Registro Diario de Avances – Sprint 3 TraveDesk

**Fecha:**

**Autor:**

**Versión:** 1.0

# 1. Introducción

El Sprint 3 del proyecto TravelDesk se enfoca en la implementación de la gestión de usuarios y autenticación, abarcando el registro, inicio de sesión y control de roles y permisos. Durante este sprint, se desarrollarán las interfaces necesarias, se establecerán las validaciones de datos y credenciales, y se configurará la estructura de roles en la base de datos para asegurar un acceso seguro y controlado al sistema. Con estas funcionalidades, se busca garantizar la protección de la información y definir los niveles de acceso según el perfil de cada usuario, fortaleciendo así la seguridad y usabilidad de la plataforma.

# 2. Lineamientos de Diseño

El diseño de las interfaces y componentes del módulo de gestión de usuarios y autenticación debe mantener coherencia visual con la identidad general del sistema TravelDesk, priorizando la usabilidad, simplicidad y seguridad. Se utilizará una paleta de colores institucional, tipografía legible y componentes consistentes que faciliten la interacción del usuario. Los formularios de registro e inicio de sesión deben ser claros, con campos bien identificados, mensajes de error visibles y validaciones inmediatas. Además, se debe garantizar un diseño responsivo, adaptable a distintos dispositivos, y seguir principios de accesibilidad para usuarios con diferentes necesidades. En el panel de administración de roles y permisos, se priorizará una distribución ordenada y jerárquica de la información, permitiendo una gestión intuitiva y segura de los usuarios del sistema.

# 3. Historias de Usuario y Tareas

**HU 2.1: Registro de usuarios**

| Backlog ID | Historia de Usuario | Responsable | Tiempo Estimado (horas) | Fecha de Inicio | Fecha de Finalización |
| --- | --- | --- | --- | --- | --- |
| 2.1.1 | Diseñar formulario de registro | Brandon | 5 | 01/10/2025 | 08/10/2025 |
| 2.1.2 | Implementar validaciones de campos | Brandon | 6 | 01/10/2025 | 08/10/2025 |
| 2.1.3 | Conectar formulario con base de datos | Edison | 8 | 01/10/2025 | 08/10/2025 |
| 2.1.4 | Pruebas de creación de usuario | Bill | 4 | 01/10/2025 | 08/10/2025 |

**HU 2.2: Inicio de sesión de usuarios**

| Backlog ID | Historia de Usuario | Responsable | Tiempo Estimado (horas) | Fecha de Inicio | Fecha de Finalización |
| --- | --- | --- | --- | --- | --- |
| 2.2.1 | Diseñar formulario de inicio de sesión | Brandon | 4 | 01/10/2025 | 08/10/2025 |
| 2.2.2 | Implementar validación de credenciales | Edison | 7 | 01/10/2025 | 08/10/2025 |
| 2.2.3 | Manejo de errores de autenticación | Edison | 4 | 01/10/2025 | 08/10/2025 |
| 2.2.4 | Pruebas de acceso correcto e incorrecto | Bill | 5 | 01/10/2025 | 08/10/2025 |

**HU 2.3: Gestión de roles y permisos**

| Backlog ID | Historia de Usuario | Responsable | Tiempo Estimado (horas) | Fecha de Inicio | Fecha de Finalización |
| --- | --- | --- | --- | --- | --- |
| 2.3.1 | Definir estructura de roles en la BD | Bill | 4 | 01/10/2025 | 08/10/2025 |
| 2.3.2 | Implementar asignación de roles en el backend | Edison | 7 | 01/10/2025 | 08/10/2025 |
| 2.3.3 | Crear interfaz de administración de roles | Brandon | 6 | 01/10/2025 | 08/10/2025 |
| 2.3.4 | Pruebas de accesos según permisos | Bill | 5 | 01/10/2025 | 08/10/2025 |

# 4. Evidencias de Código

## HU 2.1: Registro de usuarios

Archivo: TravelDesk\frontend\src\components\RegistroForm.jsx

import { useMemo, useState } from 'react';

import '../styles/registro.css';

import { passwordLevel, validateFields } from '../utils/validators';

export default function RegistroForm() {

const [values, setValues] = useState({

nombre: '',

apellido: '',

email: '',

usuario: '',

contrasena: '',

confirmar: '',

fecha\_nac: '',

genero: '',

terminos: false,

});

const [errors, setErrors] = useState({});

const [ok, setOk] = useState(false);

const level = useMemo(() => passwordLevel(values.contrasena || ''), [values.contrasena]);

function onChange(*e*) {

const { name, value, type, checked } = e.target;

setValues((*v*) => ({ ...v, [name]: type === 'checkbox' ? checked : value }));

}

function validateField(*name*) {

const fieldErrors = validateFields(values);

// Solo mantener el error del campo tocado

setErrors((*prev*) => ({ ...prev, [name]: fieldErrors[name] }));

}

async function handleSubmit(*e*) {

e.preventDefault();

const fieldErrors = validateFields(values);

setErrors(fieldErrors);

const hasErrors = Object.values(fieldErrors).some(Boolean);

setOk(!hasErrors);

if (!hasErrors) {

try {

console.log("📤 Enviando datos al backend...", values);

const res = await fetch("http://localhost:3000/api/registro", {

method: "POST",

headers: { "Content-Type": "application/json" },

body: JSON.stringify(values),

});

const data = await res.json();

console.log("📥 Respuesta del servidor:", data);

if (data.ok) {

setOk(true);

alert("✅ Registro exitoso");

