Curso de Programación Desde Cero

Tema 8: Proyecto Final Full-Stack

Duración estimada: 25 horas

1. Introducción al Desarrollo Full-Stack

Un desarrollador Full-Stack domina tanto el frontend (interfaz de usuario) como el backend (servidor y base de datos), siendo capaz de crear aplicaciones web completas de extremo a extremo.

Componentes de una Aplicación Full-Stack:

2 Frontend: React, HTML, CSS (interfaz de usuario)
2 Backend: Node.js, Express (lógica del servidor)
2 Base de Datos: MongoDB o SQL (almacenamiento)
2 APIs: Comunicación entre frontend y backend
2 Autenticación: Gestión de usuarios y seguridad
2 Deployment: Poner la aplicación en producción

2. Planificación del Proyecto

Vamos a desarrollar "TaskMaster", una aplicación completa de gestión de tareas y proyectos para equipos de trabajo.

Características Principales:

2 Sistema de autenticación de usuarios
2 Creación y gestión de proyectos
2 Asignación de tareas con estados
2 Colaboración en tiempo real
2 Dashboard con estadísticas
2 Notificaciones y comentarios
2 Diseæo responsivo

3. Arquitectura de la Aplicación

Estructura del Proyecto:

```
taskmaster/
'" client/             # Frontend React
'   '" src/
'   '   '" components/
'   '   '" pages/
'   '   '" hooks/
'   '   '" services/
'   '   ÀÀÀutils/
'   '" public/
'   ÀÀÀpackage.json
'" server/             # Backend Node.js
```

4. Diseæo de la Base de Datos

Utilizaremos MongoDB para almacenar nuestros datos. Diseæaremos esquemas eficientes y escalables.

Modelo de Usuario:

```
const userSchema = {
  _id: ObjectId,
  email: String (unique, required),
  password: String (hashed, required),
  firstName: String (required),
  lastName: String (required),
  avatar: String (URL),
  role: String (admin, user),
  createdAt: Date,
  updatedAt: Date,
  isActive: Boolean
}
```

Modelo de Proyecto:

```
const projectSchema = {
  _id: ObjectId,
  name: String (required),
  description: String,
  owner: ObjectId (ref: User),
  members: [ObjectId] (ref: User),
  status: String (active, completed, paused),
  priority: String (low, medium, high),
  deadline: Date,
  createdAt: Date,
  updatedAt: Date
}
```

Modelo de Tarea:

```
const taskSchema = {
  _id: ObjectId,
  title: String (required),
  description: String,
  project: ObjectId (ref: Project),
  assignedTo: ObjectId (ref: User),
  createdBy: ObjectId (ref: User),
  status: String (todo, in-progress, review, done),
  priority: String (low, medium, high),
  dueDate: Date,
  tags: [String],
  comments: [{
    user: ObjectId (ref: User),
    text: String,
    createdAt: Date
  }],
  createdAt: Date,
  updatedAt: Date
}
```

5. Configuraciœn del Backend

Instalaciœn de Dependencias:

6. Implementacióń del Backend

Archivo Principal (server/app.js):

```
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
require('dotenv').config();
const app = express();
const PORT = process.env.PORT || 5000;
// Middleware
app.use(cors());
app.use(express.json({ limit: '10mb' }));
app.use(express.urlencoded({ extended: true }));
// Rutas
app.use('/api/auth', require('./routes/auth'));
app.use('/api/projects', require('./routes/projects'));
app.use('/api/tasks', require('./routes/tasks'));
// Conexióń a MongoDB
mongoose.connect(process.env.MONGODB_URI)
  .then(() => console.log('MongoDB conectado'))
  .catch(err => console.error(err));
app.listen(PORT, () => {
  console.log(`Servidor ejecutÆndose en puerto ${PORT}`);
});
```

Controlador de Autenticacióń (auth.js):

```
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const User = require('../models/User');
exports.register = async (req, res) => {
  try {
    const { email, password, firstName, lastName } = req.body;

    const existingUser = await User.findOne({ email });
    if (existingUser) {
      return res.status(400).json({ message: 'Usuario ya existe' });
    }

    const hashedPassword = await bcrypt.hash(password, 12);

    const user = new User({
      email, password: hashedPassword, firstName, lastName
    });

    await user.save();

    const token = jwt.sign(
      { userId: user._id },
      process.env.JWT_SECRET,
      { expiresIn: '7d' }
    );

    res.status(201).json({ token, user: { id: user._id, email, firstName, lastName } });
  } catch (error) {
    res.status(500).json({ message: 'Error del servidor' });
  }
};
```

