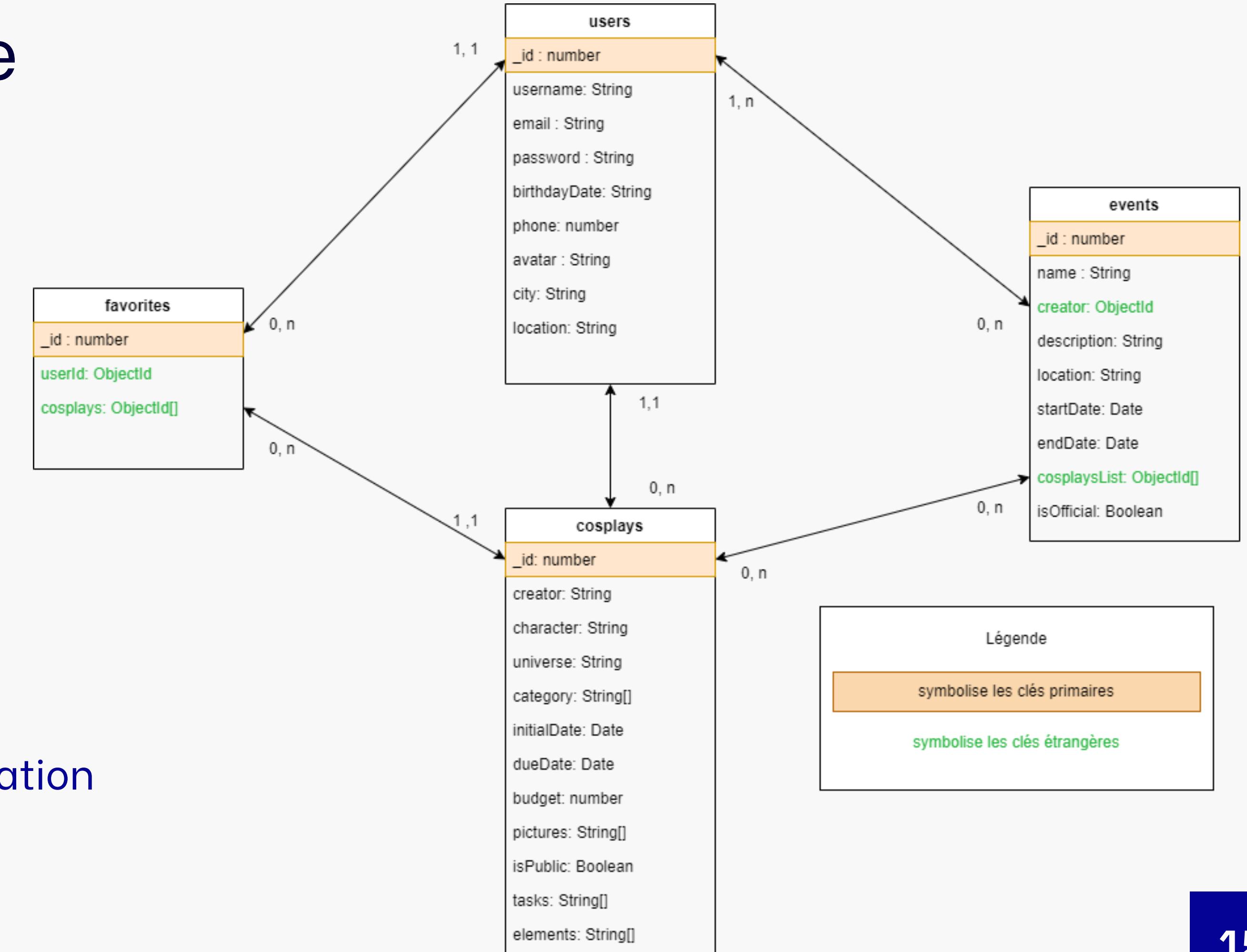


Diagramme entité-relation de la base de données

Dictionnaire de données

Entité : users

Propriété	Description	Nature	Type	Taille	Observation
_id	Identifiant unique de l'utilisateur	N	NC	12	Clé primaire, unique
username	Pseudo de l'utilisateur	AN	NC		Unique
city	Ville de l'utilisateur	AN	NC		
email	Adresse e-mail de l'utilisateur	AN	NC		Unique
password	Mot de passe de l'utilisateur	AN	NC		Unique, hashé
birthdayDate	Date de naissance de l'utilisateur	Date	NC		
phone	Téléphone de l'utilisateur	N	NC		
avatar	Image du profil de l'utilisateur	AN	NC		

MAQUETTE ET CHARTE GRAPHIQUE



Charte graphique

Avec l'application **Web Coloors**

- Vérification du contraste
- Précision des couleurs pour chaque composant

```
const themeCustomValues: ThemeConfig = {
  token: {
    colorPrimary: "#FBBA7B",
    wireframe: false,
    colorPrimaryBg: "#FBBA7B",
    colorPrimaryBgHover: "#FFF",
    borderRadius: 20,
  },
  algorithm: darkAlgorithm,
};
```

Dark Purple	#252235 Couleur de l'arrière plan
Delft Blue	#413B71 Couleur de l'arrière plan
Fawn	#FBBA7B Couleur primaire du site : boutons, au survol de la souris, en mode "actif"
Night	#141414 Composants, cards, sous-menus, tableaux
Poppy	#DC4446 Bouton de suppression
White	#FFFFFF textes, lignes de séparation navbar, body, footer