// Limpia formulario

setValues({

nombre: "",

apellido: "",

email: "",

usuario: "",

contrasena: "",

confirmar: "",

fecha\_nac: "",

genero: "",

terminos: false,

});

} else {

setErrors((*prev*) => ({

...prev,

general: data.error || "Error en el registro",

}));

}

} catch (err) {

console.error("❌ Error en conexión con el servidor:", err);

setErrors((*prev*) => ({

...prev,

general: "Error de conexión con el servidor",

}));

}

} else {

console.warn("⚠️ Hay errores de validación:", fieldErrors);

}

}

return (

<form *className*="container-reg" *onSubmit*={handleSubmit} *noValidate*>

<h1>Registro de Usuario</h1>

<div *className*="form-group">

<label *htmlFor*="nombre">Nombre</label>

<input *id*="nombre" *name*="nombre" *type*="text" *value*={values.nombre} *onChange*={onChange}

*onBlur*={() => validateField('nombre')} *placeholder*="Ej: Juan" />

<small *className*="error">{errors.nombre}</small>

</div>

<div *className*="form-group">

<label *htmlFor*="apellido">Apellido</label>

<input *id*="apellido" *name*="apellido" *type*="text" *value*={values.apellido} *onChange*={onChange}

*onBlur*={() => validateField('apellido')} *placeholder*="Ej: Pérez" />

<small *className*="error">{errors.apellido}</small>

</div>

<div *className*="form-group full">

<label *htmlFor*="email">Correo electrónico</label>

<input *id*="email" *name*="email" *type*="email" *value*={values.email} *onChange*={onChange}

*onBlur*={() => validateField('email')} *placeholder*="ejemplo@correo.com" />

<small *className*="error">{errors.email}</small>

</div>

<div *className*="form-group">

<label *htmlFor*="usuario">Nombre de usuario</label>

<input *id*="usuario" *name*="usuario" *type*="text" *value*={values.usuario} *onChange*={onChange}

*onBlur*={() => validateField('usuario')} *placeholder*="Solo letras, números y \_" />

<small *className*="hint">4-20 caracteres, sin espacios. Permitido: letras, números y guion bajo.</small>

<small *className*="error">{errors.usuario}</small>

</div>

<div *className*="form-group">

<label *htmlFor*="contrasena">Contraseña</label>

<input *id*="contrasena" *name*="contrasena" *type*="password" *value*={values.contrasena} *onChange*={onChange}

*onBlur*={() => validateField('contrasena')} *placeholder*="Mín. 8 caracteres" />

<div *className*="password-strength" *data-level*={level || undefined} *aria-hidden*="true" />

<small *className*="hint">Debe incluir mayúscula, minúscula, número y símbolo.</small>

<small *className*="error">{errors.contrasena}</small>

</div>

<div *className*="form-group">

<label *htmlFor*="confirmar">Confirmar contraseña</label>

<input *id*="confirmar" *name*="confirmar" *type*="password" *value*={values.confirmar} *onChange*={onChange}

*onBlur*={() => validateField('confirmar')} *placeholder*="Repite la contraseña" />

<small *className*="error">{errors.confirmar}</small>

</div>

<div *className*="form-group">

<label *htmlFor*="fecha\_nac">Fecha de nacimiento</label>

<input *id*="fecha\_nac" *name*="fecha\_nac" *type*="date" *value*={values.fecha\_nac} *onChange*={onChange}

*onBlur*={() => validateField('fecha')} />

<small *className*="hint">Debes tener al menos 13 años.</small>

<small *className*="error">{errors.fecha}</small>

</div>

<div *className*="form-group">

<label *htmlFor*="genero">Género</label>

<select *id*="genero" *name*="genero" *value*={values.genero} *onChange*={onChange} *onBlur*={() => validateField('genero')}>

<option *value*="">Selecciona...</option>

<option>Masculino</option>

<option>Femenino</option>

<option>Otro</option>

<option>Prefiero no decirlo</option>

</select>

<small *className*="error">{errors.genero}</small>

</div>

<div *className*="form-group inline">

<input *id*="terminos" *name*="terminos" *type*="checkbox" *checked*={values.terminos} *onChange*={onChange}

*onBlur*={() => validateField('terminos')} />

<label *htmlFor*="terminos">Acepto los Términos y Condiciones</label>

</div>

<small *className*="error">{errors.terminos}</small>

<button *type*="submit" *className*="btn">Registrarse</button>

{ok && <p *className*="success">Formulario válido. Listo para enviar.</p>}

</form>

);

}

Archivo: TravelDesk/frontend/src/pages/Registro.jsx

import RegistroForm from '../components/RegistroForm.jsx';

export default function Registro() {

return (

<main>

<RegistroForm />

</main>

);

