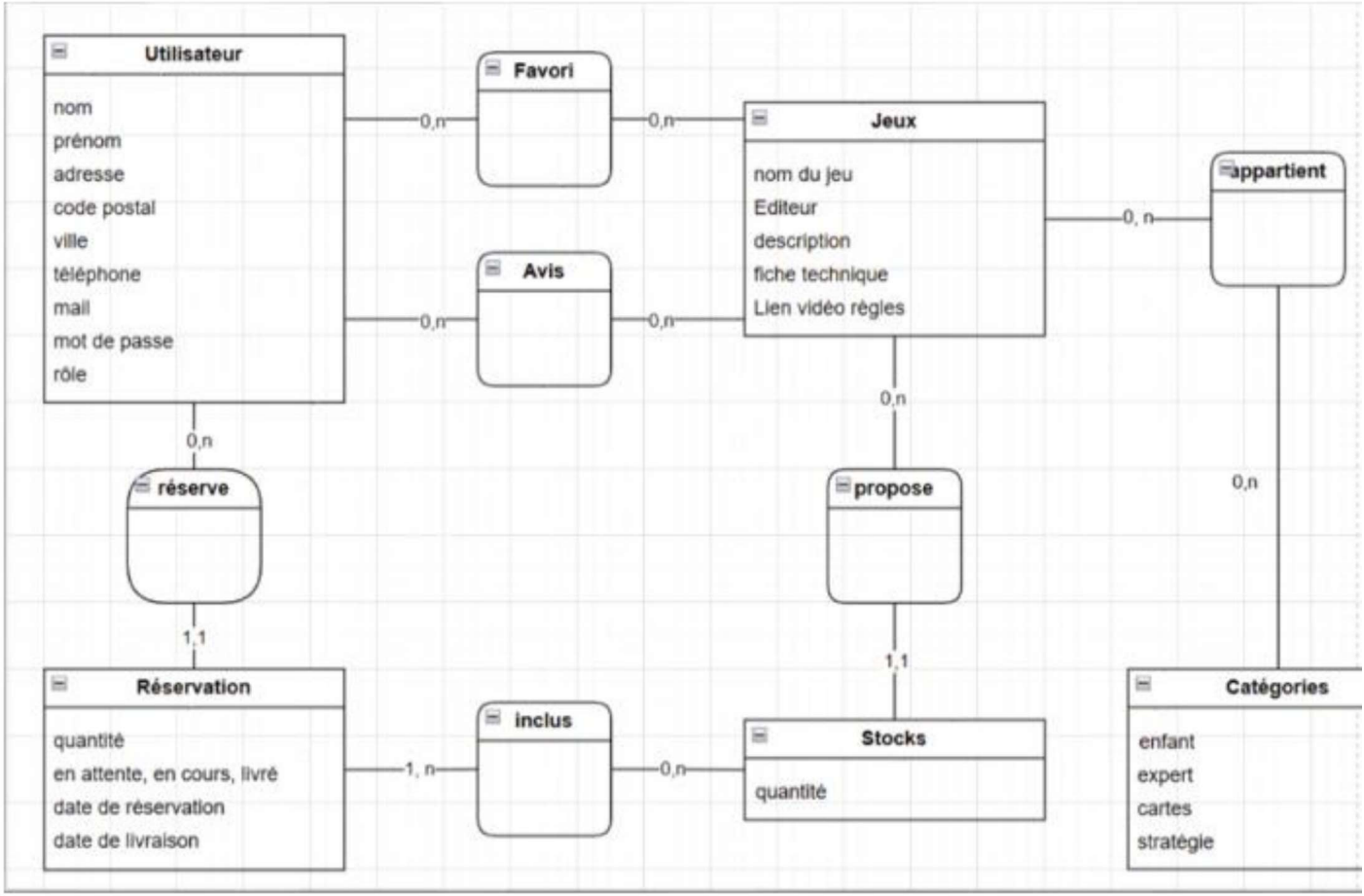
Certification DWWM

FOUQUET Damien

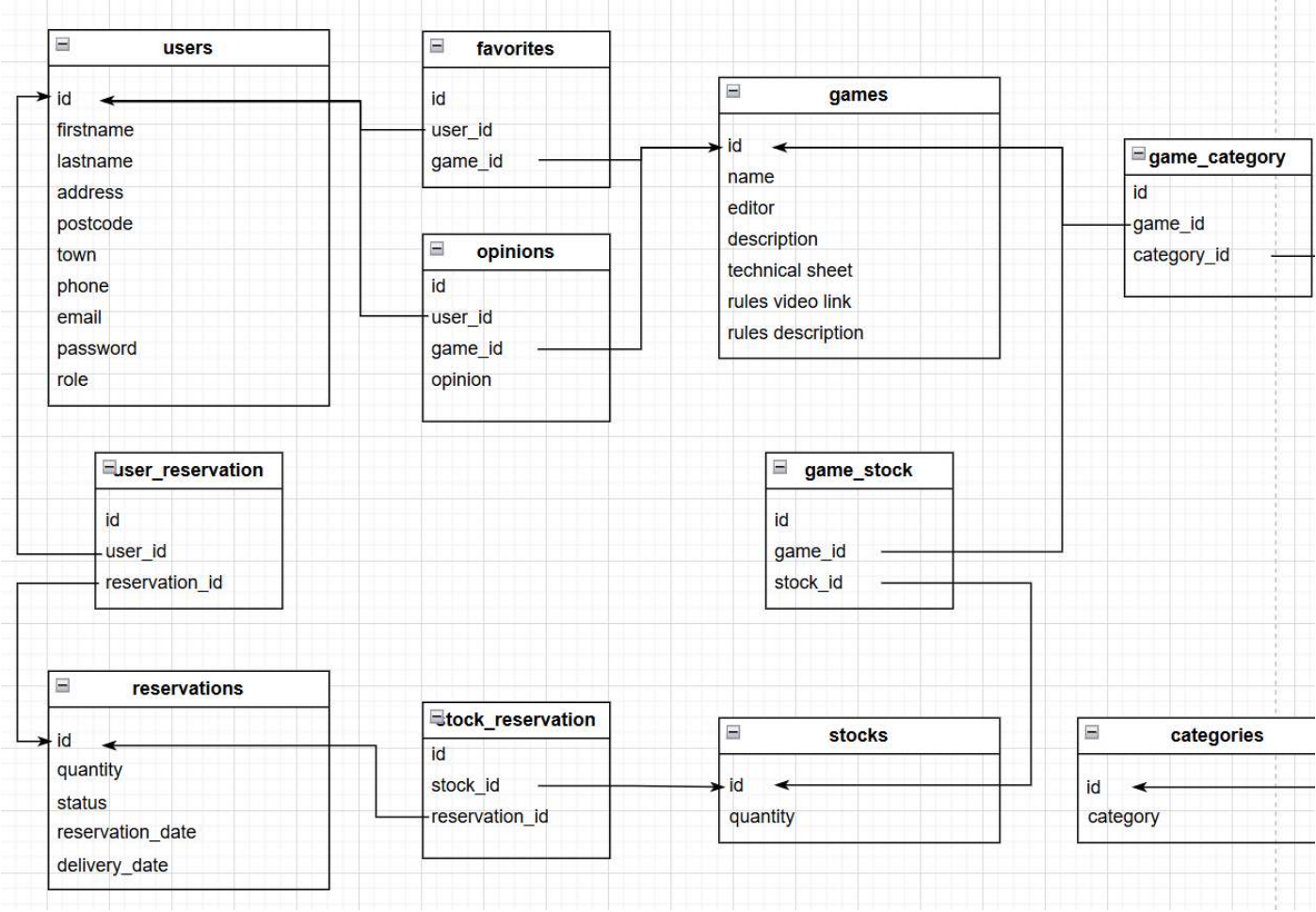
Présentation de mon site de vente de jeux de société en ligne

Backend

MCD


Utilisateur
nom
prénom
adresse
code postal
ville
téléphone
mail
mot de passe
rôle
Favori
Avis
Jeux
nom du jeu
Editeur
description
fiche technique
Lien vidéo règles
appartient
réserve
propose
Réservation
quantité
en attente, en cours, livré
date de réservation
date de livraison
inclus
Stocks
quantité
Catégories
enfant
expert
cartes
stratégie
0,n
0, n
1,1
1, n