Maquette

Avec l'application **Web Figma**

Cosplay Maker

Découvrir Mes cosmplays +

Catégorie

€ Prix total

Date de début - Date de fin

Eléments

- élément 1 URL 10 €
- élément 2 URL 10 €

Tâches

- tâche 1
- tâche 2

Nom du cosplay - univers

Nom du créateur

Cosplay Maker

Découvrir Mes cosmplays +

Mes cosmplays

Nom du cosplay
Univers Nom de l'univers
Par cet utilisateur

Nom du cosplay
Univers Nom de l'univers
Par cet utilisateur

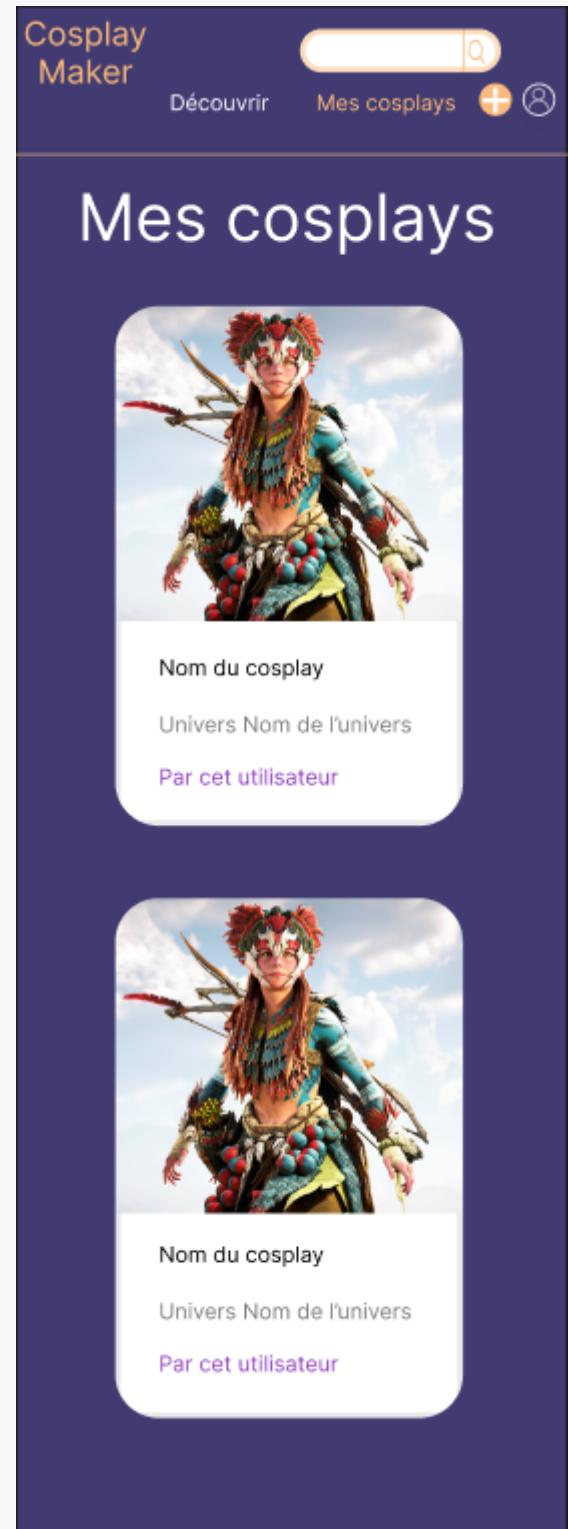
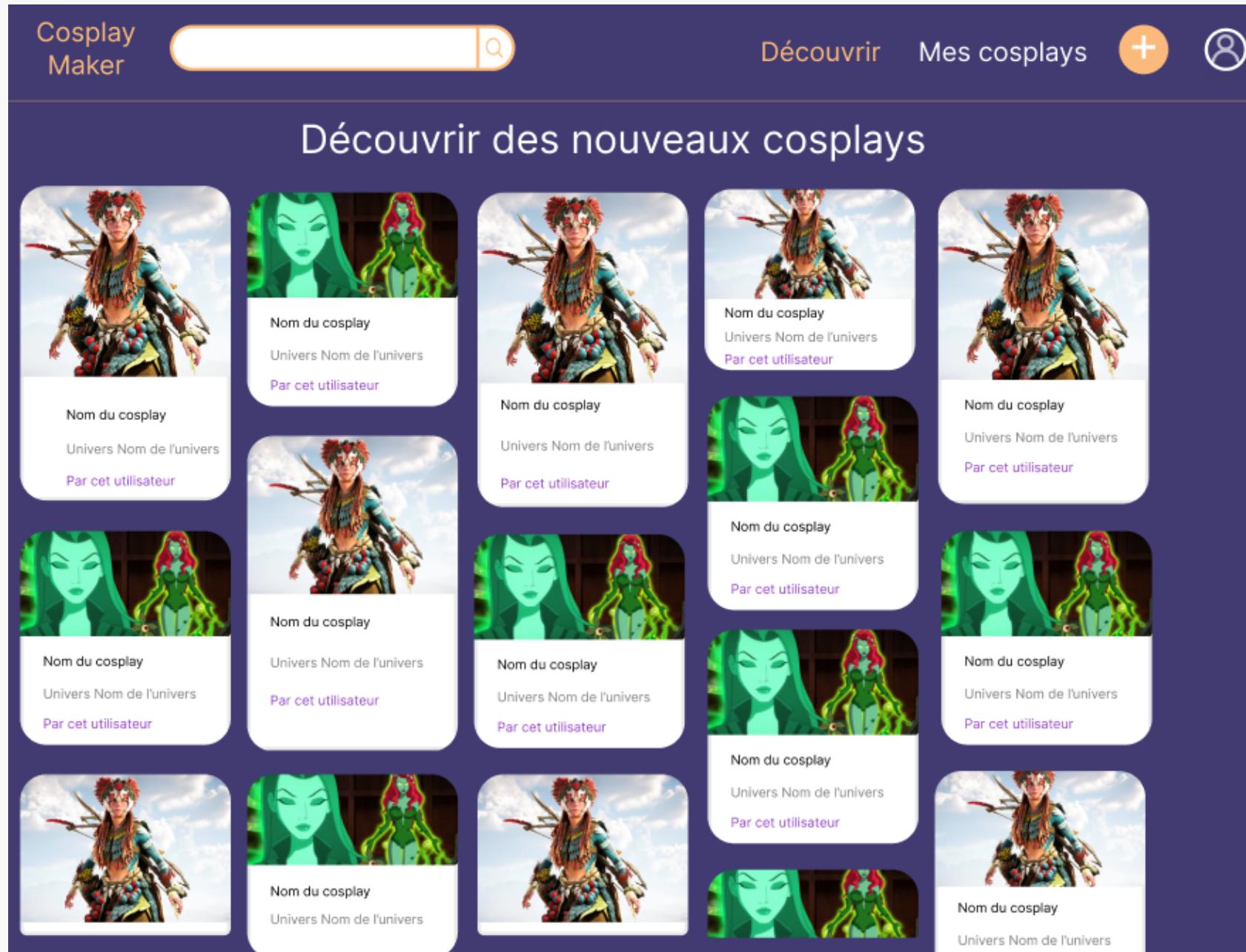
Nom du cosplay
Univers Nom de l'univers
Par cet utilisateur

