Création d'une Application Web - Full Stack

- Frontend : React, Vite, React Router, TypeScript
- Backend : Node.js, Express.js
- Base de données : PostgreSQL
- Déploiement : Docker, Docker Compose
- Hébergement : Render.com
- Outils & Qualité du Code : Biome, Npm
- Gestion de code & CI/CD : Git, GitHub

1. Initialiser le projet et les dépendances

- 1.01 - Ouvrir Git BASH et se rendre dans le dossier des projets :
```
cd ~/desktop/LOCAL
```
- 1.02 - Créer un dossier racine pour ton projet :
```
mkdir fullstack-starter-kit
```
- 1.03 - Se rendre dans ce dossier :
```
cd fullstack-starter-kit
```

Création du dossier client (frontend) avec Vite

- 1.04 - Créer le dossier client et initialiser Vite :
```
npm create vite@latest client
```
    - Use rolldown-vite (Experimental)?
    ```
    no
    ```
    - Install with npm and start now?
    ```
    no
    ```
- 1.05 - Se rendre dans le dossier client :
```
cd client
```
- 1.06 - Installer les dépendances du client :
```
npm install && npm install -D @types/react
```
- 1.07 - Installer le plugin React pour Vite :
```
npm install -D @vitejs/plugin-react
```
- 1.08 - Installer React Router :
```
npm install react-router && npm install --save-dev @types/react-router
```
- 1.09 - Installer react-toastify :
```
npm install react-toastify && npm install -D @types/react-toastify
```
- 1.10 - Installer les types Node pour TypeScript :
```
npm install --save-dev @types/node
```
- 1.11 - Installer les types React et ReactDOM :
```
npm install --save-dev @types/react @types/react-dom
```

Création du dossier server (backend)

- 1.12 - Revenir à la racine du projet :
```
cd ..
```
- 1.13 - Créer le dossier server :
```
mkdir server
```
- 1.14 - Se rendre dans le dossier server :
```
cd server
```
- 1.15 - Initialiser un projet Node :
```
npm init -y
```
- 1.16 - Installer Express :
```
npm install express && npm install -D @types/express
```
- 1.17 - Installer TypeScript et outils dev :

```
npm install --save-dev typescript tsx @types/node @types/express
```

- 1.18 - Installer dotenv :

```
npm install dotenv
```

- 1.19 - Installer PostgreSQL :

```
npm install pg && npm install -D @types/pg && npm install -D @types/node
```

- 1.20 - Installer cors :

```
npm install cors && npm install -D @types/cors
```

- 1.21 - Installer bcrypt :

```
npm install bcrypt && npm install -D @types/bcrypt
```

- 1.22 - Installer cookie parser :

```
npm install cookie-parser @types/cookie-parser
```

- 1.23 - Installer jsonwebtoken :

```
npm install jsonwebtoken && npm install --save-dev @types/jsonwebtoken
```

Installation les outils pour le projet full stack

- 1.24 - Revenir à la racine du projet :

```
cd ..
```

- 1.25 - Installer concurrently :

```
npm install concurrently@latest --save-dev
```

- 1.26 - Installer Biome en tant que dépendance de développement :

```
npm install -D @biomejs/biome
```

- 1.27 - Initialiser la configuration Biome :

```
npx biome init
```

- 1.28 - Ouvrir Vs Code :

```
code .
```

2. Préparer la structure RACINE de l'application

- 2.01 - Racine > Créer un dossier app_ressources
- 2.02 - Racine > app_ressources > Créer un fichier readme.md
- 2.03 - Racine > app_ressources > README.md > Coller le contenu suivant :

```
### app_ressources

Ce dossier contient les fichiers `annexes` au projet qui ne font pas directement partie du code source.
`Ce contenu n'est pas versionné` et est destiné uniquement à un usage local.

- `Documents PDF / Word` - Textes à intégrer dans le site, conditions générales, contrats, guides, notices
- `Screenshots / Maquettes` - Aperçus de design ou d’interface, références visuelles pour le développement
- `Documents de travail` - Cahier des charges, spécifications techniques ou fonctionnelles, idées, notes de projet, to-do lists
- `Tableaux Excel / CSV` - Liste de produits, services, utilisateurs, etc.
- `Brouillons` - Fichiers non finalisés ou archivés pour consultation ultérieure

---

Légende des couleurs utilisées dans le code :
🔘 INFO / LOAD
Processus en cours, lecture ou initialisation.
Scan de fichiers, tentative de connexion, démarrage.

🔘 SUCCESS
Action terminée avec succès ou état valide.
```

```
Compilation réussie, base de données prête, serveur en ligne.

🔴 ERROR
Échec critique, exception ou fichier manquant.
Erreurs de syntaxe, rejet de connexion, crash système.
```

- 2.04 - Racine > Créer un fichier .gitignore
- 2.05 - Racine > .gitignore > Coller le contenu suivant :

```
# Global
node_modules
.env
dist
build
.DS_Store
.vscode
.idea
*.log
npm-debug.log*

# Vite (client)
client/dist
client/.vite

# TypeScript (server)
server/dist
server/*.tsbuildinfo

# Ressources locales (non versionnées)
app_ressources/

# Logs
logs
*.log
```

- 2.06 - Racine > Créer un fichier package.json
- 2.07 - Racine > package.json > Coller le contenu suivant :

```
{
    "name": "projet",
    "version": "1.0.0",
    "private": true,
    "type": "module",
    "scripts": {
        "dev": "docker ps -q --filter \"expose=5432\" --filter \"publish=5433\" | xargs -r
docker stop && docker-compose down --remove-orphans && docker-compose up -d && concurrently -
c \"magenta,cyan\" -t \"HH:mm:ss\" -p \"[{name} {time}]\" \"npm run dev:server\" \"npm run
dev:client\"",
        "start:apps": "concurrently -c \"magenta,cyan\" -t \"HH:mm:ss\" -p \"[{name}
{time}]\" -n \"API,APP\" \"npm run dev:server\" \"npm run dev:client\"",
        "dev:server": "npm run dev --prefix server",
        "dev:client": "npm run dev --prefix client -- --host",
        "db:init": "npx tsx server/database/init-db.ts",
        "build:client": "npm run build --prefix client",
        "start": "npm run start --prefix server",
        "setup": "node setup/update-package-name.js",
        "postinstall": "npm run setup",
```

```
        "setup:project": "node setup/update-package-name.js",
        "render-build": "npm install --prefix client && npm run build --prefix client && npm
run build --prefix server"
    },
    "devDependencies": {
        "@biomejs/biome": "^2.3.9",
        "@vitejs/plugin-react": "^5.1.2",
        "concurrently": "^9.2.1",
        "vite": "^7.2.4"
    }
}
```