MPD


users
id
firstname
lastname
address
postcode
town
phone
email
password
role
favorites
id
user_id
game_id
opinions
id
user_id
game_id
opinion
games
id
name
editor
description
technical sheet
rules video link
rules description
game_category
id
game_id
category_id
user_reservation
id
user_id
reservation_id
game_stock
id
game_id
stock_id
reservations
id
quantity
status
reservation_date
delivery_date
stock_reservation
id
stock_id
reservation_id
stocks
id
quantity
categories
id
category

```
db.exec(`
  CREATE TABLE IF NOT EXISTS users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    firstname TEXT NOT NULL,
    lastname TEXT NOT NULL,
    address TEXT NOT NULL,
    postcode TEXT NOT NULL,
    town TEXT NOT NULL,
    phone TEXT NOT NULL,
    email TEXT NOT NULL,
    password TEXT NOT NULL,
    role TEXT NOT NULL
  );
`);

db.exec(`
  CREATE TABLE IF NOT EXISTS games (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    editor TEXT NOT NULL,
    description TEXT,
    technical_sheet TEXT,
    rules_video_link TEXT,
    rules_description TEXT NOT NULL
  );
`);
```

Création de la database

Redirection du back

```
import { verify } from 'hono/jwt'
import { authGuard } from '../middlewares/au
import authRouter from './auth.router.js'
import opinionsRouter from './opinions.route
import gameCategoryRouter from './game.categ
import favorisRouter from './favoris.router.
import categoriesRouter from './categories.r
import gamesRouter from './games.router.js'
import reservationsRouter from './reservatio
import usersRouter from './users.router.js'
import authService from '../services/auth.se
import env from '../config/env.js'
const app = new Hono()


app.get('/', (c) => c.text('Hello from Hono!
app.route('/api/auth', authRouter)
app.route('/api/opinions', opinionsRouter)
app.route('/api/game_category', gameCategory
app.route('/api/favoris', favorisRouter)
app.route('/api/categories', categoriesRoute
app.route('/api/games', gamesRouter)
app.route('/api/reservations', reservationsR
app.route('/api/users', usersRouter)
app.get(
  '/authenticated',
  authGuard(),
  (c) => {
    const user = c.get('user')
    return c.text('Authenticated route, hi '
  }
)
export default app
```

Exemple de router

```
gamesRouter.post(
  "/register", zValidator('json',
    z.object({
      name: z.string().min(2),
      editor: z.string().min(2),
      description: z.string(),
      tecnhicalSheet: z.string(),
      rulesVideoLink: z.string(),
      rulesDescription: z.string().min(2),
      quantity: z.number().positive(),
    })
  ),
  register
);

gamesRouter.get(
  "/get-data/:gameId",
  getData
);
```

Exemple de controller

```
async function register(c) {
  try {
    const data = c.req.valid('json')
    const gameId = await gamesService.register(data)

    return c.json({
      message: 'Registration successful. Please check your email for verification.',
      gameId
    }, 201)
  } catch (error) {
    console.error(error)

    return c.json({
      error: 'Registration failed'
    }, 400)
  }
}
```

Exemple de service

```
async function createGame(data) {
  const query = `
    INSERT INTO games (name, editor, description, technical_sheet, rules_video_link, rules_description)
    VALUES (?, ?, ?, ?, ?, ?)
  `;
  const values = [data.name, data.editor, data.description, data.technical_sheet, data.rules_video_link, data.rules_de
  const result = await db.prepare(query).run(values);
  return await result.lastInsertRowid;
}

async function register(data) {
  const gameId = await createGame(data);

  return gameId;
}

async function getData(gameId) {
  const query = `SELECT * FROM games WHERE id = ?`;
  const result = await db.prepare(query).get(gameId);
  console.log(result);
  return result;
}
```