}

Archivo: TravelDesk/frontend/src/styles/registro.css

:root{

--primary:#f97316; /\* orange \*/

--danger:#dc2626; /\* red for errors \*/

--ok:#f97316; /\* success in theme color \*/

--neutral:#6b7280; /\* subtle gray for hints \*/

--bg:#ffffff; /\* white background \*/

--card:#ffffff; /\* white cards \*/

--border:#ffe4d1; /\* light orange border \*/

}

.container-reg{

max-width:980px; /\* desktop width \*/

margin:40px auto;

background:var(--card);

padding:32px 28px;

border:1px solid var(--border);

border-radius:12px;

box-shadow:0 6px 20px rgba(2,8,23,.06);

display:grid;

grid-template-columns: repeat(2, minmax(0, 1fr));

column-gap:24px;

row-gap:16px;

}

.container-reg h1{

margin:0 0 12px;

font-size:28px;

color:var(--primary);

grid-column: 1 / -1; /\* span both columns \*/

text-align:left;

}

.form-group{ display:flex; flex-direction:column; gap:8px }

.form-group.inline{flex-direction:row; align-items:center; gap:10px}

.form-group.full{ grid-column: 1 / -1; }

/\* Make each group be a grid item \*/

.container-reg > .form-group, .container-reg > .form-group.full, .container-reg > .form-group.inline { width: 100%; }

label{font-weight:700; color: var(--primary)}

input,select{

padding:12px 10px;

border:1px solid var(--border);

border-radius:8px;

font-size:15px;

outline:none;

transition:border-color .2s, box-shadow .2s;

}

input:focus,select:focus{

border-color:var(--primary);

box-shadow:0 0 0 3px rgba(249,113,22,.2);

}

.hint{color:var(--neutral); font-size:12px}

.error{color:var(--danger); font-size:12px; min-height:16px}

.success{color:var(--ok); margin-top:10px}

.btn{

margin-top:4px;

background:var(--primary);

color:#fff;

border:none;

padding:12px 14px;

border-radius:8px;

cursor:pointer;

font-weight:600;

}

.btn:hover{filter:brightness(1.05)}

/\* Actions span both columns and align left \*/

.btn, .success { grid-column: 1 / -1; justify-self: start; }

.password-strength{

height:8px;

border-radius:6px;

background:linear-gradient(90deg,#e2e8f0 0 100%);

transition:background .3s;

}

.password-strength[data-level="1"]{background:linear-gradient(90deg,#fdba74 0 25%, #e5e7eb 25% 100%)}

.password-strength[data-level="2"]{background:linear-gradient(90deg,#fb923c 0 50%, #e5e7eb 50% 100%)}

.password-strength[data-level="3"]{background:linear-gradient(90deg,#f97316 0 75%, #e5e7eb 75% 100%)}

.password-strength[data-level="4"]{background:linear-gradient(90deg,#ea580c 0 100%)}

/\* Responsive: stack to one column on small screens \*/

@media (max-width: 768px) {

.container-reg{

margin:20px 12px;

padding:24px 18px;

grid-template-columns: 1fr;

row-gap:14px;

}

}

Archivo: TravelDesk/frontend/src/utils/validators.js

// Utilidades de validación para el formulario de registro

export const isOnlyLetters = (s) => /^[A-Za-zÁÉÍÓÚÜÑáéíóúüñ ]{2,}$/.test(s.trim());

export const isUsername = (s) => /^[A-Za-z0-9\_]{4,20}$/.test(s.trim());

export const hasUpper = (s) => /[A-ZÁÉÍÓÚÜÑ]/.test(s);

export const hasLower = (s) => /[a-záéíóúüñ]/.test(s);

export const hasNumber = (s) => /\d/.test(s);

export const hasSymbol = (s) => /[^A-Za-z0-9]/.test(s);

export function passwordLevel(s) {

let level = 0;

if (s.length >= 8) level++;

if (hasUpper(s) && hasLower(s)) level++;

if (hasNumber(s)) level++;

if (hasSymbol(s)) level++;

return Math.min(level, 4);

}

export function validateFields(values) {

const errors = {};

// Nombre

if (!values.nombre?.trim()) errors.nombre = 'Este campo es obligatorio.';

else if (values.nombre.trim().length < 2) errors.nombre = 'Mínimo 2 caracteres.';

else if (!isOnlyLetters(values.nombre)) errors.nombre = 'Solo letras y espacios.';

// Apellido

if (!values.apellido?.trim()) errors.apellido = 'Este campo es obligatorio.';

else if (values.apellido.trim().length < 2) errors.apellido = 'Mínimo 2 caracteres.';

else if (!isOnlyLetters(values.apellido)) errors.apellido = 'Solo letras y espacios.';

// Email

if (!values.email?.trim()) errors.email = 'El email es obligatorio.';

else {

// Usamos la validación del browser normalmente; aquí un check básico

const emailOk = /.+@.+\..+/.test(values.email);

if (!emailOk) errors.email = 'Formato de correo inválido.';

}

// Usuario

if (!values.usuario?.trim()) errors.usuario = 'El usuario es obligatorio.';

else if (!isUsername(values.usuario)) errors.usuario = '4-20 caracteres. Usa letras, números o \_';