- 2.08 - Racine > Créer un fichier tsconfig.json
- 2.09 - Racine > tsconfig.json > Coller le contenu suivant :

```
{
    "files": [],
    "references": [{ "path": "./client" }, { "path": "./server" }],
    "compilerOptions": {
        "target": "ES2022",
        "module": "NodeNext",
        "moduleResolution": "NodeNext",
        "lib": ["ES2022"],
        "strict": true,
        "esModuleInterop": true,
        "skipLibCheck": true,
        "forceConsistentCasingInFileNames": true,
        "baseUrl": "."
    }
}
```

- 2.10 - Racine > Créer un fichier docker-compose.yml
- 2.11 - Racine > docker-compose.yml > Coller le contenu suivant :

```YAML
services:
  postgres:
    image: postgres:16
    container_name: postgres-${POSTGRES_DB}
    restart: always
    environment:
      POSTGRES_USER: ${POSTGRES_USER}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
      POSTGRES_DB: ${POSTGRES_DB}
    ports:
      - "5433:5432"
    volumes:
      - ./setup/init.sql:/docker-entrypoint-initdb.d/init.sql
      - pgdata:/var/lib/postgresql/data

volumes:
  pgdata:
```

- 2.12 - Racine > Créer un dossier setup
- 2.13 - Racine > setup > Créer un fichier update-package-name.js
- 2.14 - Racine > setup > update-package-name.js > Coller le contenu suivant :

```
import fs from "node:fs";
import path from "node:path";

const rootFolderName = path.basename(path.resolve());

console.log(
    `\n========================\n🟡 AUTO-CONFIG PROJECT\n========================\n`,
);
console.log(`🟡 Dossier racine détecté : ${rootFolderName}`);

const constantsPath = path.resolve("setup/constants.ts");

if (fs.existsSync(constantsPath)) {
    console.log(`🟡 Analyse de ${constantsPath}...`);
    let constantsContent = fs.readFileSync(constantsPath, "utf8");

    if (constantsContent.includes("PENDING_GENERATION")) {
        constantsContent = constantsContent.replace(
            /PENDING_GENERATION/g,
            rootFolderName,
        );
        fs.writeFileSync(constantsPath, constantsContent);
        console.log(`🟡 constants.ts mis à jour avec le nom : ${rootFolderName}`);
    } else {
        console.log(
            `🟡 constants.ts est déjà à jour (aucune mention 'PENDING_GENERATION').`,
        );
    }
} else {
    console.log("🟡 Erreur : Fichier constants.ts introuvable dans /setup");
}

function updateEnv(envPath) {
    const fileName = path.basename(envPath);
    const dirName = path.dirname(envPath).split(path.sep).pop() || "root";

    if (fs.existsSync(envPath)) {
        console.log(`🟡 Mise à jour de : ${dirName}/${fileName}...`);
        let envContent = fs.readFileSync(envPath, "utf8");

        if (/PGDATABASE=/.test(envContent)) {
            envContent = envContent.replace(
                /PGDATABASE=.*/,
                `PGDATABASE=${rootFolderName}`,
            );
        } else {
            envContent += `\nPGDATABASE=${rootFolderName}`;
        }

        fs.writeFileSync(envPath, envContent);
        console.log(
            `🟡 ${fileName} (${dirName}) mis à jour avec PGDATABASE=${rootFolderName}`,
        );
    } else {
        console.log(`🟡 Fichier non trouvé : ${dirName}/${fileName}`);
    }
}
```

```
console.log(`\n🟢 Mise à jour des fichiers d'environnement...`);
updateEnv(path.resolve("server/.env"));
updateEnv(path.resolve("client/.env"));
updateEnv(path.resolve(".env"));

console.log(
    `\n========================\n🟢 CONFIGURATION TERMINÉE\n========================\n`,
);
```

- 2.15 - Racine > setup > Créer un fichier constants.ts
- 2.16 - Racine > setup > constants.ts > Coller le contenu suivant :

```
// Importez simplement `constants` où vous en avez besoin pour garantir une cohérence et
faciliter la maintenance.
// import { constants } from "../../../../setup/constants"

export const constants = {
    ROOT_FOLDER_NAME: "PENDING_GENERATION",
    APP_NAME: "PENDING_GENERATION",

    DEFAULT_USER_NAME: "utilisateur",
    DEFAULT_AVATAR: "/images/avatar_profil.png",

    ROUTE_HOME: "/",
    ROUTE_LOGIN: "/login",
    ROUTE_AUTH: "/auth",
    ROUTE_DASHBOARD: "/dashboard",
    ROUTE_CONTACT: "/contact",

    TEXT_WELCOME: "Bienvenue",
    TEXT_LOGOUT: "Se déconnecter",
    TEXT_LOGIN: "Se connecter",
    TEXT_SEARCH_PLACEHOLDER: "Rechercher...",

    DEFAULT_LANGUAGE: "fr",
    DEFAULT_THEME: "light",
    DEFAULT_TIMEOUT: 10000,
    DEFAULT_PAGE_SIZE: 10,

    API_BASE_URL: "http://localhost:3310/api",
    API_AUTH_ENDPOINT: "/auth",
};
```