Nom du cosplay
Univers Nom de l'univers
Par cet utilisateur

Nom du cosplay
Univers Nom de l'univers
Par cet utilisateur

Maquette

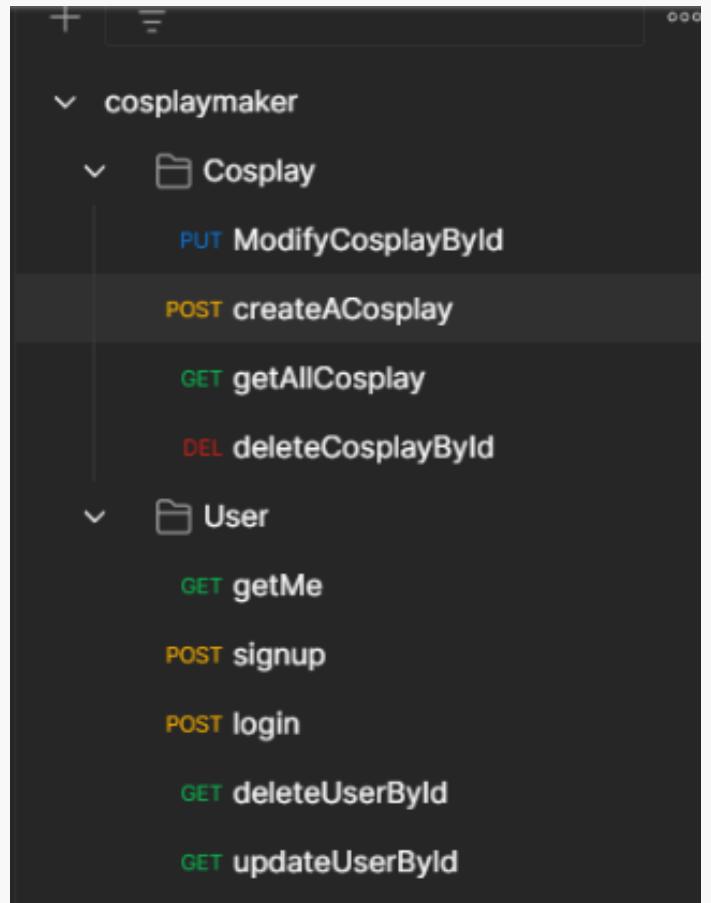
Avec l'application **Web Figma**



ENVIRONNEMENT DE TRAVAIL

Outils utilisés

Postman



Visual Studio Code

- Modulable grâce aux extensions
- Gratuit
- Facile d'utilisation
- multi-langages

MongoDB Compass

- Vérification des schémas
- Modifications manuelles
- Vérification manuelle des données

Docker Desktop

- Conteneur MongoDB lancé localement

tasks.json dans le .vscode

```
tasks.json x
.vscode > tasks.json > ...
1  [
2    {
3      "version": "2.0.0",
4      "tasks": [
5        {
6          "label": "start-back",
7          "type": "shell",
8          "command": "npm run dev",
9          "options": {
10            "cwd": "${workspaceFolder}/back"
11          },
12          "problemMatcher": []
13        },
14        {
15          "label": "start-front",
16          "type": "shell",
17          "command": "npm run start",
18          "options": {
19            "cwd": "${workspaceFolder}/front"
20          },
21          "problemMatcher": []
22        },
23        {
24          "label": "start-all",
25          "dependsOn": ["start-back", "start-front"],
26          "problemMatcher": []
27        }
28      ]
29    }
]
```

Github

- Versionner le code, historique
- Partager mon code

Github Actions

Les étapes du workflow

- Vérifier et récupérer les fichiers du référentiel.
- Configurer l'environnement Node.js en utilisant la version 18.15.
- Installer les dépendances du projet dans le dossier "back"
- Transpiler les sources et tester la partie back
- Installer les dépendances du projet dans le dossier "front"
- Transpiler les sources et tester la partie front

