**1. Configura el servidor i la base de dades**

**1.1. Configuració del servidor amb Node.js i Express**

* Instal·la Node.js i npm (si no ho tens ja instal·lat).
* Crea una carpeta per al projecte i inicialitza un projecte Node.js:

mkdir arbres-monumentals-cat

cd arbres-monumentals-cat

npm init -y

bash

Copy

mkdir arbres-monumentals-cat

cd arbres-monumentals-cat

npm init -y

* Instal·la les dependències necessàries:

npm install express mysql2 body-parser cors

bash

Copy

npm install express mysql2 body-parser cors

* Crea l'arxiu server.js i configura el servidor:

const express = require('express');

const mysql = require('mysql2');

const bodyParser = require('body-parser');

const cors = require('cors');

const app = express();

const port = 3000;

// Configuració de la base de dades

const db = mysql.createConnection({

host: 'localhost',

user: 'root',

password: '', // Introdueix la teva contrasenya

database: 'arbres\_monumentals'

});

db.connect((err) => {

if (err) throw err;

console.log('Conectat a la base de dades');

});

// Middleware

app.use(bodyParser.urlencoded({ extended: false }));

app.use(bodyParser.json());

app.use(cors());

// Ruta d'inici

app.get('/', (req, res) => {

res.send('Benvingut a Arbres Monumentals de Catalunya!');

});

// Iniciar el servidor

app.listen(port, () => {

console.log(`Servidor escoltant a http://localhost:${port}`);

});

javascript

Copy

const express = require('express');

const mysql = require('mysql2');

const bodyParser = require('body-parser');

const cors = require('cors');

const app = express();

const port = 3000;

// Configuració de la base de dades

const db = mysql.createConnection({

host: 'localhost',

user: 'root',

password: '', // Introdueix la teva contrasenya

database: 'arbres\_monumentals'

});

db.connect((err) => {

if (err) throw err;

console.log('Conectat a la base de dades');

});

// Middleware

app.use(bodyParser.urlencoded({ extended: false }));

app.use(bodyParser.json());

app.use(cors());

// Ruta d'inici

app.get('/', (req, res) => {

res.send('Benvingut a Arbres Monumentals de Catalunya!');

});

// Iniciar el servidor

app.listen(port, () => {

console.log(`Servidor escoltant a http://localhost:${port}`);

});

**1.2. Configuració de la base de dades**

* Utilitza MySQL o PostgreSQL. Aquí utilitzarem MySQL com a exemple.
* Crea la base de dades i les taules:

CREATE DATABASE arbres\_monumentals;

USE arbres\_monumentals;

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(50) NOT NULL,

email VARCHAR(100) NOT NULL,

password VARCHAR(255) NOT NULL

);

CREATE TABLE trees (

id INT AUTO\_INCREMENT PRIMARY KEY,

scientific\_name VARCHAR(100) NOT NULL,

popular\_name VARCHAR(100) NOT NULL,

description TEXT,

photo\_url VARCHAR(255),

location\_id INT,

FOREIGN KEY (location\_id) REFERENCES locations(id)

);

CREATE TABLE locations (

id INT AUTO\_INCREMENT PRIMARY KEY,

municipality VARCHAR(100) NOT NULL,

coordinates POINT NOT NULL,

owner\_email VARCHAR(100),

town\_hall\_email VARCHAR(100) NOT NULL

);

sql

Copy

CREATE DATABASE arbres\_monumentals;

USE arbres\_monumentals;

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(50) NOT NULL,

email VARCHAR(100) NOT NULL,

password VARCHAR(255) NOT NULL

);

CREATE TABLE trees (

id INT AUTO\_INCREMENT PRIMARY KEY,

scientific\_name VARCHAR(100) NOT NULL,

popular\_name VARCHAR(100) NOT NULL,

description TEXT,

photo\_url VARCHAR(255),

location\_id INT,

FOREIGN KEY (location\_id) REFERENCES locations(id)

);

CREATE TABLE locations (

id INT AUTO\_INCREMENT PRIMARY KEY,

municipality VARCHAR(100) NOT NULL,

coordinates POINT NOT NULL,

owner\_email VARCHAR(100),

town\_hall\_email VARCHAR(100) NOT NULL

);

**2. Implementa les funcionalitats d'autenticació i gestió d'arbres**

**2.1. Autenticació d'usuaris**

* Instal·la passport i passport-local:

npm install passport passport-local bcrypt

bash

Copy

npm install passport passport-local bcrypt

* Configura l'autenticació a server.js:

const passport = require('passport');

const LocalStrategy = require('passport-local').Strategy;

const bcrypt = require('bcrypt');

// Configuració de Passport

passport.use(new LocalStrategy(

async (username, password, done) => {

const [rows] = await db.promise().query('SELECT \* FROM users WHERE username = ?', [username]);

if (rows.length === 0) return done(null, false);

const user = rows[0];

if (await bcrypt.compare(password, user.password)) {

return done(null, user);