// Contraseña

const v = values.contrasena || '';

if (!v) errors.contrasena = 'La contraseña es obligatoria.';

else if (v.length < 8) errors.contrasena = 'Mínimo 8 caracteres.';

else if (!(hasUpper(v) && hasLower(v))) errors.contrasena = 'Incluye mayúscula y minúscula.';

else if (!hasNumber(v)) errors.contrasena = 'Incluye al menos un número.';

else if (!hasSymbol(v)) errors.contrasena = 'Incluye al menos un símbolo.';

// Confirmación

if (!values.confirmar) errors.confirmar = 'Confirma la contraseña.';

else if (values.confirmar !== v) errors.confirmar = 'Las contraseñas no coinciden.';

// Fecha de nacimiento

if (!values.fecha\_nac) errors.fecha = 'La fecha es obligatoria.';

else {

const hoy = new Date();

const f = new Date(values.fecha\_nac + 'T00:00:00');

const anos = hoy.getFullYear() - f.getFullYear() - ((hoy.getMonth() < f.getMonth() || (hoy.getMonth() === f.getMonth() && hoy.getDate() < f.getDate())) ? 1 : 0);

if (isNaN(f.getTime())) errors.fecha = 'Fecha inválida.';

else if (anos < 13) errors.fecha = 'Debes tener al menos 13 años.';

}

// Términos

if (!values.terminos) errors.terminos = 'Debes aceptar los términos.';

// Género (opcional); si quieres obligatorio, descomenta:

// if (!values.genero) errors.genero = 'Selecciona una opción.';

return errors;

}

Archivo: TravelDesk\backend\scripts\backup\backup-verify-restore.ps1

$ErrorActionPreference = "Stop"

. "$PSScriptRoot\\backup.config.ps1"

# Verificación simple de restauración del último respaldo completo (.sql)

$mysql = Join-Path $MYSQL\_BIN "mysql.exe"

if (!(Test-Path $mysql)) { throw "No se encontró mysql.exe en '$mysql'" }

$fullDir = Join-Path $BACKUP\_DIR "full"

$ultimo = Get-ChildItem $fullDir -Filter "\*.sql" | Sort-Object LastWriteTime -Descending | Select-Object -First 1

if (-not $ultimo) { throw "No se encontró ningún archivo .sql en '$fullDir'" }

$dbPrueba = "${DB\_NAME}\_prueba\_restore"

& $mysql -h $DB\_HOST -P $DB\_PORT -u $DB\_USER -p$DB\_PASSWORD -e "DROP DATABASE IF EXISTS ``$dbPrueba``; CREATE DATABASE ``$dbPrueba`` CHARACTER SET utf8mb4;" | Out-Null

# Importar el dump

# PowerShell no implementa el operador de redirección de entrada '<'. Usamos cmd.exe para hacer la redirección de forma segura.

$cmd = '"{0}" -h {1} -P {2} -u {3} -p{4} {5} < "{6}"' -f $mysql, $DB\_HOST, $DB\_PORT, $DB\_USER, $DB\_PASSWORD, $dbPrueba, $ultimo.FullName

cmd.exe /c $cmd

Write-Host "Restauración de prueba completada en: $dbPrueba (desde $($ultimo.Name))"

## HU 2.1.3 — Conectar formulario con base de datos

Archivo: ‎TravelDesk/backend/server.js

const app = express();

const PORT = process.env.PORT || 3000;

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

const pool = require("./src/config/db.js").default;

const cors = require('cors');

app.use(cors({

origin: 'http://localhost:5173', // Solo permite tu frontend

credentials: true

}));

app.use(bodyParser.json());

// Ruta de prueba

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

res.send("🚀 Backend TravelDesk funcionando!");

});

app.post("/api/registro", async (req, res) => {

const { nombre, apellido, email, usuario, contrasena, fecha\_nac, genero } = req.body;

if (!nombre || !apellido || !email || !usuario || !contrasena || !fecha\_nac) {

return res.status(400).json({ error: "Faltan campos obligatorios" });

}

try {

// Aquí deberías hashear la contraseña antes de guardar (usa bcrypt en producción)

await pool.query(

"INSERT INTO usuarios (nombre\_usuario, email, password) VALUES (?, ?, ?)",

[usuario, email, contrasena]

);

// Puedes insertar en otras tablas según tu modelo (turistas, etc.)

res.json({ ok: true });

} catch (err) {

res.status(500).json({ error: "Error al registrar usuario" });

}

});

app.listen(PORT, () => {

console.log(`✅ Servidor corriendo en http://localhost:${PORT}`);

});

Archivo: ‎TravelDesk/frontend/src/components/RegistroForm.jsx

if (data.ok) { setOk(true); // Opcional: limpiar formulario o mostrar mensaje de éxito } else { setErrors((prev) => ({ ...prev, general: data.error || "Error en el registro" })); } } catch (err) { setErrors((prev) => ({ ...prev, general: "Error de conexión con el servidor" })); } } } return (

**HU 2.2: Inicio de sesión de usuarios**

**HU-2.2.1 Y HU-2.3.3 DISEÑOS IMPLEMENTADOS**

Archivo: ‎TravelDesk/backend/package.json

"license": "ISC",

"packageManager": "pnpm@10.17.0",

"dependencies": {

"body-parser": "^2.2.0",

"cors": "^2.8.5",

"dotenv": "^17.2.2",

"express": "^5.1.0",

"mysql2": "^3.15.0"