- 2.17 - Racine > setup > Créer un dossier exports_writings
- 2.18 - Racine > setup > exports_writings > Créer un fichier export_writings_all.js
- 2.19 - Racine > setup > exports_writings > export_writings_all.js > Coller le contenu suivant :

```
// ## Script pour exporter tout le code
// Racine > setup > exports_writings > Créer export_writings_all.js

import fs from "node:fs";
import path from "node:path";
import { fileURLToPath } from "node:url";

const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
```

```
const outputFileName = "export_writings_all.md";
const outputPath = path.join(__dirname, outputFileName);

const excludedPaths = [
    "node_modules",
    "build",
    "dist",
    "client/dist",
    "server/dist",
    "logs",
    ".vscode",
    ".idea",
    ".vite",
    "client/.vite",
    ".git",
    "setup",
    "exports_writings",
];

const excludedFiles = [".DS_Store", "npm-debug.log", outputFileName];
const excludedExtensions = [
    ".tsbuildinfo",
    ".log",
    ".png",
    ".jpg",
    ".jpeg",
    ".gif",
    ".ico",
    ".pdf",
    ".zip",
];

function detectLanguage(file) {
    const ext = path.extname(file).slice(1).toLowerCase();
    const map = {
        js: "javascript",
        ts: "typescript",
        sh: "bash",
        css: "css",
        html: "html",
        py: "python",
        json: "json",
        yml: "yaml",
        tsx: "tsx",
        md: "markdown",
    };
    return map[ext] || "";
}

function isExcluded(filePath) {
    const normalized = filePath.replace(/\\/g, "/");
    return (
        excludedPaths.some(
            (folder) =>
                normalized.includes(`/${folder}/`) ||
                normalized.split("/").includes(folder),
        ) ||
        excludedFiles.some((name) => path.basename(normalized) === name) ||
```

```
        excludedExtensions.some((ext) => normalized.endsWith(ext))
    );
}

function isBinary(buffer) {
    let nonAscii = 0;
    const length = Math.min(buffer.length, 1000);
    for (let i = 0; i < length; i++) {
        if (buffer[i] > 127) nonAscii++;
    }
    return buffer.length > 0 && nonAscii / length > 0.3;
}

function getAllFiles(dir) {
    try {
        const entries = fs.readdirSync(dir, { withFileTypes: true });
        return entries.flatMap((entry) => {
            const fullPath = path.join(dir, entry.name);
            if (entry.isDirectory()) {
                if (isExcluded(fullPath)) return [];
                return getAllFiles(fullPath);
            }
            return fullPath;
        });
    } catch (_err) {
        return [];
    }
}

try {
    console.log(
        `\n========================\n🟢 EXPORT GLOBAL\n========================\n`,
    );

    let output = "# CONTENU DU PROJET\n";
    const projectRoot = process.cwd();

    console.log(`🟢 Analyse des fichiers dans : ${projectRoot}...`);

    const allFiles = getAllFiles(projectRoot)
        .filter(
            (f) =>
                fs.existsSync(f) &&
                path.basename(f) !== outputFileName &&
                path.basename(f) !== path.basename(__filename) &&
                !isExcluded(f),
        )
        .sort();

    console.log(
        `🟢 ${allFiles.length} fichiers trouvés. Génération du Markdown...`,
    );

    for (const file of allFiles) {
        const relPath = path.relative(projectRoot, file);
        const lang = detectLanguage(file);
        const raw = fs.readFileSync(file);
        if (isBinary(raw)) {
            output += `\n### \`${relPath}\`\n⚠ **Fichier ignoré (binaire)**\n`;
```

```
            continue;
        }
        output += `\n### \`${relPath}\`\n\`\`\`${lang}\n${raw.toString("utf8")}\n\`\`\`\n`;
    }

    fs.writeFileSync(outputPath, output, "utf8");

    console.log(`🟢 EXPORT RÉUSSI : ${outputFileName} créé.`);
    console.log(`🟢 Emplacement : setup/exports_writings/`);
    console.log(`\n-------------------------\n`);
} catch (error) {
    console.error(`\n🟢 ERREUR lors de l'export :`, error.message);
    console.log(`\n-------------------------\n`);
}
// ## Commande Bash pour executer le script :
// node setup/exports_writings/export_writings_all.js
```

- 2.20 - Racine > setup > exports_writings > Créer un fichier export_writings_gitDiff.js
- 2.21 - Racine > setup > exports_writings > export_writings_gitDiff.js > Coller le contenu suivant :

```
// ## Script pour exporter les différences de Git
// Racine > setup > exports_writings > Créer export_writings_gitDiff.js

import { exec } from "node:child_process";
import fs from "node:fs";
import path from "node:path";
import { fileURLToPath } from "node:url";

const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);

const outputFileName = "export_writings_gitDiff.md";
const outputPath = path.join(__dirname, outputFileName);

console.log(
    `\n=========================\n🟢 GIT DIFF EXPORT\n=========================\n`,
);
console.log(`🟢 Analyse des changements non commités...`);

exec("git diff", (err, stdout) => {
    if (err) {
        console.error(`🟢 ERREUR lors du git diff :`, err.message);
        return;
    }

    if (!stdout) {
        console.log(`🟢 Aucun changement détecté dans le code.`);
        stdout = "Aucune différence détectée (git diff vide).";
    }

    const markdown = `# Git Diff - ${new
Date().toLocaleString()}\n\n\`\`\`diff\n${stdout}\n\`\`\`\n`;

    fs.writeFile(outputPath, markdown, "utf8", (writeErr) => {
        if (writeErr) {
            console.error(`🟢 ERREUR d'écriture du fichier :`, writeErr.message);
        } else {
            console.log(`🟢 EXPORT RÉUSSI : ${outputFileName} généré.`);
```

```
            console.log(`🟢 Emplacement : setup/exports_writings/`);
            console.log(`\n-------------------------\n`);
        }
    });
});
// ## Commande Bash pour executer le script :
// node setup/exports_writings/export_writings_gitDiff.js
```