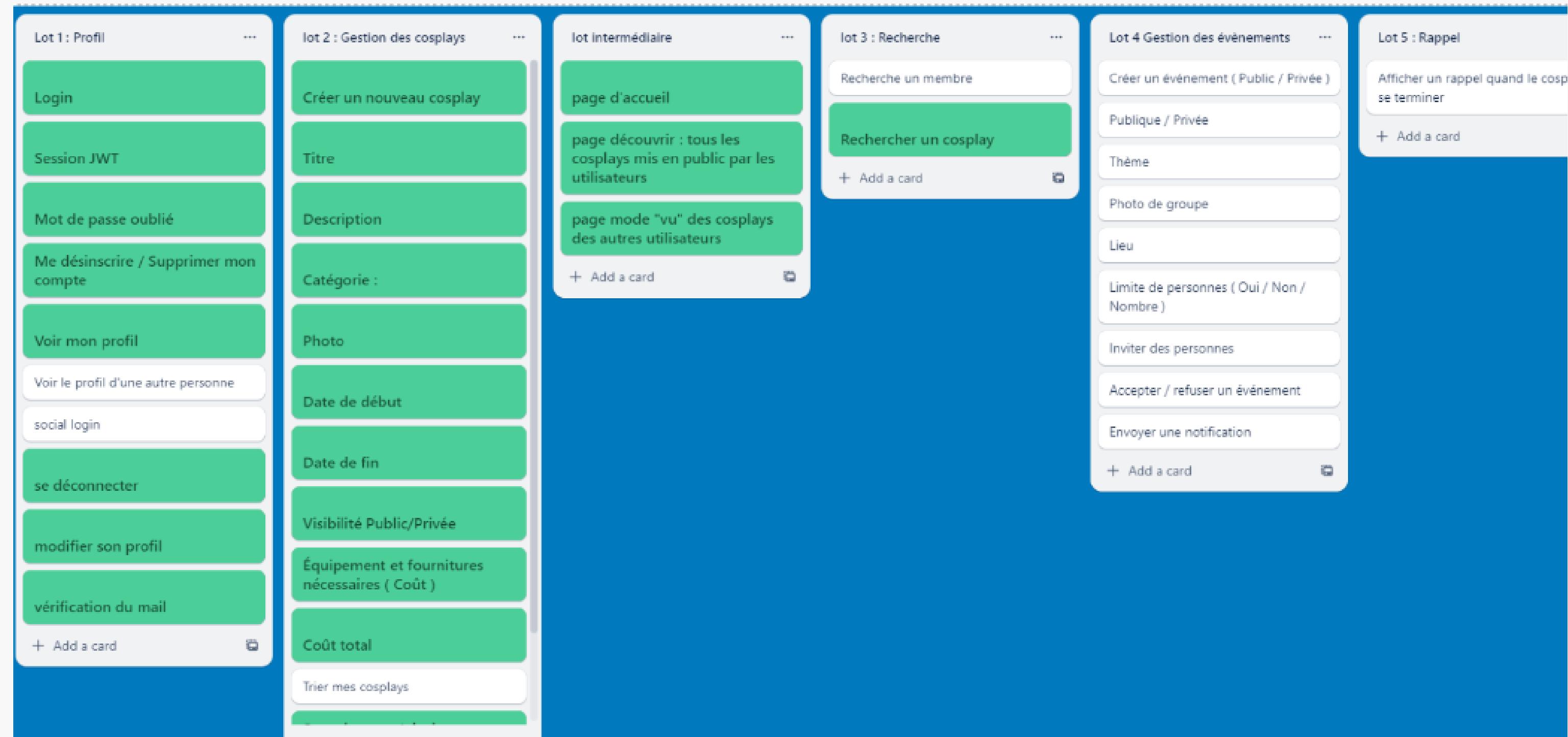
```

1 name: CI
2
3 on:
4   push:
5     branches:
6       - "*"
7
8 jobs:
9   build-and-test-back:
10    runs-on: ubuntu-latest
11
12 steps:
13   - uses: actions/checkout@v3
14   - uses: actions/setup-node@v3
15     with:
16       node-version: "18.15"
17       cache: "npm"
18       cache-dependency-path: /
19         back/package-lock.json
20         front/package-lock.json
21
22   - name: Install dependencies ( back )
23     working-directory: back
24     run: |
25       npm install
26
27   - name: Build and test ( back )
28     working-directory: back
29     run: |
30       npm run test
31
32   - name: Install dependencies ( front )
33     working-directory: front
34     run: |
35       npm install
36
37   - name: Build and test ( front )
38     working-directory: front
39     run: |
40       npm run test

```

Trello

- tableau Kanban
- fonctionnement par lots



PARTIE BACK-END

Les routes

```
// CREATE
// Créer un cosplay dans la BDD
router.post("/cosplay/create", checkAuth, fileUpload, createACosplay);

// UPDATE
// Mettre à jour un cosplay
router.put("/cosplay/:id", checkAuth, fileUpload, updateACosplay);

// Mettre à jour un cosplay avec une ou plusieurs images
router.post("/update/:id", checkAuth, referencesUpload, addImage);

// READ
// Obtenir tous les cosplays sans filtre
router.get("/cosplay", checkAuth, searchCosplays);

// Récupérer tous mes cosplays
router.get("/cosplays/me", checkAuth, getAllMyCosplays);

// Obtenir un cosplay selon son Id
router.get("/cosplay/:id", checkAuth, getCosplayById);

// DELETE
// Supprimer un cosplay selon son Id
router.delete("/cosplay/:id", checkAuth, deleteCosplayById);

// Supprimer une image d'un cosplay
router.put("/cosplay/:id/delete-picture", checkAuth, deleteImage);

module.exports = router;
```

Le modèle

```
1 const mongoose = require("mongoose");
2 const Schema = mongoose.Schema;
3
4 // Définition du schéma du modèle "Cosplay"
5 const cosplaySchema = new Schema({
6   character: String,
7   universe: String,
8   initialDate: Date,
9   dueDate: Date,
10  budget: Number,
11  category: String,
12  variant: String,
13  elements: [String],
14  status: {
15    type: String,|      Cassandra Forestier, 2 months ago * expand user infos inside cosplay using aggregatio...
16    enum: ["In progress", "Planned", "Finished"],
17    default: "Planned",
18  },
19  pictures: [String],
20  creator: { ref: "User", type: mongoose.Schema.Types.ObjectId },
21  public: Boolean,
22  tasks: [String],
23});
24
25 // Création du modèle "Cosplay" à partir du schéma défini
26 const Cosplay = mongoose.model("Cosplay", cosplaySchema);
27
28 // L'index du schema cosplay permet de faire des recherches textuelles sur les champs character et universe
29 // Par défaut, Mongo créé un index sur l'id, mais pas sur les champs textuels
30 // la création d'un index permet l'optimisation des requêtes car Mongo peut utiliser l'index pour accélérer les recherches
31 cosplaySchema.index({ character: "text", universe: "text" });
32 module.exports = Cosplay;
```

Un exemple de contrôleur : suppression d'un cosplay

```
function deleteCosplayById(req, res) {
  const id = req.params["id"];
  const creator = req.user._id;

  Cosplay.findOne({ _id: id }, (err, cosplay) => {
    if (!cosplay) return res.status(404).json({ msg: "Cosplay non trouvé" });
    if (!cosplay.creator.equals(creator)) {
      return res
        .status(403)
        .json({ msg: "Vous n'êtes pas le créateur de ce cosplay" });
    }
    Cosplay.deleteOne({ _id: id }, (err, cosplayToDelete) => {
      if (!cosplayToDelete)
        return res.status(404).json({ msg: "Cosplay non trouvé" });
      if (err) {
        console.error(err);
        res.status(500).json({
          error: "Une erreur est survenue pendant la recherche du cosplay",
        });
      } else {
        res.status(200).json({ msg: "Cosplay supprimé" });
      }
    });
  });
}
```