Archivo: ‎TravelDesk/frontend/src/components/RegistroForm.jsx

setErrors((prev) => ({ ...prev, [name]: fieldErrors[name] }));

}

function handleSubmit(e) {

async function handleSubmit(e) {

e.preventDefault();

const fieldErrors = validateFields(values);

setErrors(fieldErrors);

const hasErrors = Object.values(fieldErrors).some(Boolean);

setOk(!hasErrors);

if (!hasErrors) {

try {

const res = await fetch("http://localhost:3000/api/registro", {

method: "POST",

headers: { "Content-Type": "application/json" },

body: JSON.stringify(values),

});

const data = await res.json();

if (data.ok) {

setOk(true);

// Opcional: limpiar formulario o mostrar mensaje de éxito

} else {

setErrors((prev) => ({ ...prev, general: data.error || "Error en el registro" }));

}

} catch (err) {

setErrors((prev) => ({ ...prev, general: "Error de conexión con el servidor" }));

}

}

}

return (

## 

## S3-HU2.2.2 Implementar validación de credenciales

Archivo: TravelDesk/backend/server.js

// Endpoint de login

app.post("/api/login", async (req, res) => {

const { usuario, contrasena } = req.body;

if (!usuario || !contrasena) {

return res.status(400).json({ error: "Usuario y contraseña requeridos." });

}

try {

const [rows] = await pool.query(

"SELECT \* FROM usuarios WHERE nombre\_usuario = ?",

[usuario]

);

if (rows.length === 0) {

return res.status(401).json({ error: "Credenciales incorrectas." });

}

const user = rows[0];

// Comparación simple, en producción usa bcrypt

if (user.password !== contrasena) {

return res.status(401).json({ error: "Credenciales incorrectas." });

}

res.json({ ok: true, usuario: user.nombre\_usuario });

} catch (err) {

res.status(500).json({ error: "Error en el servidor." });

}

});

Archivo: TravelDesk/frontend/src/components/LoginForm.jsx

-- Itinerarios con sus grupos y estado

SELECT i.id\_itinerario, g.nombre AS grupo, e.nombre\_estado, i.fecha\_inicio, i.fecha\_fin

FROM itinerarios i

JOIN grupos g ON i.id\_grupo = g.id\_grupo

JOIN estados\_presupuesto e ON i.estado\_presupuesto\_id = e.id\_estado;

-- Programas por itinerario

SELECT ip.id\_itinerario\_programa, i.id\_itinerario, p.nombre, ip.fecha, ip.hora\_inicio, ip.hora\_fin

FROM itinerario\_programas ip

JOIN programas p ON ip.id\_programa = p.id\_programa

JOIN itinerarios i ON ip.id\_itinerario = i.id\_itinerario;

-- Turistas en cada itinerario

SELECT it.id\_itinerario, t.nombre, t.apellido, t.nacionalidad

FROM itinerario\_turistas it

JOIN turistas t ON it.id\_turista = t.id\_turista;

-- Transporte asignado

SELECT dti.id\_detalle\_transporte, ip.id\_itinerario\_programa, tr.empresa, tr.tipo, dti.horario\_recojo, dti.lugar\_recojo

FROM detalle\_transporte\_itinerario dti

JOIN transportes tr ON dti.id\_transporte = tr.id\_transporte

JOIN itinerario\_programas ip ON dti.id\_itinerario\_programa = ip.id\_itinerario\_programa;

-- Detalle Machu Picchu

SELECT dmi.id\_itinerario\_programa, dmi.empresa\_tren, dmi.horario\_tren\_ida, dmi.horario\_tren\_retor, dmi.nombre\_guia

FROM detalle\_machu\_itinerario dmi;

Archivo: rendimientoinicial.sql

EXPLAIN SELECT i.id\_itinerario, g.nombre, e.nombre\_estado

FROM itinerarios i

JOIN grupos g ON i.id\_grupo = g.id\_grupo

JOIN estados\_presupuesto e ON i.estado\_presupuesto\_id = e.id\_estado;

**HU 2.3: Gestión de roles y permisos**

S3-HU2.3.1 Definir estructura de roles en la BD

https://www.phpmyadmin.net/

--

-- Servidor: 127.0.0.1

-- Tiempo de generación: 22-09-2025 a las 16:58:11

-- Tiempo de generación: 13-10-2025 a las 08:03:52

-- Versión del servidor: 10.4.32-MariaDB

-- Versión de PHP: 8.2.12

@@ -179,6 +179,15 @@ CREATE TABLE `usuarios` (

`updated\_at` datetime DEFAULT current\_timestamp() ON UPDATE current\_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

--

-- Volcado de datos para la tabla `usuarios`

--

INSERT INTO `usuarios` (`id\_usuario`, `nombre\_usuario`, `email`, `password`, `rol`, `created\_at`, `updated\_at`) VALUES

(1, 'Edison', 'edi@gmail.com', 'Edi123-', 'admin', '2025-10-13 00:59:14', '2025-10-13 00:59:14'),

(2, 'Brandon', 'bran@gmail.com', 'Bran123-', 'agente', '2025-10-13 00:59:14', '2025-10-13 00:59:14'),