- 2.22 - Racine > setup > exports_writings > Créer un fichier export_writings_noCommit.js
- 2.23 - Racine > setup > exports_writings > export_writings_noCommit.js > Coller le contenu suivant :

```
// ## Script pour exporter les fichiers non suivi par git
// Racine > setup > exports_writings > Créer export_writings_noCommit.js

import { execSync } from "node:child_process";
import fs from "node:fs";
import path from "node:path";
import { fileURLToPath } from "node:url";

const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
const selfName = path.basename(__filename);

const outputFileName = "export_writings_noCommit.md";
const outputPath = path.join(__dirname, outputFileName);

function detectLanguage(file) {
    const ext = path.extname(file).slice(1);
    const map = {
        js: "javascript",
        ts: "typescript",
        sh: "bash",
        css: "css",
        html: "html",
        json: "json",
        md: "markdown",
        tsx: "tsx",
    };
    return map[ext] || "";
}

try {
    console.log(
        `\n=========================\n🟢 EXPORT NO-COMMIT\n=========================\n`,
    );

    console.log(`🟢 Analyse des fichiers modifiés et non suivis...`);

    const modifiedFiles = execSync("git diff --name-only", { encoding: "utf8" })
        .split("\n")
        .filter(
            (f) => f && fs.existsSync(f) && f !== outputFileName && f !== selfName,
        );

    const untrackedFiles = execSync("git ls-files --others --exclude-standard", {
        encoding: "utf8",
    })
        .split("\n")
```

```
        .filter(
            (f) => f && fs.existsSync(f) && f !== outputFileName && f !== selfName,
        );

    let output = "## FICHIERS À MODIFIER\n";
    if (modifiedFiles.length > 0) {
        console.log(`🟢 ${modifiedFiles.length} fichier(s) modifié(s) trouvé(s).`);
        for (const file of modifiedFiles) {
            output += `\n###
\`${file}\`\n\`\`\`${detectLanguage(file)}\n${fs.readFileSync(file, "utf8")}\n\`\`\`\n`;
        }
    } else {
        output += "\n_Aucun fichier modifié._\n";
    }

    output += "\n## NOUVEAUX FICHIERS\n";
    if (untrackedFiles.length > 0) {
        console.log(
            `🟢 ${untrackedFiles.length} nouveau(x) fichier(s) détecté(s).`,
        );
        for (const file of untrackedFiles) {
            output += `\n###
\`${file}\`\n\`\`\`${detectLanguage(file)}\n${fs.readFileSync(file, "utf8")}\n\`\`\`\n`;
        }
    } else {
        output += "\n_Aucun fichier non suivi._\n";
    }

    fs.writeFileSync(outputPath, output, "utf8");

    console.log(`🟢 EXPORT RÉUSSI : ${outputFileName} généré.`);
    console.log(`🟢 Emplacement : setup/exports_writings/`);
    console.log(`\n-------------------------\n`);
} catch (error) {
    console.error(`\n🟢 ERREUR lors de l'export noCommit :`, error.message);
    console.log(`\n-------------------------\n`);
}

// ## Commande Bash pour executer le script :
// node setup/exports_writings/export_writings_noCommit.js
```

- 2.24- Racine > Créer un fichier .env.sample
- 2.25 - Racine > .env.sample > Coller le contenu suivant :

```md
VITE_API_URL=/api
CLIENT_URL=http://localhost:5173
MODE=development

# Identifiants pour Docker et le Serveur
POSTGRES_USER=Your_user
POSTGRES_PASSWORD=Your_password
POSTGRES_DB=Your_database_name

# Variables pour la connexion Node.js
DB_USER=Your_user
DB_PASSWORD=Your_password
DB_NAME=Your_database_name
```

```
DB_HOST=localhost
DB_PORT=5433
```

- 2.26 - Racine > Créer une copie du fichier .env.sample, puis renommer en .env et renseigne les vraies infos.

3. Préparer la structure SERVER de l'application

- 3.01 - server > package.json > Ajouter et supprimer les scripts par le contenu suivant :

```
{
  "private": true,
  "type": "module",
  "scripts": {
    "dev": "tsx src/index.ts",
    "build": "tsc",
    "start": "ts-node --transpile-only src/index.ts",
  }
}
et supprumer celui ci
"type": "commonjs",
```

- 3.02 - server > Créer un fichier tsconfig.json
- 3.03 - server > tsconfig.json > Coller le contenu suivant :

```
{
    "compilerOptions": {
        "target": "ES2022",
        "module": "NodeNext",
        "moduleResolution": "NodeNext",
        "lib": ["ES2022"],
        "strict": true,
        "esModuleInterop": true,
        "skipLibCheck": true,
        "forceConsistentCasingInFileNames": true,
        "composite": true
    },
    "include": ["src", "services"]
}
```

- 3.04 - server > Créer un dossier database
- 3.05 - server > database > Créer un fichier init-db.ts

```
import pg from "pg";
import "dotenv/config";
import { initDB } from "../services/db/initDB.js";

const { Client } = pg;

async function run() {
    const config = {
        user: process.env.DB_USER,
        password: process.env.DB_PASSWORD,
        host: "localhost",
        port: 5433,
        database: "postgres",
```

```
    };

    const targetDb = process.env.DB_NAME;
    const client = new Client(config);

    try {
        await client.connect();
        const res = await client.query(
            "SELECT 1 FROM pg_database WHERE datname = $1",
            [targetDb],
        );

        if (res.rowCount === 0) {
            console.log(`🟡 Base "${targetDb}" absente. Création...`);
            await client.query(`CREATE DATABASE "${targetDb}"`);
            console.log(`🟢 Base "${targetDb}" créée.`);
        }
        await client.end();
        await initDB();
    } catch (err: unknown) {
        if (err instanceof Error) {
            console.error("🔴 Erreur Init Script :", err.message);
        } else {
            console.error("🔴 Erreur Init Script : Une erreur inconnue est survenue");
        }
        process.exit(1);
    }
}

run();
```

- 3.06 - server > database > Créer un dossier script
- 3.07 - server > database > script > Créer un fichier schema.sql
- 3.08 - server > database > script > schema.sql > Coller le contenu suivant :

```
CREATE TABLE IF NOT EXISTS users (
  id SERIAL PRIMARY KEY,
  username VARCHAR(100) NOT NULL UNIQUE,
  email VARCHAR(255) NOT NULL UNIQUE,
  password VARCHAR(255) NOT NULL,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE IF NOT EXISTS user_profiles (
  user_id INT PRIMARY KEY,
  address VARCHAR(255),
  city VARCHAR(100),
  postal_code VARCHAR(20),
  profile_photo VARCHAR(510),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);

DO $$
BEGIN
  IF NOT EXISTS (SELECT 1 FROM pg_type WHERE typname = 'role_type') THEN
    CREATE TYPE role_type AS ENUM ('free', 'premium', 'admin');
  END IF;
```

```
END$$;

CREATE TABLE IF NOT EXISTS roles (
  id SERIAL PRIMARY KEY,
  role role_type NOT NULL UNIQUE
);

CREATE TABLE IF NOT EXISTS user_roles (
  user_id INT NOT NULL,
  role_id INT NOT NULL,
  PRIMARY KEY(user_id, role_id),
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
  FOREIGN KEY (role_id) REFERENCES roles(id) ON DELETE CASCADE
);
```