} else {

return done(null, false);

}

}

));

passport.serializeUser((user, done) => done(null, user.id));

passport.deserializeUser(async (id, done) => {

const [rows] = await db.promise().query('SELECT \* FROM users WHERE id = ?', [id]);

done(null, rows[0]);

});

app.use(passport.initialize());

app.use(passport.session());

javascript

Copy

const passport = require('passport');

const LocalStrategy = require('passport-local').Strategy;

const bcrypt = require('bcrypt');

// Configuració de Passport

passport.use(new LocalStrategy(

async (username, password, done) => {

const [rows] = await db.promise().query('SELECT \* FROM users WHERE username = ?', [username]);

if (rows.length === 0) return done(null, false);

const user = rows[0];

if (await bcrypt.compare(password, user.password)) {

return done(null, user);

} else {

return done(null, false);

}

}

));

passport.serializeUser((user, done) => done(null, user.id));

passport.deserializeUser(async (id, done) => {

const [rows] = await db.promise().query('SELECT \* FROM users WHERE id = ?', [id]);

done(null, rows[0]);

});

app.use(passport.initialize());

app.use(passport.session());

**2.2. Gestió d'arbres**

* Crea una API per afegir i obtenir arbres:

// Afegir un arbre

app.post('/trees', async (req, res) => {

const { scientific\_name, popular\_name, description, photo\_url, location\_id } = req.body;

const [result] = await db.promise().query(

'INSERT INTO trees (scientific\_name, popular\_name, description, photo\_url, location\_id) VALUES (?, ?, ?, ?, ?)',

[scientific\_name, popular\_name, description, photo\_url, location\_id]

);

res.status(201).json({ id: result.insertId });

});

// Obtenir tots els arbres

app.get('/trees', async (req, res) => {

const [rows] = await db.promise().query('SELECT \* FROM trees');

res.json(rows);

});

javascript

Copy

// Afegir un arbre

app.post('/trees', async (req, res) => {

const { scientific\_name, popular\_name, description, photo\_url, location\_id } = req.body;

const [result] = await db.promise().query(

'INSERT INTO trees (scientific\_name, popular\_name, description, photo\_url, location\_id) VALUES (?, ?, ?, ?, ?)',

[scientific\_name, popular\_name, description, photo\_url, location\_id]

);

res.status(201).json({ id: result.insertId });

});

// Obtenir tots els arbres

app.get('/trees', async (req, res) => {

const [rows] = await db.promise().query('SELECT \* FROM trees');

res.json(rows);

});

**3. Millora el disseny i la interactivitat del mapa**

**3.1. Integració de Leaflet.js**

* Afegeix un mapa interactiu a la pàgina principal:

<div id="map" style="height: 500px;"></div>

<script>

const map = L.map('map').setView([41.5912, 1.5209], 8); // Coordenades de Catalunya

L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {

maxZoom: 18,

}).addTo(map);

// Afegir marcadors per als arbres

fetch('/trees')

.then(response => response.json())

.then(trees => {

trees.forEach(tree => {

L.marker([tree.latitude, tree.longitude]).addTo(map)

.bindPopup(`<b>${tree.popular\_name}</b><br>${tree.scientific\_name}`);

});

});

</script>

html

Copy

<div id="map" style="height: 500px;"></div>

<script>

const map = L.map('map').setView([41.5912, 1.5209], 8); // Coordenades de Catalunya

L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {

maxZoom: 18,

}).addTo(map);

// Afegir marcadors per als arbres

fetch('/trees')

.then(response => response.json())

.then(trees => {

trees.forEach(tree => {

L.marker([tree.latitude, tree.longitude]).addTo(map)

.bindPopup(`<b>${tree.popular\_name}</b><br>${tree.scientific\_name}`);

});

});

</script>

Run HTML

**3.2. Disseny responsive amb Bootstrap**

* Utilitza Bootstrap per crear un disseny atractiu i responsive.

**4. Prova i depura el projecte**

* Prova totes les funcionalitats:
  + Registre i autenticació d'usuaris.
  + Afegir i visualitzar arbres.
  + Interactuar amb el mapa.
* Utilitza eines com Postman per provar les APIs.
* Depura errors amb console.log o eines com nodemon per reiniciar el servidor automàticament.

**5. Desplega la pàgina web en un servidor**

**5.1. Desplegament amb Heroku**

* Instal·la Heroku CLI i crea un compte.
* Inicialitza un repositori Git:

git init

git add .

git commit -m "Initial commit"

bash

Copy

git init

git add .

git commit -m "Initial commit"

* Crea una aplicació a Heroku:

heroku create

bash

Copy

heroku create

* Desplega el projecte:

git push heroku main

bash

Copy

git push heroku main

**5.2. Configuració de la base de dades a Heroku**

* Utilitza un add-on de Heroku com **Heroku Postgres** per gestionar la base de dades.

**5.3. Configuració del domini**

* Configura un domini personalitzat des del panell de control de Heroku.

n