(3, 'Shank', 'shank@one.piece', 'Shank123-', 'cliente', '2025-10-09 21:59:53', '2025-10-09 21:59:53');

--

-- Índices para tablas volcadas

--

@@ -315,7 +324,7 @@ ALTER TABLE `turistas`

-- AUTO\_INCREMENT de la tabla `usuarios`

--

ALTER TABLE `usuarios`

MODIFY `id\_usuario` int(11) NOT NULL AUTO\_INCREMENT;

MODIFY `id\_usuario` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=6;

S3-HU2.3.1 Definir estructura de roles en la BD

Archivo: TravelDesk/backend/[server.js](http://server.js)

// backend/server.js

import express from "express";

import cors from "cors";

import bodyParser from "body-parser";

import express from 'express';

import cors from 'cors';

import bodyParser from 'body-parser';

import jwt from 'jsonwebtoken';

import pool, { testConnection } from "./src/config/db.js";

const SECRET = 'tu\_clave\_secreta'; // Usa una variable de entorno en producción

const app = express();

const PORT = process.env.PORT || 3000;

@@ -58,17 +60,80 @@ app.post("/api/login", async (req, res) => {

return res.status(401).json({ error: "Credenciales incorrectas." });

}

const user = rows[0];

// Comparación simple, en producción usa bcrypt

if (user.password !== contrasena) {

return res.status(401).json({ error: "Credenciales incorrectas." });

}

res.json({ ok: true, usuario: user.nombre\_usuario });

// Genera el token con el rol

const token = jwt.sign(

{ id: user.id\_usuario, usuario: user.nombre\_usuario, role: user.rol.toUpperCase() },

SECRET,

{ expiresIn: '2h' }

);

res.json({ ok: true, usuario: user.nombre\_usuario, role: user.rol.toUpperCase(), token });

} catch (err) {

res.status(500).json({ error: "Error en el servidor." });

}

});

// Endpoint para actualizar el rol de un usuario

app.put("/api/usuarios/:id/rol", requireRole(["ADMIN"]), async (req, res) => {

const { id } = req.params;

const { role } = req.body;

if (!role) {

return res.status(400).json({ error: "El rol es obligatorio." });

}

try {

const [result] = await pool.query(

"UPDATE usuarios SET rol = ? WHERE id\_usuario = ?",

[role.toLowerCase(), id]

);

if (result.affectedRows === 0) {

return res.status(404).json({ error: "Usuario no encontrado." });

}

res.json({ ok: true });

} catch (err) {

console.error("❌ Error al actualizar rol:", err.message);

res.status(500).json({ error: "Error al actualizar el rol." });

}

});

// Endpoint para obtener la lista de usuarios y sus roles

app.get("/api/usuarios", async (req, res) => {

try {

const [rows] = await pool.query(

"SELECT id\_usuario AS id, nombre\_usuario AS name, email, rol FROM usuarios"

);

// Normaliza el rol a mayúsculas

const users = rows.map(u => ({

...u,

role: u.rol ? u.rol.toUpperCase() : "CLIENTE"

}));

res.json(users);

} catch (err) {

console.error("❌ Error al obtener usuarios:", err.message);

res.status(500).json({ error: "Error al obtener usuarios." });

}

});

// Iniciar servidor

app.listen(PORT, () => {

console.log(`✅ Servidor corriendo en http://localhost:${PORT}`);

});

function requireRole(roles) {

return (req, res, next) => {

const auth = req.headers.authorization;

if (!auth) return res.status(401).json({ error: "Token requerido" });

const token = auth.split(' ')[1];

try {

const decoded = jwt.verify(token, SECRET);

req.user = decoded;

if (!roles.includes(decoded.role)) {

return res.status(403).json({ error: "No autorizado" });

}

next();

} catch {

return res.status(401).json({ error: "Token inválido" });

}

};

}

Archivo: TravelDesk/frontend/src/components/LoginForm.jsx

const [error, setError] = useState(""); const [loading, setLoading] = useState(false); const user = JSON.parse(localStorage.getItem("user")); async function handleSubmit(e) { e.preventDefault(); setError("");@@ -17,17 +19,24 @@ export default function LoginForm() { body: JSON.stringify({ usuario, contrasena }), }); const data = await res.json(); if (res.ok && data.ok) { alert("¡Bienvenido " + data.usuario + "!"); // Aquí puedes redirigir o guardar el usuario en contexto if (data.ok) { localStorage.setItem( "user", JSON.stringify({ name: data.usuario, role: data.role, token: data.token }) ); window.location.reload(); // Recarga para actualizar el estado } else { setError(data.error || "Error desconocido. Intenta de nuevo."); setError(data.error || "Credenciales incorrectas"); } } catch { setError("Error de conexión con el servidor."); } finally { setLoading(false); setError("Error de conexión con el servidor"); } setLoading(false); } function logout() { localStorage.removeItem("user"); window.location.reload(); } return (@@ -40,6 +49,7 @@ export default function LoginForm() { value={usuario} onChange={(e) => setUsuario(e.target.value)} required disabled={!!user} /> </div> <div>@@ -49,12 +59,18 @@ export default function LoginForm() { value={contrasena} onChange={(e) => setContrasena(e.target.value)} required disabled={!!user} /> </div> {error && <div style={{ color: "red", marginTop: 8 }}>{error}</div>} <button type="submit" disabled={loading}> <button type="submit" disabled={loading || !!user}> {loading ? "Verificando..." : "Entrar"} </button> {user && ( <button type="button" onClick={logout} style={{ marginLeft: 12 }}> Cerrar sesión </button> )} </form> );}