- 3.09 - server > Créer un dossier services
- 3.10 - server > services > Créer un dossier db
- 3.11 - server > services > db > Crée un fichier index.ts
- 3.12 - server > services > db > index.ts > Coller le contenu suivant :

```
import path from "node:path";
import { fileURLToPath } from "node:url";
import dotenv from "dotenv";
import pkg from "pg";

const { Pool } = pkg;

const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);

dotenv.config({ path: path.resolve(__dirname, "../../../.env") });

export const db = new Pool({
    user: process.env.DB_USER,
    password: process.env.DB_PASSWORD,
    host: process.env.DB_HOST,
    port: Number(process.env.DB_PORT),
    database: process.env.DB_NAME,
});

console.log(`\n=========================`);
console.log(`🟢 DB SERVICE INIT`);
console.log(`=========================`);
console.log(`🟢 DATABASE : ${process.env.DB_NAME}`);
console.log(`🟢 HOST     : ${process.env.DB_HOST}:${process.env.DB_PORT}`);
console.log(`🟢 POOL     : Connecté avec ${process.env.DB_USER}`);
console.log(`-------------------------\n`);
```

- 3.13 - server > services > db > Créer un fichier initDB.ts
- 3.14 - server > services > db > initDB.ts > Coller le contenu suivant :

```
import { readFileSync } from "node:fs";
import path from "node:path";
import { fileURLToPath } from "node:url";
import { db } from "./index.js";

const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
```

```
export async function initDB(): Promise<void> {
    try {
        const schemaPath = path.resolve(
            __dirname,
            "../../database/script/schema.sql",
        );

        const schema = readFileSync(schemaPath, "utf-8");
        const client = await db.connect();

        try {
            await client.query(schema);
            console.log(
                `🟢 Base de données "${process.env.DB_NAME}" : Structure créée ou mise à
jour.`,
            );
        } finally {
            client.release();
        }
    } catch (err: unknown) {
        const error = err as Error;
        console.error("🟢 Erreur SQL détaillée :", error.message);
        throw error;
    }
}
```

- 3.15 - server > services > Créer un fichier index.ts
- 3.16 - server > services > index.ts > Coller le contenu suivant :

```
export { db } from "./db/index.js";
export { initDB } from "./db/initDB.js";
```

- 3.17 - server > Créer le dossier src
- 3.18 - server > src > Créer le dossier routes
- 3.19 - server > src > routes > api.ts > Coller le contenu suivant :

```
import { Router } from "express";

const router = Router();

router.get("/", (_req, res) => {
    res.json({ message: "API prête" });
});

export default router;
```

- 3.20 - server > src > Créer un fichier index.ts
- 3.21 - server > src > index.ts > Coller le contenu suivant :

```
import cors from "cors";
import "dotenv/config";
import express from "express";
import { db } from "../services/db/index.js";
import { initDB } from "../services/db/initDB.js";
import apiRoutes from "./routes/api.js";
```

```
const app = express();
const PORT = process.env.SERVER_PORT || 3310;
const CLIENT_URL = process.env.CLIENT_URL || "http://localhost:5173";

app.use(express.json());
app.use(cors({ origin: CLIENT_URL, credentials: true }));
app.use("/api", apiRoutes);

db.query("SELECT 1")
    .then(async () => {
        try {
            await initDB();
            app.listen(PORT, () => {
                console.log("\n========================================");
                console.log("🟢 SECTION : SERVER BACKEND");
                console.log("========================================");
                console.log(`🟢 DATABASE : CONNECTED`);
                console.log(`🟢 SERVER   : http://localhost:${PORT}`);
                console.log(`🟢 CLIENT   : ${CLIENT_URL}\n`);
            });
        } catch (err: unknown) {
            const error = err as Error;
            console.error("🟢 Erreur initialisation tables :", error.message);
            process.exit(1);
        }
    })
    .catch((err: unknown) => {
        const error = err as Error;
        console.error("\n========================================");
        console.error("🟢 ERREUR CRITIQUE DATABASE");
        console.error(`Détail technique : ${error.message}`);
        console.error("========================================\n");
        process.exit(1);
    });
```

- 3.22 - server > Créer un fichier .gitignore
- 3.23 - server > .gitignore > Coller le contenu suivant :

```
# Dépendances
node_modules/

# Environnement
.env

# Build TypeScript
dist/
*.tsbuildinfo

# Logs
logs/
*.log
npm-debug.log*

# IDE / OS
.DS_Store
.vscode/
.idea/
```

- 3.24 - server > Créer un fichier .env.sample
- 3.25 - server > .env.sample > Coller le contenu suivant :

```
NODE_ENV=development
PORT=3310
APP_SECRET= Pour générer la key, coller dans le terminal bash cette commande : node -e
"console.log(require('crypto').randomBytes(48).toString('hex'))"

PGHOST=localhost
PGPORT=5433
PGUSER=Your_user
PGPASSWORD=Your_password
PGDATABASE=Your_db_name

CLIENT_URL=http://localhost:5173
```

- 3.26 - server > Faire une copie du fichier .env.sample, puis renommer en .env et renseigne les vraies infos.

4. Préparer la structure CLIENT de l'application

- 4.01 - client > public > Ajouter les différents éléments accessibles par le public
- 4.02 - client > public > Supprimer l'image vite.svg
- 4.03 - client > src > assets > Supprimer l'image react.svg
- 4.04 - client > src > Créer les dossiers et fichiers :

```
components
    header
        Header.css
        Header.tsx
pages
    homePage
        HomePage.css
        HomePage.tsx
    notFoundPage
        NotFoundPage.css
        NotFoundPage.tsx
```

- 4.05 - client > src > Les différents composants ont les contenus suivants :

Header.tsx

```jsx
import "./Header.css";
import { constants } from "../../../../setup/constants";

function Header() {
    return (
        <header className="header">
            <div className="left-side">
                <h1>{constants.APP_NAME}</h1>
            </div>
        </header>
    );
}

export default Header;
```