Middleware : checkAuth

```
async function checkAuth(req, res, next) {
  try {
    const { cookies, headers } = req;
    /* On vérifie que Le JWT est présent dans Les cookies de La requête */
    if (!cookies || !cookies.access_token) {
      return res.status(401).json({ message: "Missing token in cookie" });
    }

    const accessToken = cookies.access_token;

    /* On vérifie que le token CSRF est présent dans Les en-têtes de La requête */
    if (!headers || !headers["x-xsrf-token"]) {
      return res.status(401).json({ message: "Missing XSRF token in headers" });
    }

    const xsrfToken = headers["x-xsrf-token"];

    /* On vérifie et décode Le JWT à L'aide du secret et de L'algorithme utilisé pour Le générer */
    const decodedToken = jwt.verify(accessToken, secret, {
      algorithms: algorithm,
    });

    /* On vérifie que Le token CSRF correspond à celui présent dans Le JWT */
    if (xsrfToken !== decodedToken.xsrfToken) {
      return res.status(401).json({ message: "Bad xsrf token" });
    }

    /* On vérifie que L'utilisateur existe bien dans notre base de données */
    const userId = decodedToken.sub;
    const user = await User.findOne({ _id: userId });

    if (!user) {
      return res.status(401).json({ message: `User ${userId} not exists` });
    }

    /* On passe L'utilisateur dans notre requête afin que celui-ci soit disponible pour Les prochains middlewares */
    req.user = user;

    /* On appelle Le prochain middleware */
    return next();
  } catch (err) {
    console.error(err);
    return res.status(500).json({ message: "Internal error" });
  }
}
```

Middleware : fileUpload

```
const multer = require("multer");
const path = require("path");

const storage = multer.diskStorage({
    // Permet de donner la destination des fichiers
    destination: (req, file, callback) => {
        callback(null, path.join(__dirname, "../uploads/"));
    },

    // Permet de formatter le nom de fichier
    filename: (req, file, callback) => {
        callback(
            null,
            file.fieldname + "-" + Date.now() + path.extname(file.originalname)
        );
    },
    // Permet de filtrer les fichiers pour n'accepter que les photos
    filefilter: (req, file, callback) => {
        const ext = path.extname(file.originalname);
        if (ext !== ".png" && ext !== ".jpeg" && ext !== ".jpg") {
            return callback(new Error("Seulement les images sont autorisées."));
        }
        callback(null, true);
    },
});

const upload = multer({ storage: storage });
const fileUpload = upload.fields([
    {
        name: "image-file",
        maxCount: 1,
    },
]);
const avatarUpload = upload.fields([
    {
        name: "avatar",
        maxCount: 1,
    },
]);
const referencesUpload = upload.fields([
    {
        name: "pictures",
        maxCount: 1,
    },
]);
module.exports = { fileUpload, avatarUpload, referencesUpload };
```

CAS DE RECHERCHE



Cas de recherche



Problématique

Effectuer une recherche **globale et efficace** sur la collection "Cosplay"



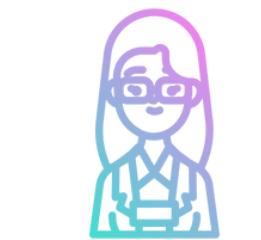
Objectifs

Couvrir les champs "characters" et "universe"



Contrainte

Insensibilité à la casse pour des résultats complets



Exigence

Implémentation évolutive

Méthodologie

1. **Implémentation initiale** : Deux requêtes séparées pour "characters" et "universe"
 - **Utilisation de la fonction "populate"** : Remplacer l'id du champ "creator" par l'utilisateur complet
2. **Optimisation avec l'opérateur "\$or"** : Fusionner les deux requêtes en utilisant l'opérateur "\$or"
3. **Découverte des alternatives** : Recherche sur l'aggregation pipeline et la fonctionnalité full text search
 - **Avantages de l'index de texte** : Performances élevées
 - **Limitations de l'opérateur "\$text"** : Pas de support pour les correspondances partielles
 - **Avantages du pipeline d'agrégation** : Optimisation automatique des étapes d'aggrégation

Résolution

```
function searchCosplays(req, res) {
  // Si il n'y a pas de paramètre "q" dans la requête, on retourne tous les cosplays
  // car l'onglet "Découvrir" est sélectionné
  if (!req.query.q) return getAllCosplays(req, res);
  const query = req.query.q; // Récupère la valeur du paramètre "q" dans la requête

  Cosplay.aggregate(
    aggregatePopulateCreator({
      $match: {
        $text: {
          $search: query,
          $caseSensitive: false,
          $diacriticSensitive: false,
        },
      },
    })
  ).exec(function (err, data) {
    if (err) {
      console.log(err);
      return res.send(500).json({ msg: "cannot query cosplays" });
    }
    return res.status(200).json(data);
  });
}
```

```
31
32  // l'index du schema cosplay permet de faire des recherches textuelles sur les champs character et universe
33  // Par défaut, Mongo crée un index sur l'id, mais pas sur les champs textuels
34  // la création d'un index permet l'optimisation des requêtes car Mongo peut utiliser l'index pour accélérer les re
35  cosplaySchema.index({ character: "text", universe: "text" });
36  module.exports = Cosplay;
```

```
/** Cassandra Forestier, 2 months ago * expand user infos inside cosplays */
* @param {Object} matchCriteria i.e { $match: { creator: '123' } } */
* @returns
*/
const aggregatePopulateCreator = (matchCriteria) => {
  return [
    matchCriteria,
    {
      $lookup: {
        from: "users",
        localField: "creator",
        foreignField: "_id",
        as: "creator",
      },
    },
    {
      $unwind: {
        path: "$creator",
        preserveNullAndEmptyArrays: true,
      },
    },
    {
      $project: {
        _id: 1,
        character: 1,
        universe: 1,
        initialDate: 1,
        dueDate: 1,
        budget: 1,
        category: 1,
        variant: 1,
        elements: 1,
        status: 1,
        pictures: 1,
        tasks: 1,
        "creator._id": 1,
        "creator.username": 1,
      },
    },
  ];
};

module.exports = aggregatePopulateCreator;
...
```