7. Desarrollo del Frontend con React

Configuracio ́n del Frontend:

```
npx create-react-app client
cd client
npm install axios react-router-dom
npm install @mui/material @emotion/react @emotion/styled
npm install @mui/icons-material
npm install react-hook-form yup @hookform/resolvers
```

Estructura de Componentes:

```
src/
"' components/
'   "' common/
'   '   "' Header.jsx
'   '   "' Sidebar.jsx
'   '   ÀÀÀLoading.jsx
'   "' auth/
'   '   "' LoginForm.jsx
'   '   ÀÀÀRegisterForm.jsx
'   "' projects/
'   '   "' ProjectList.jsx
'   '   "' ProjectCard.jsx
'   '   ÀÀÀProjectForm.jsx
'   ÀÀÀtasks/
'       "' TaskBoard.jsx
'       "' TaskCard.jsx
'       ÀÀÀTaskForm.jsx
"' pages/
'   "' Dashboard.jsx
'   "' Projects.jsx
'   "' Tasks.jsx
'   ÀÀÀProfile.jsx
"' hooks/
'   "' useAuth.js
'   ÀÀÀuseApi.js
"' services/
'   "' authService.js
'   "' projectService.js
'   ÀÀÀtaskService.js
"' context/
'   ÀÀÀAuthContext.js
ÀÀÀApp.js
```

Componente Principal (App.js):

```
import React from ’react’;
import { BrowserRouter as Router, Routes, Route } from ’react-router-dom’;
import { ThemeProvider, createTheme } from ’@mui/material/styles’;
import CssBaseline from ’@mui/material/CssBaseline’;
import { AuthProvider } from ’./context/AuthContext’;
import ProtectedRoute from ’./components/common/ProtectedRoute’;
import Login from ’./pages/Login’;
import Dashboard from ’./pages/Dashboard’;
import Projects from ’./pages/Projects’;
import Tasks from ’./pages/Tasks’;
const theme = createTheme({
  palette: {
    primary: { main: ’#1976d2’ },
    secondary: { main: ’#dc004e’ }
  }
});
function App() {
```

8. Implementacióń de Autenticacióń

Context de Autenticacióń (AuthContext.js):

```
import React, { createContext, useContext, useReducer, useEffect } from 'react';
import authService from '../services/authService';
const AuthContext = createContext();
const initialState = {
  user: null,
  token: localStorage.getItem('token'),
  isLoading: true,
  isAuthenticated: false
};
const authReducer = (state, action) => {
  switch (action.type) {
    case 'LOGIN_SUCCESS':
      localStorage.setItem('token', action.payload.token);
      return {
        ...state,
        user: action.payload.user,
        token: action.payload.token,
        isAuthenticated: true,
        isLoading: false
      };
    case 'LOGOUT':
      localStorage.removeItem('token');
      return { ...initialState, isLoading: false };
    default:
      return state;
  }
};
```

Hook useAuth:

```
export const useAuth = () => {
  const context = useContext(AuthContext);
  if (!context) throw new Error('useAuth must be used within AuthProvider');
  return context;
};
```

Componente de Login:

```
import React from 'react';
import { useForm } from 'react-hook-form';
import { Box, Card, TextField, Button, Typography, Alert } from '@mui/material';
import { useAuth } from '../context/AuthContext';
const LoginForm = () => {
  const { login, isLoading, error } = useAuth();
  const { register, handleSubmit, formState: { errors } } = useForm();
  const onSubmit = async (data) => {
    try {
      await login(data.email, data.password);
    } catch (err) {
      console.error('Error de login:', err);
    }
  };
  return (
    <Box display="flex" justifyContent="center" alignItems="center" minHeight="100vh">
      <Card sx={{ p: 4, maxWidth: 400, width: '100%' }}>
        <Typography variant="h4" align="center" gutterBottom>
          TaskMaster
        </Typography>
        {error && <Alert severity="error">{error}</Alert>}
        <Box component="form" onSubmit={handleSubmit(onSubmit)} sx={{ mt: 2 }}>
          <TextField
            fullWidth margin="normal" label="Email"
            {...register('email', { required: 'Email es requerido' })}
            error={!!errors.email}
            helperText={errors.email?.message}
          />
          <TextField
            fullWidth margin="normal" label="Contraseæa" type="password"
```