Header.css

```css
.header {
    position: fixed;
    top: 0;
    left: 0;
    right: 0;
    z-index: 1000;
    display: flex;
    align-items: center;
    padding: 14px 24px;
    height: 72px;
    background: linear-gradient(
        135deg,
        var(--teal-color) 0%,
        var(--teal-dark-color) 100%
    );
    color: var(--white-color);
}

.header .left-side h1 {
    font-size: 20px;
    font-weight: bold;
    margin: 0;
    color: inherit;
}
```

HomePage.tsx

```tsx
import Header from "../../components/header/Header";
import "./HomePage.css";
import { constants } from "../../../../setup/constants";

function HomePage() {
    return (
        <div className="home-page">
            <Header />
            <h2>Bienvenue sur votre nouvelle application web</h2>

            <h1>{constants.ROOT_FOLDER_NAME}</h1>
            <p>
                sera bientôt disponible pour offrir une expérience optimale à vos
                utilisateurs.
            </p>

            <section className="tech-stack">
                <h3>Architecture & Technologies</h3>
                <ul>
                    <li>
                        <strong>Frontend :</strong> React, Vite, React Router, TypeScript
                    </li>
                    <li>
                        <strong>Backend :</strong> Node.js, Express.js
                    </li>
                    <li>
                        <strong>Base de données :</strong> PostgreSQL
                    </li>
                    <li>
```

```
                        <strong>Déploiement :</strong> Docker, Docker Compose
                    </li>
                    <li>
                        <strong>Hébergement :</strong> Render.com
                    </li>
                    <li>
                        <strong>Outils & Qualité :</strong> Biome, NPM
                    </li>
                    <li>
                        <strong>CI/CD :</strong> Git, GitHub
                    </li>
                </ul>
            </section>
        </div>
    );
}

export default HomePage;
```

HomePage.css

```
body {
    padding-top: 72px;
}

.home-page {
    display: flex;
    flex-direction: column;
    align-items: center;
    text-align: center;
    padding: 120px var(--spacing) 40px;
    min-height: 100vh;
    background-color: var(--white-color);
    color: var(--teal-color);
}

.home-page h1 {
    font-size: 2.8rem;
    animation: fadeIn 0.8s ease-out;
    margin-bottom: 10px;
}

.home-page p {
    font-size: 1.2rem;
    max-width: 600px;
    margin-bottom: 30px;
    animation: fadeInUp 1s ease-out 0.3s both;
}

.tech-stack {
    margin-top: 40px;
    padding: 25px;
    border-radius: 12px;
    background-color: rgba(0, 0, 0, 0.03);
    animation: fadeInUp 1s ease-out 0.6s both;
    max-width: 500px;
    width: 100%;
}
```

```css
.tech-stack h3 {
    margin-bottom: 20px;
    font-size: 1.5rem;
    border-bottom: 2px solid var(--teal-color);
    display: inline-block;
    padding-bottom: 5px;
}

.tech-stack ul {
    list-style: none;
    padding: 0;
    text-align: left;
}

.tech-stack li {
    padding: 12px 0;
    font-size: 1rem;
    display: flex;
    justify-content: space-between;
    align-items: center;
    border-bottom: 1px dotted rgba(0, 0, 0, 0.1);
}

.tech-stack li:last-child {
    border-bottom: none;
}

.tech-stack li strong {
    color: var(--teal-color);
    white-space: nowrap;
}

.tech-stack li span {
    text-align: right;
    padding-left: 10px;
}

@media (max-width: 600px) {
    .tech-stack li {
        flex-direction: column;
        align-items: flex-start;
        gap: 5px;
        padding: 15px 0;
    }

    .tech-stack li span {
        text-align: left;
        padding-left: 0;
        font-size: 0.95rem;
        opacity: 0.9;
    }

    .home-page h1 {
        font-size: 2rem;
    }
}

@keyframes fadeIn {
```

```
    from {
        opacity: 0;
    }
    to {
        opacity: 1;
    }
}

@keyframes fadeInUp {
    from {
        opacity: 0;
        transform: translateY(20px);
    }
    to {
        opacity: 1;
        transform: translateY(0);
    }
}
```

NotFoundPage.tsx

```
import { useNavigate } from "react-router";
import Header from "../../components/header/Header";
import "./NotFoundPage.css";

function NotFoundPage() {
    const navigate = useNavigate();

    return (
        <div className="not-found-page">
            <Header />
            <h1>404 - Page non trouvée</h1>
            <button type="button" onClick={() => navigate("/")}>
                Retour à l'accueil
            </button>
        </div>
    );
}

export default NotFoundPage;
```

NotFoundPage.css

```
.not-found-page {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    text-align: center;

    min-height: 100vh;
    background-color: var(--primary-color);
    color: var(--secondary-color);
    padding: 40px;
}

.not-found-page h1 {
```

```
    font-size: 2.5rem;
    margin-bottom: 20px;
    animation: fadeIn 0.8s ease-out;
}

.not-found-page button {
    padding: 10px 20px;
    font-size: 1rem;
    font-weight: 600;
    border-radius: 8px;
    border: 2px solid var(--teal-color);
    background-color: var(--teal-color);
    color: var(--white-color);
    cursor: pointer;
    transition: all 0.3s ease;
}

.not-found-page button:hover {
    background-color: var(--white-color);
    color: var(--teal-color);
    border-color: var(--teal-color);
}
```

- 4.06 - client > package.json > Remplacer le contenu suivant :

```
"build": "tsc -b && vite build",
par
   "build": "npx vite build",
    "vite:build": "npx vite build",
```

- 4.07 - client > tsconfig.json > Coller le contenu suivant :

```
{
    "compilerOptions": {
        "target": "ESNext",
        "module": "ESNext",
        "moduleResolution": "node",
        "jsx": "react-jsx",
        "strict": true,
        "esModuleInterop": true,
        "skipLibCheck": true,
        "forceConsistentCasingInFileNames": true
    },
    "include": ["src"]
}
```

- 4.08 - client > src > Créer un fichier vite-env.d.ts
- 4.09 - client > src > vite-env.d.ts > Coller le contenu suivant :

```
/// <reference types="vite/client" />

interface ImportMetaEnv {
    readonly VITE_API_URL: string;
    readonly VITE_PROJECT_NAME: string;
}
```

```
declare global {
    interface ImportMeta {
        readonly env: ImportMetaEnv;
    }
}
```