PARTIE FRONT-END

React Router Dom

- React Router DOM => gestion des routes
- App contient un ConfigProvider
- Les routes : chemins et pages associées
- App dans BrowserRouter

```
const root = ReactDOM.createRoot(  
  document.getElementById("root") as HTMLElement  
);  
root.render(  
  <BrowserRouter>  
    <App />  
  </BrowserRouter>  
);
```

```
return (  
  <ConfigProvider  
    locale={frFR}  
    theme={themeCustomValues}  
    form={{ validateMessages }}  
  >  
  <Routes>  
    <Route path="/" element={<Layout isLoggedIn={isLoggedIn} />}>  
    <Route index element={<Home />} />  
    <Route path="cosplay" element={  
      <ProtectedRoute isLoggedIn={isLoggedIn}>  
        <CreateCosplay />  
      </ProtectedRoute>  
    } />  
    <Route  
      path="showCosplays"  
      element={  
        <ProtectedRoute isLoggedIn={isLoggedIn}>  
          <ShowAllCosplay onlyMyCosplays={true} />  
        </ProtectedRoute>  
      } />  
    <Route path="searchResults" element={  
      <ProtectedRoute isLoggedIn={isLoggedIn}>  
        <SearchCosplay />  
      </ProtectedRoute>  
    } />  
    <Route  
      path="allCosplays"  
      element={  
        <ProtectedRoute isLoggedIn={isLoggedIn}>  
          <ShowAllCosplay onlyMyCosplays={false} />  
        </ProtectedRoute>  
      } />  
    <Route path="profil" element={  
      <ProtectedRoute isLoggedIn={isLoggedIn}>  
        <User />  
      </ProtectedRoute>  
    } />
```

Layout et Outlet

Le Layout :

- Chargé par défaut lors de la visite

L'Outlet :

- Point d'entrée pour le rendu des composants enfants en fonction des routes

```
return (
  <>
    <div className="navbar">
      <Link to="/">
        <img alt="logo" src={image} className="logo" />
      </Link>

      <div className="menu-general">
        {props.isLoggedIn ? (
          <Search
            placeholder="Rechercher un cosplay"
            allowClear
            onSearch={(fileList: string) => handleSearch(fileList)}
            className="search"
          />
        ) : null}
        <Menu
          onClick={onClick}
          selectedKeys={[current]}
          mode="horizontal"
          items={props.isLoggedIn ? privateItems : publicItems}
          className="menu"
        />
      </div>
    </div>
    <Outlet />
    <div className="clear"></div>
    <div>Cassandra Forestier, 3 months ago</div>
    <div>Cosplay Maker ©2023 Créeée par Neytiria_</div>
  </div>
);
```

layout.tsx

```
const publicItems: MenuProps["items"] = [
  {
    label: <Link to="/">Accueil</Link>,
    key: "Home",
  },
  {
    label: <Link to="/signup">Créer un compte</Link>,
    key: "createAccount",
    className: "menu-item-create",
  },
  {
    label: <Link to="/login">Se connecter</Link>,
    key: "loggin",
  },
];
You, 4 months ago • add search input in the menu
function handleSearch(value: string) {
```

Les fichiers de service

- Requêtes centralisées vers l'API (modularité du code et éviter les duplications)
- Instance personnalisée d'Axios : gérer l'expiration du token d'accès et son renouvellement

```
export class CosplayService {  
  
    static async getAllCosplays(): Promise<Cosplay[]> {  
        const result = await axiosApiInstance.get<Cosplay[]>( ` ${DOMAIN_URL}/api/cosplay` , {  
            withCredentials: true,  
            headers: prepareHeaders(true),  
        });  
        return result.data;  
    }  
}
```

```
const { Paragraph } = Typography;  
const { Meta } = Card;  
type ShowAllCosplayProps = {  
    onlyMyCosplays?: boolean;  
};  
  
function ShowAllCosplay(props: ShowAllCosplayProps) {  
  
    const { onlyMyCosplays } = props;  
    const [data, setData] = useState<Cosplay[]>([]);  
    const breakpointColumnsObj = { ... };  
  
    const userId = localStorage.getItem("userId");  
    const navigate = useNavigate();  
    const redirectToCosplay = (cosplay: Cosplay) => {  
        navigate(` /cosplayDetails/${cosplay._id}/${hasCosplayUser(cosplay, userId) ? "edit" : "view"}`);  
    };  
  
    useEffect(() => {  
        const fetchData = async () => {  
            const result = onlyMyCosplays  
                ? await CosplayService.getMyCosplays()  
                : await CosplayService.getAllCosplays();  
            setData(result);  
        };  
        fetchData();  
    }, [onlyMyCosplays]);  
}  
| You, 6 months ago • Add user CRUD, add Passport for authentication ...
```

Composant React

- Utilisation des composants Ant Design
- Utilisation de règles "required: true" pour certains champs du formulaire
- Utilisation de className pour les styles dans la feuille create-cosplay.css associée
- Utilisation d'interface pour le typage avec TypeScript

```
return (
  <>
    <Title level={2} className="create-cosplay-title">
      Nouveau cosplay
    </Title>
    <Form
      {...layout}
      labelCol={{ span: 6 }}
      wrapperCol={{ span: 16 }}
      form={form}
      className="create-cosplay-form"
      name="control-hooks"
      onFinish={onFinish}
    >
      <Form.Item name="status" label="Statut" rules={[{ required: true }]}>
        <Radio.Group>
          <Radio value="In progress"> En cours </Radio>
          <Radio value="Planned"> Planifié </Radio>
        </Radio.Group>
      </Form.Item>
      <Form.Item
        name="character"
        label="Personnage"
        rules={[{ required: true }]}
      >
    
```

Cassandra Forestier, 2 months ago | 1 author (Cassandra Forestier)
`import { UploadChangeParam } from "antd/es/upload";`