9. Dashboard y Gestión de Proyectos

Componente Dashboard:

```
import React, { useState, useEffect } from 'react';
import { Grid, Card, CardContent, Typography, Box } from '@mui/material';
import { BarChart, Bar, XAxis, YAxis, CartesianGrid, Tooltip, ResponsiveContainer } from 'recharts';
import projectService from '../services/projectService';
import taskService from '../services/taskService';
const Dashboard = () => {
  const [stats, setStats] = useState({
    totalProjects: 0,
    activeTasks: 0,
    completedTasks: 0,
    teamMembers: 0
  });
  const [chartData, setChartData] = useState([]);
  useEffect(() => {
    const fetchDashboardData = async () => {
      try {
        const [projects, tasks] = await Promise.all([
          projectService.getProjects(),
          taskService.getTasks()
        ]);
        setStats({
          totalProjects: projects.length,
          activeTasks: tasks.filter(t => t.status !== 'done').length,
          completedTasks: tasks.filter(t => t.status === 'done').length,
          teamMembers: new Set(projects.flatMap(p => p.members)).size
        });
        const statusCounts = tasks.reduce((acc, task) => {
          acc[task.status] = (acc[task.status] || 0) + 1;
          return acc;
        }, {});
        setChartData(Object.entries(statusCounts).map(([status, count]) => ({
          status: status.charAt(0).toUpperCase() + status.slice(1),
          count

      } catch (error) {
        console.error('Error fetching dashboard data:', error);
      }
    };
    fetchDashboardData();
  }, []);
```

Sistema de Gestión de Estado:

```
// useProjects hook
import { useState, useEffect } from 'react';
import projectService from '../services/projectService';
export const useProjects = () => {
  const [projects, setProjects] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const fetchProjects = async () => {
    try {
      setLoading(true);
      const data = await projectService.getProjects();
      setProjects(data);
    } catch (err) {
      setError(err.message);
    } finally {
      setLoading(false);
    }
  };
  const createProject = async (projectData) => {
    try {
      const newProject = await projectService.createProject(projectData);
      setProjects(prev => [...prev, newProject]);
      return newProject;
    } catch (err) {
```

10. Testing y Deployment

Configuracion de Tests:

```
npm install --save-dev jest supertest
npm install --save-dev @testing-library/react @testing-library/jest-dom
```

Test de API (backend):

```
const request = require('supertest');
const app = require('../app');
describe('Auth Endpoints', () => {
  test('POST /api/auth/register', async () => {
    const userData = {
      email: 'test@test.com',
      password: 'password123',
      firstName: 'Test',
      lastName: 'User'
    };
    const response = await request(app)
      .post('/api/auth/register')
      .send(userData)
      .expect(201);
    expect(response.body).toHaveProperty('token');
    expect(response.body.user).toHaveProperty('email', userData.email);
  });
});
```

Variables de Entorno (.env):

```
NODE_ENV=production
PORT=5000
MONGODB_URI=mongodb+srv://username:password@cluster.mongodb.net/taskmaster
JWT_SECRET=super_secret_jwt_key_change_in_production
CORS_ORIGIN=https://taskmaster-frontend.vercel.app
```

Deployment con Vercel (Frontend):

```
# vercel.json
{
  "builds": [
    { "src": "package.json", "use": "@vercel/static-build" }
  ],
  "routes": [
    { "src": "/static/(.*)", "dest": "/static/$1" },
    { "src": "/(.*)", "dest": "/index.html" }
  ]
}
```

Deployment con Railway (Backend):

```
# railway.json
{
  "$schema": "https://railway.app/railway.schema.json",
  "build": {
    "builder": "NIXPACKS"
  },
  "deploy": {
    "startCommand": "npm start",
    "restartPolicyType": "ON_FAILURE",
    "restartPolicyMaxRetries": 10
  }
}
```