- 4.10 - client > src > Créer un fichier router.tsx
- 4.11 - client > src > router.tsx > Coller le contenu suivant :

```
import { createBrowserRouter } from "react-router";
import HomePage from "../src/pages/homePage/HomePage";
import NotFoundPage from "../src/pages/notFoundPage/NotFoundPage";
import App from "./App";

const router = createBrowserRouter([
    {
        path: "/",
        element: <App />,
        errorElement: <NotFoundPage />,
        children: [{ index: true, element: <HomePage /> }],
    },
    {
        path: "*",
        element: <NotFoundPage />,
    },
]);
export default router;
```

- 4.12 - client > src > Supprimer le fichier app.css
- 4.13 - client > src > index.css > vider le contenu
- 4.14 - client > src > index.css > Renommer ce fichier global.css
- 4.15 - client > src > global.css > Coller le contenu suivant :

```
@import url("https://fonts.googleapis.com/css2?
family=Baloo+2:wght@400;600;800&display=swap");

/* Reset de base */
* {
    box-sizing: border-box;
}

:root {
    /* --- COLORS --- */
    --gold-color: #ffd700;
    --teal-color: #008080;
    --teal-dark-color: #004d4d;
    --white-color: #f1f4f9;

    /* --- SPACING --- */
    --spacing: 20px;
    --spacing-sm: calc(var(--spacing) / 2);
    --spacing-lg: calc(var(--spacing) * 2);

    /* --- TYPOGRAPHY --- */
    --font-main: "Baloo 2", cursive;
}

body {
```

```
    margin: 0;
    background-color: var(--white-color);
    color: var(--teal-color);
    font-family: var(--font-main);
    line-height: 1.6;
}

/* --- ANIMATIONS --- */
@keyframes fadeIn {
    from {
        opacity: 0;
        transform: translateY(-10px);
    }
    to {
        opacity: 1;
        transform: translateY(0);
    }
}

@keyframes fadeInUp {
    from {
        opacity: 0;
        transform: translateY(20px);
    }
    to {
        opacity: 1;
        transform: translateY(0);
    }
}
```

- 4.16 - client > src > app.tsx > Coller le contenu suivant :

```
import { useEffect } from "react";
import { Outlet } from "react-router";
import { ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";

import { constants } from "../../setup/constants";

function App() {
    useEffect(() => {
        const appName = constants.APP_NAME;

        document.title = appName;

        console.log(`🟢 CLIENT : Titre du projet configuré sur "${appName}"`);
    }, []);

    return (
        <>
            <Outlet />
            <ToastContainer position="bottom-right" aria-label="notification" />
        </>
    );
}

export default App;
```

- 4.17 - client > src > main.tsx > Remplacer le code par le contenu suivant :

```
import React from "react";
import ReactDOM from "react-dom/client";
import { RouterProvider } from "react-router";
import "./global.css";
import router from "../src/router";

const rootElement = document.getElementById("root");

if (!rootElement) {
    throw new Error("Root element not found");
}

ReactDOM.createRoot(rootElement).render(
    <React.StrictMode>
        <RouterProvider router={router} />
    </React.StrictMode>,
);
```

- 4.18 - client > .gitignore > Remplacer le code par le contenu suivant :

```
# Dépendances
node_modules/

# Builds
dist/
.vite/

# Env local
.env


# Logs
*.log
npm-debug.log*

# IDE / OS
.DS_Store
.vscode/
.idea/
```

- 4.19 - client > Créer un fichier .env.sample
- 4.20 - client > .env.sample > Coller le contenu suivant :

```
VITE_API_URL=http://localhost:3310/api
```

- 4.21 - client > Faire une copie du fichier .env.sample, puis renomme-le .env.
- 4.22 - client > vite.config.ts > Remplacer le code par le contenu suivant :

```
import path from "node:path";
import react from "@vitejs/plugin-react";
import { defineConfig, loadEnv } from "vite";

export default defineConfig(({ mode }) => {
    const env = loadEnv(mode, path.resolve(__dirname, "../"), "");
```

```
    const projectName = env.DB_NAME || "DefaultProject";
    const apiPort = env.SERVER_PORT || 3310;
    const clientPort = Number(env.CLIENT_PORT) || 5173;

    console.log("\n========================================");
    console.log("🟢 SECTION : CLIENT FRONTEND");
    console.log("========================================");
    console.log(`🟢 MODE    : ${mode}`);
    console.log(`🟢 PROJECT : ${projectName}`);
    console.log(`🟢 API URL : http://localhost:${apiPort}/api`);
    console.log("🟢 Configuration Vite chargée.\n");

    return {
        plugins: [react()],
        server: {
            port: clientPort,
            proxy: {
                "/api": {
                    target: env.VITE_API_URL || `http://localhost:${apiPort}`,
                    changeOrigin: true,
                },
            },
        },
        resolve: {
            alias: {
                "@": path.resolve(__dirname, "./src"),
            },
        },
    };
});
```

- 4.23 - client > tsconfig.node.json > Ajouter et remplcaer le contenu suivant :

```
{
    "compilerOptions": {
        "tsBuildInfoFile": "./node_modules/.tmp/tsconfig.node.tsbuildinfo",
        "useDefineForClassFields": true,
        "target": "ES2023",
        "lib": ["ES2022", "DOM", "DOM.Iterable"],
        "module": "ESNext",
        "types": ["vite/client"],
        "skipLibCheck": true,

        /* Bundler mode */
        "moduleResolution": "bundler",
        "allowImportingTsExtensions": true,
        "verbatimModuleSyntax": true,
        "moduleDetection": "force",
        "noEmit": true,

        /* Linting */
        "strict": true,
        "noUnusedLocals": true,
        "noUnusedParameters": true,
        "erasableSyntaxOnly": true,
        "noFallthroughCasesInSwitch": true,
        "noUncheckedSideEffectImports": true
```

```
    },
    "include": ["vite.config.ts"]
}
```