Collapse file: TravelDesk/frontend/src/components/UsersRolesTable.jsx‎TravelDesk/frontend/src/components/UsersRolesTable.jsx‎Copy file name to clipboardExpand all lines: TravelDesk/frontend/src/components/UsersRolesTable.jsx+28-13Lines changed: 28 additions & 13 deletionsOriginal file line numberDiff line numberDiff line change@@ -2,15 +2,35 @@ import { useMemo, useState } from 'react';import { ROLES } from '../utils/rbac.js';import '../styles/registro.css';// Diseño-only: tabla de usuarios con selector de rol (sin conexión a backend)export default function UsersRolesTable({ initialUsers }) { const [users, setUsers] = useState(initialUsers || []); const roles = useMemo(() => Object.values(ROLES), []); const [filter, setFilter] = useState(''); const [roleFilter, setRoleFilter] = useState(''); const [saving, setSaving] = useState(false); function onChangeRole(id, role) { setUsers((list) => list.map((u) => (u.id === id ? { ...u, role } : u))); async function onChangeRole(id, role) { setSaving(true); const user = JSON.parse(localStorage.getItem("user")); try { const res = await fetch(`http://localhost:3000/api/usuarios/${id}/rol`, { method: "PUT", headers: { "Content-Type": "application/json", "Authorization": `Bearer ${user.token}` }, body: JSON.stringify({ role }), }); const data = await res.json(); if (res.ok && data.ok) { setUsers(list => list.map(u => (u.id === id ? { ...u, role } : u))); } else { alert(data.error || "Error al actualizar el rol"); } } catch { alert("Error de conexión con el servidor"); } setSaving(false); } const filtered = users.filter((u) => {@@ -54,7 +74,11 @@ export default function UsersRolesTable({ initialUsers }) { <td style={td}>{u.name}</td> <td style={td}>{u.email}</td> <td style={td}> <select value={u.role} onChange={(e) => onChangeRole(u.id, e.target.value)}> <select value={u.role} onChange={e => onChangeRole(u.id, e.target.value)} disabled={saving} > {roles.map((r) => ( <option key={r} value={r}>{r}</option> ))}@@ -65,15 +89,6 @@ export default function UsersRolesTable({ initialUsers }) { ))} </tbody> </table> <div style={{ display: 'flex', gap: 8, marginTop: 12 }}> <button className="btn" type="button" onClick={() => { /\* diseño-only \*/ }}> Guardar cambios </button> <button className="btn" type="button" onClick={() => setUsers(initialUsers || [])}> Deshacer cambios </button> </div> </div> );}

Archivo: TravelDesk/frontend/src/pages/RolesAdmin.jsx

import { useEffect, useState } from 'react';

import UsersRolesTable from '../components/UsersRolesTable.jsx';

import { ROLES } from '../utils/rbac.js';

const mockUsers = [

{ id: 1, name: 'Ana López', email: 'ana@turismo.com', role: ROLES.ADMIN, active: true },

{ id: 2, name: 'Carlos Pérez', email: 'carlos@turismo.com', role: ROLES.SUBADMIN, active: true },

{ id: 3, name: 'Lucía Gómez', email: 'lucia@cliente.com', role: ROLES.TURISTA, active: true },

{ id: 4, name: 'Miguel Ruiz', email: 'miguel@cliente.com', role: ROLES.TURISTA, active: false },

];

export default function RolesAdmin() {

const [users, setUsers] = useState([]);

const [loading, setLoading] = useState(true);

useEffect(() => {

fetch('http://localhost:3000/api/usuarios')

.then(res => res.json())

.then(data => {

setUsers(data);

setLoading(false);

})

.catch(() => setLoading(false));

}, []);

if (loading) return <main>Cargando usuarios...</main>;

return (

<main>

<UsersRolesTable initialUsers={mockUsers} />

<UsersRolesTable initialUsers={users} />

</main>

);