Cassandra Forestier, 2 months ago | 1 author (Cassandra Forestier)
`export interface CreateCosplayFormValues {
 status: "En cours" | "Plan";
 character: string;
 universe: string;
 variant: string;
 category: string;
 budget: number;
 pictures: UploadChangeParam<string | Blob>;
}`

axiosApinstance

- Instance personnalisée d'Axios
- Intercepteur gérant les erreurs d'expiration du token d'accès (401 Unauthorized)
- Tentative de rafraîchissement du token en appelant UserService.refreshToken()
- Stockage des nouveaux tokens dans le localStorage.
- Renvoi de la requête d'origine avec les nouveaux tokens pour retraitement.
- En cas d'échec du rafraîchissement, redirection vers la page de connexion avec un indicateur de déconnexion.

```
Cassandra Forestier, 3 months ago | 1 author (Cassandra Forestier)
1 import axios from "axios";
2 import { UserService } from "./services/user.service";
3
4 const axiosApiInstance = axios.create();
5
6 axiosApiInstance.interceptors.response.use(
7   function (response) {
8     return response;
9   },
10  async function (error) {
11    const originalRequest = error.config;
12    if (error.response.status === 401 && !originalRequest._retry) {
13      originalRequest._retry = true;
14      try {
15        const res = await UserService.refreshToken();
16        const xsrfToken = res.xsrfToken;
17        const expireDate = new Date();
18        expireDate.setMilliseconds(
19          expireDate.getMilliseconds() + res.accessTokenExpiresIn
20        );
21        localStorage.setItem("accessTokenExpiresIn", expireDate.toString());
22        localStorage.setItem("xsrfToken", xsrfToken);
23        originalRequest.headers["x-xsrf-token"] = xsrfToken;
24      } catch (err) {
25        window.location.href = "/login?disconnected=true";
26        return Promise.reject(err);
27      }
28      return axiosApiInstance(originalRequest);
29    }
30    return Promise.reject(error);
31  });
32
33 export default axiosApiInstance;
```

Composant réutilisable

- Il s'agit d'un composant qui est réutilisé sur plusieurs pages :
 - Edition du cosplay
 - sur les éléments
 - sur les tâches
 - sur les informations générales
 - page "mes informations"
- Lui-même injecté dans un composant réutilisé "table-editable-field"
- Non duplication de code
- Modularité du code
- Caractère "réutilisable" de React

```
interface EditableFieldProps {  
  type?: string;  
  placeholder: string;  
  defaultValue: string | number | undefined;  
  onSubmit: (field: string, value: string) => void;  
  name: string;  
  validationMessage: string;  
 isRequired?: boolean;  
}  
  
export const EditableField = (props: EditableFieldProps): JSX.Element => {  
  const {  
    type,  
    placeholder,  
    defaultValue,  
    onSubmit,  
    name,  
    validationMessage,  
    isRequired = true,  
  } = props;  
  
  const [isEditing, setIsEditing] = useState(false);  
  return isEditing ? (  
    <Form  
      initialValues={{ [name]: defaultValue }}  
      onFinish={({ values }) => {  
        onSubmit(name, values[name]);  
        setIsEditing(false);  
      }}  
      layout="inline"  
    >  
    <Form.Item  
      rules={[{ required: isRequired, message: validationMessage }]}  
      name={name}  
    >  
      <Input  
        placeholder={placeholder}  
        defaultValue={defaultValue}  
        autoFocus  
      />  
    </Form.Item>  
    <Form.Item>  
      <Button htmlType="submit">Enregistrer</Button>  
      <Button onClick={() => setIsEditing(false)}>Annuler</Button>  
    </Form.Item>  
  </Form>  
);  
Cassandra Forestier, 3
```

Composant “gardien”

- Sécuriser les routes front :

Si "non connecté" -> redirection vers la page de login

```
interface ProtectedRouteProps {  
  isLoggedIn: boolean;  
  children: JSX.Element;  
}  
  
const ProtectedRoute = ({ isLoggedIn, children }: ProtectedRouteProps) => {  
  
  if (!isLoggedIn) {  
    return <Navigate to="/login" replace />;  
  }  
  
  return children;  
};  
  
export default ProtectedRoute;
```

TEST SET VALIDATIONS

Les tests avec Jest

- Utilisation dans le front-end et back-end

File	%Stmts	%Branch	%Funcs	%Lines
All files	59.4	59.78	50	59.4
back	77.61	16.66	100	77.61
app.js	77.61	16.66	100	77.61
back/features/auth	100	100	100	100
refreshToken.model.js	100	100	100	100
back/features/cosplay	67.02	70.37	70	67.02
cosplay.controller.js	51.19	69.23	66.66	51.19
cosplay.model.js	100	100	100	100
cosplay.query-helper.js	100	100	100	100
cosplay.route.js	100	100	100	100
back/features/event	50.35	100	0	50.35
event.controller.js	16.66	100	0	16.66
event.model.js	100	100	100	100
event.route.js	100	100	100	100
back/features/user	60.19	53.84	50	60.19
user.controller.js	51.83	53.57	46.15	51.83
user.model.js	87.69	54.54	100	87.69
user.route.js	100	100	100	100
back/middlewares	45.92	70	55.55	45.92
auth.js	38.02	60	33.33	38.02
sgMail.js	42.3	33.33	50	42.3
validator.js	74	100	75	74

```
describe("With valid cosplay and token", () => {
  it("should return 201", async () => {
    const cosplay = {
      variant: "test",
      character: "test",
      universe: "test",
      initialDate: "1111-11-10T23:50:39.000Z",
      dueDate: "1111-11-10T23:50:39.000Z",
      budget: 10,
      category: "test",
      elements: [],
      tasks: [],
      status: "Planned",
    };
    const res = await agent
      .post("/api/cosplay/create")
      .type("form")
      .field("cosplay-data", JSON.stringify(cosplay))
      .attach("image-file", __dirname + "/assets/Cosplay.png")
      .set("Accept", "form-data")
      .set("Cookie", cookies)
      .set("x-xsrf-token", xsrfToken);
    expect(res.status).toEqual(201);
    expect(res.body).toEqual({
      __v: 0,
      _id: expect.any(String),
      ...cosplay,
      pictures: [expect.any(String)],
      creator: expect.any(String),
    });
  });
});
```