5. Finalisation et Test du Démarrage

- 5.01 - Ouvrir le terminal Ctrl+T
- 5.02 - Verifier d'être à la racine du projet
- 5.03 - Installer toutes les dépendances :

```
npm install
```

```
$ npm install

> projet@1.0.0 postinstall
> npm run setup


> projet@1.0.0 setup
> node setup/update-package-name.js


==========================
🔘 AUTO-CONFIG PROJECT
==========================

🔘 Dossier racine détecté : fullstack-starter-kit
🔘 Analyse de C:\Users\Cédric\Desktop\LOCAL\fullstack-starter-kit\setup\constants.ts...
🔘 constants.ts est déjà à jour (aucune mention 'PENDING_GENERATION').

🔘 Mise à jour des fichiers d'environnement...
🔘 Mise à jour de : server/.env...
🔘 .env (server) mis à jour avec PGDATABASE=fullstack-starter-kit
🔘 Mise à jour de : client/.env...
🔘 .env (client) mis à jour avec PGDATABASE=fullstack-starter-kit
🔘 Mise à jour de : fullstack-starter-kit/.env...
🔘 .env (fullstack-starter-kit) mis à jour avec PGDATABASE=fullstack-starter-kit

==========================
🔘 CONFIGURATION TERMINÉE
==========================


added 88 packages, and audited 89 packages in 15s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

- 5.04 - Lancer le logiciel Docker Desktop
- 5.05 - Vérifier si l'installation fonctionne :

```
npm run dev
```

```
 npm run dev
```

```
> projet@1.0.0 dev
> docker ps -q --filter "expose=5432" --filter "publish=5433" | xargs -r docker stop &&
docker-compose down --remove-orphans && docker-compose up -d && concurrently -c
"magenta,cyan" -t "HH:mm:ss" -p "[{name} {time}]" "npm run dev:server" "npm run dev:client"

41343b7acba8
[+] Running 2/2
 ✓ Container postgres-fullstack-starter-kit  Removed
0.0s
 ✓ Network fullstack-starter-kit_default     Removed
0.3s
[+] Running 2/2
 ✓ Network fullstack-starter-kit_default     Created
0.1s
 ✓ Container postgres-fullstack-starter-kit  Started
0.6s
[ 14:22:13]
[ 14:22:13] > projet@1.0.0 dev:server
[ 14:22:13] > npm run dev --prefix server
[ 14:22:13]
[ 14:22:13]
[ 14:22:13] > projet@1.0.0 dev:client
[ 14:22:13] > npm run dev --prefix client -- --host
[ 14:22:13]
[ 14:22:14]
[ 14:22:14] > server@1.0.0 dev
[ 14:22:14] > tsx src/index.ts
[ 14:22:14]
[ 14:22:14]
[ 14:22:14] > client@0.0.0 dev
[ 14:22:14] > vite --host
[ 14:22:14]
[ 14:22:15]
[ 14:22:15] ========================================
[ 14:22:15] 🔘 SECTION : CLIENT FRONTEND
[ 14:22:15] ========================================
[ 14:22:15] 🔘 MODE    : development
[ 14:22:15] 🔘 PROJECT : fullstack-starter-kit
[ 14:22:15] 🔘 API URL : http://localhost:3310/api
[ 14:22:15] 🔘 Configuration Vite chargée.
[ 14:22:15]
[ 14:22:15]
[ 14:22:15]   VITE v7.3.0  ready in 436 ms
[ 14:22:15]
[ 14:22:15]   ➜  Local:   http://localhost:5173/
[ 14:22:15]   ➜  Network: http://192.168.160.1:5173/
[ 14:22:15]   ➜  Network: http://192.168.1.30:5173/
[ 14:22:15] [dotenv@17.2.3] injecting env (10) from ..\.env -- tip: 🔄 add secrets lifecycle
management: https://dotenvx.com/ops
[ 14:22:15]
[ 14:22:15] ========================
[ 14:22:15] 🔘 DB SERVICE INIT
[ 14:22:15] ========================
[ 14:22:15] 🔘 DATABASE : fullstack-starter-kit
[ 14:22:15] 🔘 HOST     : localhost:5433
[ 14:22:15] 🔘 POOL     : Connecté avec cedevs
[ 14:22:15] ------------------------
[ 14:22:15]
[ 14:22:15] 🔘 Base de données "fullstack-starter-kit" : Structure créée ou mise à jour.
```

```
[ 14:22:15]
[ 14:22:15] ========================================
[ 14:22:15] 🔘 SECTION : SERVER BACKEND
[ 14:22:15] ========================================
[ 14:22:15] 🔘 DATABASE : CONNECTED
[ 14:22:15] 🔘 SERVER   : http://localhost:3310
[ 14:22:15] 🔘 CLIENT   : http://localhost:5173
[ 14:22:15]
```

6. Publication du Projet sur GitHub

- 6.01 - Revenir à la racine du projet à lier >
```
cd ..
```
- 6.02 - Formater les fichiers >
```
npx @biomejs/biome format --write
```
- 6.03 - Initialiser le dépôt Git >
```
git init
```
- 6.04 - Vérifier les fichiers à suivre >
```
git status
```
- 6.05 - Ajouter les fichiers au staging >
```
git add .
```
- 6.06 - Vérifier l'intégration dans le staging >
```
git status
```
- 6.07 - Créer un commit avec un message >
```
git commit -m "first project commit"
```
- 6.08 - Lier Git au dépôt GitHub (clé SSH ou HTTPS) >
```
git remote add origin clé
```
- 6.09 - Vérifier la connexion au dépôt distant >
```
git remote -v
```
- 6.10 - Envoyer le projet sur GitHub (branche main) >
```
git push -u origin main
```
- 6.11 - 🔘 Vérifier votre dépot sur github

7. Déploiement de l'Application sur Render.com

- 7.01 - Se rendre sur le dashboard de Render.com
- 7.02 - Ajouter un nouveau projet en cliquant sur "+ Add New"
- 7.03 - Choisir "Static Site"
- 7.04 - Sélectionner la source du code (connexion GitHub)
- 7.05 - Renseigner la commande de build :
```
cd client && npm install && npm run build
```
- 7.06 - Indiquer le répertoire de publication "Publish Directory":
```
client/dist
```
- 7.07 - Lancer le déploiement en cliquant sur "Deploy Static Site"
- 7.08 se rendre dans le menu Redirects/rewrites et aouter une nouvelle valeur : source :
```
/*
```
destination:
```
/index.html
```
Action: Selectionner Rewrite
- 7.09 - 🔘 Votre site est live
- 7.10 - Visualiser l'application web en cliquant sur le lien en haut de la page https://fullstack-starter-kit.onrender.com