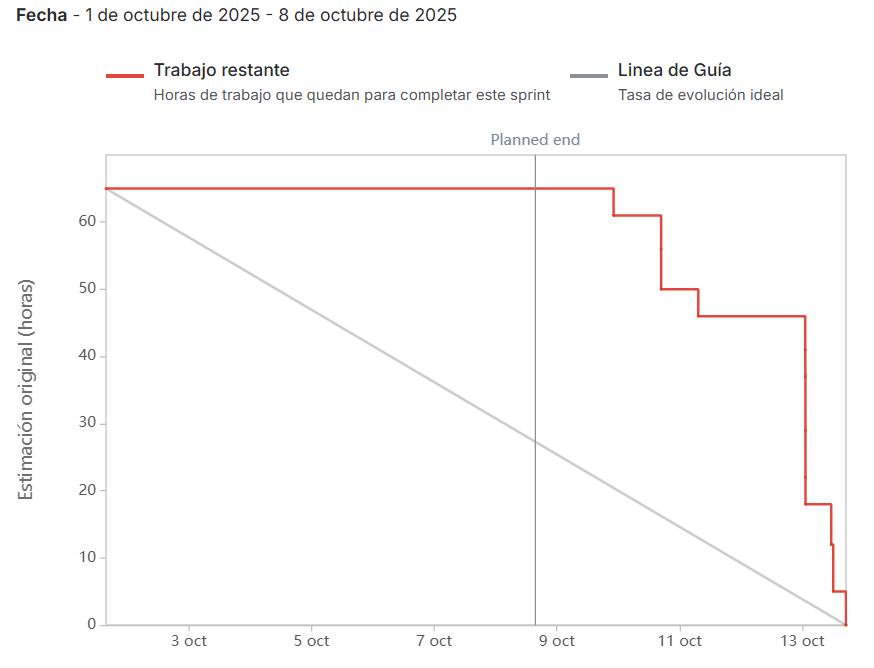
}

Archivo: TravelDesk/frontend/src/utils/rbac.js

export const ROLES = { ADMIN: 'ADMIN', SUBADMIN: 'SUBADMIN', TURISTA: 'TURISTA', AGENTE: 'AGENTE', CLIENTE: 'CLIENTE',};export const PERMISSIONS = {@@ -25,14 +25,14 @@ export const ROLE\_PERMISSIONS = { PERMISSIONS.VIEW\_REPORTS, PERMISSIONS.MANAGE\_CONTENT, ], [ROLES.SUBADMIN]: [ [ROLES.AGENTE]: [ PERMISSIONS.MANAGE\_PACKAGES, PERMISSIONS.VIEW\_BOOKINGS, PERMISSIONS.CREATE\_BOOKING, PERMISSIONS.CANCEL\_BOOKING, PERMISSIONS.MANAGE\_CONTENT, ], [ROLES.TURISTA]: [ [ROLES.CLIENTE]: [ PERMISSIONS.VIEW\_BOOKINGS, PERMISSIONS.CREATE\_BOOKING, PERMISSIONS.CANCEL\_BOOKING,

# 5. Burndown Chart

| **Backlog ID** | **Historia de Usuario** | **Tiempo Estimado** | **1/10/2025** | **2/10/2025** | **3/10/2025** | **4/10/2025** | **5/10/2025** | **6/10/2025** | **7/10/2025** | **8/10/2025** | **9/10/2025** | **10/10/2025** | **11/10/2025** | **12/10/2025** | **13/10/2025** | **14/10/2025** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Dia1** | **Dia2** | **Dia3** | **Dia4** | **Dia5** | **Dia6** | **Dia7** | **Dia8** | **Dia9** | **Dia10** | **Dia11** | **Dia12** | **Dia13** | **Dia14** |
| **HU-2.1** | **Registro de usuarios** | 23 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 6 | 8 | 4 | 0 | 0 |
| **HU-2.2** | **Inicio de sesión de usuarios** | 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 7 | 9 | 0 |
| **HU-19.3** | **Gestión de roles y permisos** | 22 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 11 | 11 | 0 |



# 6. Análisis del Sprint

* Se evidenció un avance irregular a lo largo del periodo planificado. En los primeros días no se registraron progresos significativos, lo que indica que el equipo estuvo concentrado en tareas de preparación del entorno, revisión de requisitos y coordinación técnica. Esta fase inicial, aunque necesaria, retrasó el inicio efectivo de las actividades de desarrollo.
* Hubo problemas de coordinación con respecto a la organización de ventanas y accesos, que tuvieron que coordinarse directamente.
* Pese al esfuerzo sostenido durante la segunda mitad del sprint, el proyecto presentó un retraso aproximado de una semana en su culminación, debido principalmente a la acumulación de tareas en los últimos días, la subestimación de tiempos en ciertas actividades. Este retraso afectó el cumplimiento del cronograma previsto, aunque finalmente se logró completar las tres historias de usuario con éxito.
* Se asignaron permisos según los roles, protegiendo las rutas desde el backend y fortaleciendo la seguridad que tendrán los usuarios al usar el sistema.
* Ya se pueden realizar la inserción de usuarios a la base de datos

# 7. Conclusiones

El Sprint 3 culminó con la entrega completa de las tres historias de usuario planificadas, a pesar del retraso de una semana respecto al cronograma original. Este desfase se debió principalmente a la concentración de tareas en la fase final del sprint, derivada de una planificación inicial ajustada y de la necesidad de coordinar dependencias técnicas entre los módulos de registro, inicio de sesión y gestión de roles. No obstante, el equipo mostró una buena capacidad de adaptación para reorganizar el flujo de trabajo y cumplir los objetivos propuestos.

El análisis del Burndown Chart evidencia que, si bien el progreso inicial fue limitado, el equipo logró recuperar el ritmo en la segunda mitad del sprint, manteniendo un trabajo intensivo que permitió alcanzar la finalización de todas las funcionalidades.

Con respecto a las coordinaciones se recomienda realizar mas a menudo las actas de reuniones para evitar posibles conflictos en la construcción del software.

Con respecto a la las pruebas piloto se completaron y a su vez los resultados confirmaron la integridad de los datos. En comparación con el anterior sprint, se observó una mejor continuidad en la ejecución de las tareas.