DinoVisión

📘 Project Description

DinoVision is an accessible educational platform that offers an inclusive experience for people with visual impairments. The platform includes interactive courses, voice assistants and accessible navigation according to WCAG standards.

📝 Introduction

DinoVision is an accessible educational platform for people with visual disabilities. It integrates modern technologies to offer interactive courses and inclusive navigation based on web accessibility standards (WCAG).

Inspired by platforms such as Duolingo, Udemy and Moodle, DinoVision combines gamification elements, extensive educational resources and management tools to create an enriching experience for all users.

🚨 Problem

People with visual impairments face major challenges when accessing educational content online. DinoVision addresses these barriers by offering:

- Screen readers for easy navigation.
- Voice assistants to improve the user experience.
- Interactive course structures for effective learning.

🎯 Objective

Provide an inclusive digital environment that allows people with disabilities to develop new skills, participate in interactive educational experiences and access a variety of learning resources.

Create an accessible educational web platform for people with visual disabilities that:

- Integrate a screen reading system for keyboard navigation.
- Deploy a voice assistant to simplify interaction.
- Follow WCAG guidelines to ensure accessibility.

💡 Idea

The platform will be aimed at the distribution of interactive courses for learning in various areas Inspired by:

- Duolingo: Incentives and interactive exercises.
- Udemy: Broad and organized course structure.

Technologies used:

- **Interface**: React, JavaScript, CSS.
- **Server**: Node.js, MySQL.
- **Database**: MySQL.
- **Version** control: GitHub.

Additionally, we plan to develop a business model inspired by Moodle, offering accessible tools for educational institutions and businesses, while keeping the majority of courses free.

🌟 Benefits

- Equitable access to online educational resources.
- Simplified navigation for the visually impaired.
- Interactive and didactic tools that promote learning.

📚 Conclusion

DinoVision not only seeks to meet the UN Sustainable Development Goals, but also to offer a reliable and accessible tool for everyone. We believe that with the right tools, anyone can learn, grow and reach their full potential.

🚀 Installation and use

Prerequisites

1. **Node.js** installed on your system.
2. **MongoDB** configured and running locally or on a cloud service.

Configuration

1. Clone the repository:
   ```
   git clone https://github.com/UrielMonarrez17/Desarrollo_web.git
   cd Desarrollo_web
   ```
2. Install dependencies:

```
npm install

npm run client-install
```

3. Configure the `.env` file with the necessary environment variables.

```
MONGO_USER=usuario_mongo
MONGO_PASS=contraseña_mongo
MONGO_DB=nombre_base_datos
JWT_SECRET=clave_secreta_jwt
API_KEY=clave_api
```

4. Run the project in development mode:
   `npm run dev`

Guiones Disponibles

Definidos en `package.json`:

- `npm start`: Inicia el servidor en modo producción.
- `npm run server`: Inicia el servidor con nodemon.
- `npm run client`:Inicia el cliente React.
- `npm run dev`: Ejecuta el servidor y cliente simultáneamente.

🔧 Technologies used

Backend

- **Node.js** with Express for the server.
- **MongoDB** and **Mongoose** for the database.
- **JWT** and **bcrypt** for authentication and security.

Interface

- **React** for UI.
- **React-Scripts** for configuration and development tools.

Other dependencies

- **dotenv**: Management of environment variables.
- **cors** : Allow CORS requests.
- **axios**: HTTP requests.

📂 Project structure

```
Desarrollo_web/
├── backend/     # Server logic and database management.
├── client/      # User interface developed with React
├── Documentation/ # Documentation related files
├── .env         # environment variables
├── .gitignore   # Files and folders ignored by Git
├── package.json  # Dependencies and scripts
├── server.js    # Server configuration
└── README.md    # Main documentation
```

🗀 Backend structure (backend/)

The `backend` folder contains the server logic and models for the MongoDB database.

```
backend/
├── models/                 # Data models for MongoDB
│   ├── cursos.js
│   ├── cursosInside.js
│   ├── filters.js
│   ├── lessons_menu.js
│   ├── sub-lessons.js
│   └── user.js
├── api.js                  # Define API routes and logic
├── connectDB.js            # MongoDB connection configuration
├── database.js             # Database related endpoints
└── front.js                # integrations between frontend and backend
```

🗀 Client Structure (client/)

The client folder contains the frontend of the application developed with React.

```
client/
 ├── public/                # Browser-accessible static files
 │   ├── favicon.ico
 │   ├── index.css
 │   ├── index.html
 │   ├── logo_prueba.ico
 │   ├── logo-risk.png
 │   ├── logo192.png
 │   ├── logo512.png
 │   ├── manifest.json      # Progressive Web App (PWA) Setup
 │   └── robots.txt         # Indexing control by search engines
 ├── src/                    # Main source code
```

```
│   ├── auth/                # Authentication related modules
│   ├── images/              # Image Resources
│   ├── styles/              # CSS style files
│   ├── Views/               # Main app views
│   ├── App.js               # root component
│   ├── constants.js         # project constants
│   └── index.js             # main entrance
├── .gitignore               # Files to ignore in the repository
├── package-lock.json
└── package.json
```

🌐 Server Functionality

The server.js file configures the Express server:

- **Available endpoints:**
  - /database: Database management (implemented in backend/database).
  - /api: General endpoints for the application.

- **Production:** Serves the static files generated by the React client.

Server configuration example:

```
const express = require('express');
const app = express();
const PORT = process.env.PORT || 5000;

app.use(express.json());
app.use('/api', require('./backend/api'));

app.listen(PORT, () => console.log(`Servidor corriendo en el puerto ${PORT}`));
```

📈 Data models

Models defined in backend/models include:

- **User**: Information about registered users.
- **Courses**: Structure of the available courses.
- **Filters**: Setting filters for searches.
- **Lessons_menu**: Content organized by lessons.
- **Sublessons**: Subtopics of each lesson.

🛠️ Contribution

1. Fork the project.
2. Create a branch for your changes:
   ```
   git checkout -b mi-rama
   ```
3. Make your changes and commit:
   ```
   git commit -m "Descripción de los cambios"
   ```

4. Envíe una solicitud de extracción.

🤝 Authors

- Juan Pablo
- Uriel Monarrez
- Diego Alexis