Frontend

Lien entre le front et le back

```
import instance from "./config";
import axios from "axios"

async function registerGames(data) {
    return await instance.post("/games/register", data)
}

async function updateGames(data) {
    return await instance.put("/games/update", data)
}

async function getGames(gameId) {
    try {
        const url = "http://localhost:3000/api/games/get-data/" + gameId;
        console.log(url)
        const response = await axios.get(url)
        console.log("response", response.data.data)
        localStorage.setItem("name", response.data.data.name)
        localStorage.setItem("editor", response.data.data.editor)
        localStorage.setItem("description", response.data.data.description)
        localStorage.setItem("technical_sheet", response.data.data.technical_sh
        localStorage.setItem("rules_video_link", response.data.data.rules_video
        localStorage.setItem("rules_description", response.data.data.rules_desc
        return response.data.data
    } catch (error) {
        return error
    }
}

export { registerGames, updateGames, getGames }
```

Exemple de formulaire de creation

```
<form id="form-etablissement" className="flex flex-col gap-2" onSubmit={handleSubmit(onSubmi
    <label htmlFor="raison-sociale" className="font-bold mb-1 mt-3">
        Nom du jeu :
    </label>
    <input
        type="text"
        id="name"
        name="raison_sociale"
        placeholder=" Nom du jeu"
        required
        className="p-2 border border-gray-300 rounded"
        {...register("name")}
    />

    <label htmlFor="editeur" className="font-bold mb-1 mt-3">
        Editeur :
    </label>
    <textarea
        id="editor"
        name="editeur"
        placeholder="Nom de l'éditeur"
        required
        className="p-2 border border-gray-300 rounded"
        {...register("editor")}>
    </textarea>
```

```
})

export default function AdminGameCreation() {
    const { register, handleSubmit } = useForm({
        resolver: zodResolver(registerSchema),
        defaultValues: {
            name: "Puzzle",
            editor: "Ravensburger",
            description: "Un puzzle à reconstituer",
            technical_sheet: "Un texte pour tester",
            rules_video_link: "",
            rules_description: "Prenez les pièces et assemblez le
        }
    })

    const registerMutation = useMutation({
        mutationFn: async (newTodo) => {
            console.log("newTodo", newTodo)
            return await registerGames(newTodo)
        },
        onSuccess: (data, variables, context) => {
            console.log("data", data)
            window.location = "/"
```

Exemple de formulaire de mise à jou

```
<form id="form-etablissement" className="flex flex-col gap-2" onSubmit={handleSubmit(onSubmit)
    <label htmlFor="raison-sociale" className="font-bold mb-1 mt-3">
        Nom du jeu :
    </label>
    <input
        type="text"
        id="name"
        name="raison_sociale"
        placeholder=" Nom du jeu"
        required
        className="p-2 border border-gray-300 rounded"
        {...register("name")}
    />

    <label htmlFor="editeur" className="font-bold mb-1 mt-3">
        Editeur :
    </label>
    <textarea
        id="editor"
        name="editeur"
        placeholder="Nom de l'éditeur"
        required
        className="p-2 border border-gray-300 rounded"
        {...register("editor")}>
    </textarea>
```

```
})

export default function AdminCompte() {
    const [complement, setComplement] = useState('')
    const { register, handleSubmit } = useForm ({
        resolver: zodResolver(updateSchema),
        defaultValues: {
            organismId: localStorage.getItem("gameId"),
            name: "Puzzle",
            editor: "Ravensburger",
            description: "Un puzzle à reconstituer",
            technical_sheet: "Un texte pour tester",
            rules_video_link: "",
            rules_description: "Prenez les pièces et assemblez le
        }
    })

    const updateMutation = useMutation({
        mutationFn: async (newTodo) => {
            console.log("newTodo", newTodo)
            return await updateGames(newTodo)
        },
        onSuccess: (data, variables, context) => {
```

Exemple de récupération de données

```
export default function AdminGameGet() {
    const { data, isLoading, isError, error } = useQuery({
        queryKey: ["infoGame"],
        queryFn: getGames(localStorage.getItem("gameId")),
        refetchOnWindowFocus: false,
    });
    console.log(localStorage.getItem("name"))
    console.log(localStorage.getItem("editor"))
    console.log(localStorage.getItem("description"))
```