Memoria - Portal Web

Memoria del Laboratorio 1 — Portal Web

Asignatura: Desarrollo Web / Prácticas

Alumno: Favaron Giacomo

Link App: https://express-app-44zm.onrender.com

Resumen

Este documento recoge el desarrollo de un portal web mínimo solicitado en el laboratorio. El objetivo era crear una aplicación Express con tres versiones: una página de bienvenida personalizada (`/`), una nueva ruta `/servicios` con listado de servicios, y una página `/mejoras` donde se describan propuestas y mejoras implementadas.

Índice

---

1. Objetivo

Crear un pequeño portal web con diferentes vistas según la ruta elegida y documentar el código nuevo implementado. El trabajo se entregará en GitHub y debe poder desplegarse en Render.

2. Entorno y herramientas

- Node.js (>=14)
- npm
- Express (generado con `express-generator`)
- EJS como motor de vistas
- Editor de texto (VS Code o otro)

Herramientas adicionales utilizadas

- `git` para control de versiones
- `curl` o navegador para probar rutas
- `pandoc` y una distribución TeX si se desea generar PDF localmente

Si se desplegará en Render, se usa la integración con GitHub y el `start` script definido en `package.json`.

3. Estructura del proyecto (relevante)

- `app.js` — Configuración Express
- `bin/www` — Script de arranque
- `routes/index.js` — Rutas del sitio (se añadieron `/servicios` y `/mejoras`)
- `views/` — Plantillas EJS (`index.ejs`, `servicios.ejs`, `mejoras.ejs`)
- `public/` — Recursos estáticos (`stylesheets/style.css`, `images/`)
- `lab_report_full.md` — Memoria (este archivo)

Estructura de carpetas (resumen):

```
render_app/
- app.js
- bin/www
- package.json
- routes/
  - index.js
- views/
  - index.ejs
  - servicios.ejs
  - mejoras.ejs
- public/
  - images/
  - stylesheets/style.css
```

4. Desarrollo por versiones

v1 — Página de bienvenida (`/`) - Personalicé la página de inicio para que represente la portada de la "empresa" (ACME Consulting en el ejemplo), con cabecera, navegación y sección hero. - Se sustituyó el contenido por defecto por un diseño sencillo y profesional.

v2 — Nueva ruta `/servicios` - Añadida la ruta y la vista para mostrar un

listado de servicios que ofrece la empresa. Los datos se sirven desde el servidor como un array simple.

v3 — `/mejoras` - Añadida la ruta y la vista que detalla las mejoras propuestas y señala las implementadas (navegación, estilos, ruta de servicios, hero con imagen).

Detalles por versión (pasos y decisiones)

- v1: Después de generar el esqueleto con `express-generator`, se personalizó `views/index.ejs` y `public/stylesheets/style.css`. Se preservó la configuración de vista EJS en `app.js`.
- v2: Se añadió lógica en `routes/index.js` para proporcionar un array `services` al renderizar la vista `servicios.ejs`. Esta aproximación es suficiente para el laboratorio; la persistencia no fue requerida.
- v3: Se añadió `mejoras.ejs` y se documentaron las mejoras. También se reorganizó el `hero` para mostrar la imagen y el botón superpuesto.

5. Fragmentos de código relevantes

A continuación se incluyen extractos de los cambios más importantes realizados durante el laboratorio. Estos fragmentos se incluyen como ejemplos para que el corrector identifique qué se programó.

- `routes/index.js` (añadidas rutas `/servicios` y `/mejoras`):

```
var express = require('express');
var router = express.Router();

/* GET home page. */
router.get('/', function(req, res, next) {
  res.render('index', { title: 'ACME Consulting', description: 'Soluciones tecnológicas a me
});

/* GET servicios page. */
router.get('/servicios', function(req, res, next) {
  const services = [
    { name: 'Desarrollo Web', desc: 'Construcción de aplicaciones web modernas.' },
    { name: 'Consultoría IT', desc: 'Asesoría en transformación digital.' },
    { name: 'Soporte y Mantenimiento', desc: 'Planes de soporte 24/7.' }
  ];
  res.render('servicios', { title: 'Servicios - ACME Consulting', services: services });
});

/* GET mejoras page. */
router.get('/mejoras', function(req, res, next) {
  const improvements = [
    'Navegación principal con accesos a secciones',
    'Estilos básicos y diseño responsive',
```

```
    'Página de servicios con datos dinámicos'
  ];
  res.render('mejoras', { title: 'Mejoras propuestas', improvements: improvements });
});

module.exports = router;
```

- `views/index.ejs` (hero full-bleed con overlay y botón sobre la imagen):

```
<section class="hero">
  <div class="hero-figure">
    <img src="/images/lab-image.png" alt="Imagen del laboratorio" class="hero-image" />
    <div class="hero-overlay">
      <h2 class="hero-title"><%= title %></h2>
      <p class="lead"><%= description %></p>
      <a class="btn btn-hero" href="/servicios">Ver servicios</a>
    </div>
  </div>
</section>
```

- `views/servicios.ejs` (listado dinámico de servicios):

```
<h2>Servicios</h2>
<ul class="services">
  <% services.forEach(function(s){ %>
    <li class="service-item">
      <h3><%= s.name %></h3>
      <p><%= s.desc %></p>
    </li>
  <% }); %>
</ul>
```

- `views/mejoras.ejs` (propuestas y mejora implementada):

```
<h2>Mejoras propuestas</h2>
<ol>
  <% improvements.forEach(function(it){ %>
    <li><%= it %></li>
  <% }); %>
</ol>

<section>
  <h3>Mejora implementada</h3>
  <p>Se añadió navegación superior y estilos básicos. También se creó la ruta <code>/servici
</section>
```

- Fragmento de `public/stylesheets/style.css` (estilos del hero y botón):

```
.hero-figure{position:relative;width:100%;height:420px;overflow:hidden}
.hero-image{width:100%;height:100%;object-fit:cover;display:block}
```

```
.hero-overlay{position:absolute;left:50%;top:50%;transform:translate(-50%,-50%);color:#fff;t
.btn-hero{padding:12px 18px;font-weight:600;box-shadow:0 6px 18px rgba(0,0,0,0.2)}
```

6. Comandos para ejecución local y con Render

- Ejecutar localmente:

```
cd render_app
npm install
npm start
# Abrir http://localhost:3000/
```

Para la ejecución con Render abrir Render (https://render.com/) y crear un Web Service con node como lenguaje. Después conectalo al repo del projecto y pulsa 'Build'.

link app: https://express-app-44zm.onrender.com

7. Control de versiones (sugerencia para entrega)

Hacer commits y tags por versión, por ejemplo:

```
# después de implementar la versión 1
git add .
git commit -m "v1: Página de bienvenida personalizada"
git tag -a v1 -m "Versión 1 - Página de bienvenida"

# después de implementar la versión 2
git commit -am "v2: Añadida ruta /servicios y vista"
git tag -a v2 -m "Versión 2 - /servicios"

# después de implementar la versión 3
git commit -am "v3: Añadida /mejoras y mejoras de UI"
git tag -a v3 -m "Versión 3 - /mejoras"

# subir tags y commits
git push origin main --tags
```

8. Conclusiones

- Se ha implementado un portal web básico con las tres versiones solicitadas.
- El código añadido está documentado en este informe con fragmentos representativos.
- Como mejoras futuras se propone añadir persistencia (base de datos), formulario de contacto y despliegue automático en Render mediante integración con GitHub.