Les validations

- Avec la bibliothèque Express Validator
- Valider les données entrées par l'utilisateur
- en tant que middleware sur les routes back

```
/* PUBLIC ROUTES */
router.post("/signup", userValidationRules, signupUser, sgMail);
router.post("/login", logginValidationRules, loginUser);
```

```
const { body, validationResult } = require("express-validator");
function userValidationRules(req, res, next) {
  return validate([
    body("city").isString(),
    body("username").isString(),
    body("email").isEmail(),
    body("password").isLength({ min: 8 }),
    body("birthdayDate").isISO8601().notEmpty(),
  ])(req, res, next);
}

function logginValidationRules(req, res, next) {
  return validate([
    body("email").isEmail(),
    body("password").isLength({ min: 8 }),
  ])(req, res, next);
}

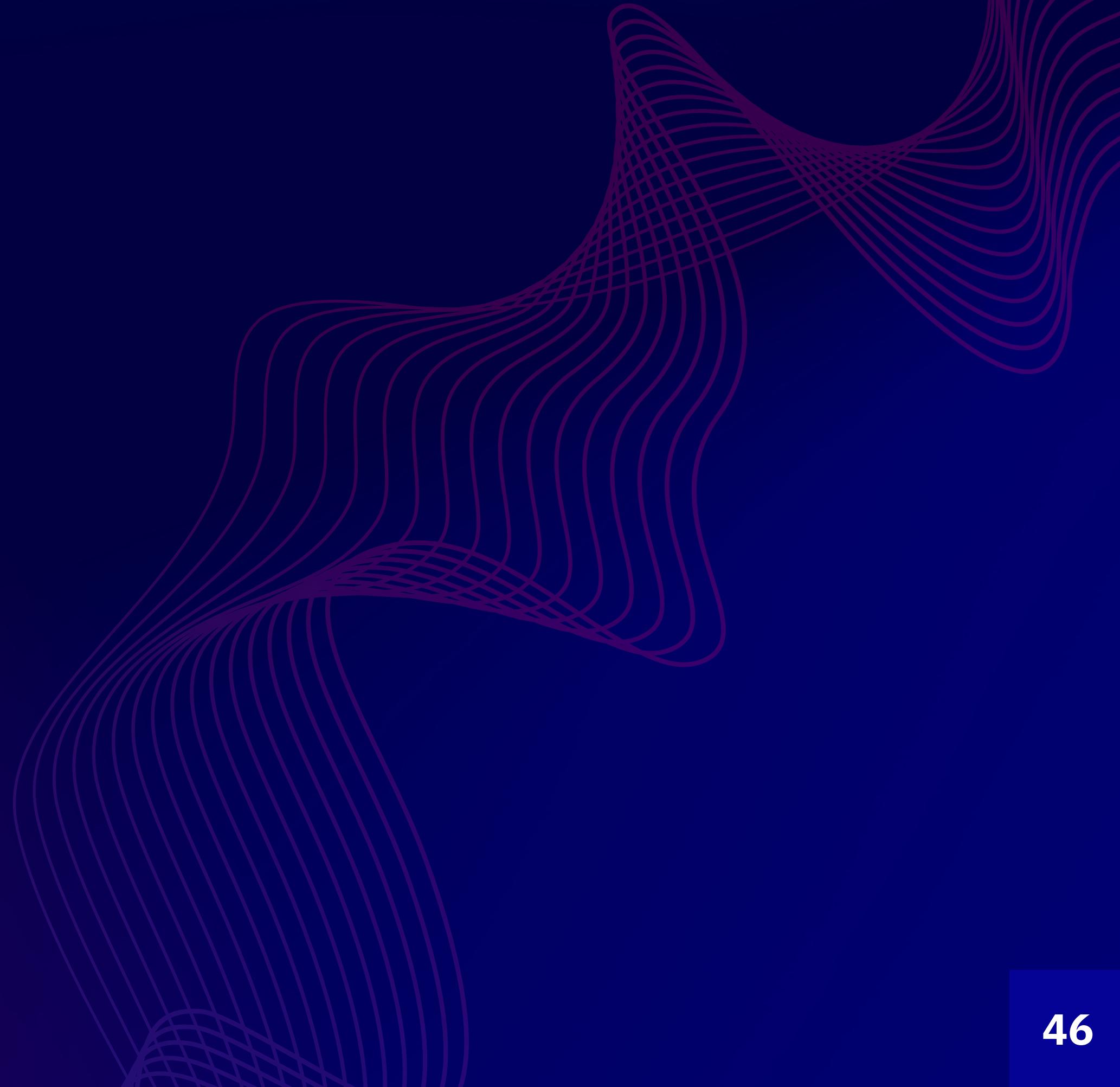
const validate = (validations) => {
  return async (req, res, next) => {
    await Promise.all(validations.map((validation) => validation.run(req)));

    const errors = validationResult(req);
    if (errors.isEmpty()) {
      return next();
    }

    res.status(400).json({ errors: errors.array() });
  };
};

module.exports = {
  userValidationRules,
  validate,
};
```

SÉCURITÉ



La sécurité générale du site

Méthode STRIDE :

- Usurpation d'identité (Spoofing)
- Falsification (Tampering)
- Refus d'assumer (Repudiation)
- Divulgation d'informations (Information disclosure)
- Déni de service (Denial of service)
- Elévation des privilèges (Elevation of privileges)

Utilisation de SNYK :

- Analyser les dépendances et détecter les vulnérabilités connues.
- Intégration de Snyk dans le pipeline de développement pour une analyse continue de la sécurité (SCA).
- Utilisation de la technique d'analyse SAST pour détecter les vulnérabilités dans le code.



Bilan du projet

- Mise en pratique des connaissances acquises
 - Amélioration de la capacité à résoudre des problèmes complexes
 - Apprendre à se former constamment
 - Expérience enrichissante et stimulante
-
- Amélioration du site à l'avenir en ajoutant les features manquantes
 - NSFW.js pour l'ajout des photos (middleware)
 - Application mobile
 - Prévoir des partenariats
 - Liens d'affiliation sur les liens ajoutés sur les éléments



Merci !

Cassandra Forestier