



## Documento de Verificación

**Integrantes:**

Mathías Fernández  
Eduardo Flores  
Federico Acosta  
Gabriel Suárez  
Pablo Perdomo  
Alexander Rodriguez

**Tutor:**

Fernando Arrieta

<b>1. Introducción.....</b>	<b>3</b>
1.1 Propósito.....	3
<b>2. Plan de Testing.....</b>	<b>3</b>
<b>3. Categorías de Testing.....</b>	<b>3</b>
3.1 Pruebas unitarias.....	3
3.2 Pruebas E2E.....	4
3.3 Pruebas de interfaz.....	4
<b>4. Estrategia de verificación y trazabilidad.....</b>	<b>4</b>
4.1 Verificación unitaria por componente.....	4
4.2 Trazabilidad de requisitos.....	5
4.3 Cobertura de Pruebas.....	7
<b>5. Incidencias encontradas.....</b>	<b>7</b>
<b>6. Conclusiones.....</b>	<b>8</b>
<b>7. Anexos.....</b>	<b>8</b>
7.1 Cobertura tests unitarios de servicios.....	8
7.2 Cobertura tests unitarios de controladores.....	8
7.3 Cobertura de Requisitos.....	9

# 1. Introducción

## 1.1 Propósito

Este documento detalla el plan de verificación, ejecución de pruebas, tipos de pruebas utilizadas y resultados obtenidos durante el desarrollo del sistema online para la venta de pasajes de ómnibus de larga distancia. Su objetivo es asegurar que el software cumple con los requerimientos funcionales y no funcionales definidos en la etapa de análisis y diseño.

## 2. Plan de Testing

Se define un enfoque centrado principalmente en pruebas funcionales automatizadas y manuales aplicadas sobre los servicios y controladores del backend. Estas pruebas permiten validar tanto la lógica de negocio como la correcta respuesta de los endpoints REST. Se complementa con un conjunto reducido pero representativo de pruebas automatizadas de extremo a extremo (E2E) sobre el frontend web, utilizando Playwright, con foco en los flujos más críticos desde la perspectiva del usuario. Esta estrategia habilita una verificación efectiva del sistema completo sin necesidad de cubrir exhaustivamente el frontend, lo cual optimiza el mantenimiento sin comprometer la calidad.

## 3. Categorías de Testing

### 3.1 Pruebas unitarias

Las pruebas unitarias se enfocan en validar el comportamiento de métodos o clases individuales del backend, en particular aquellos que implementan la lógica de negocio en servicios desarrollados con Java y Spring Boot. Estas pruebas se ejecutan de forma aislada y sin involucrar capas externas del sistema, utilizando

simulaciones (mocks) de dependencias a través de frameworks como JUnit y Mockito. Permiten verificar la consistencia lógica de cada componente y detectar errores de forma rápida y localizada.

### 3.2 Pruebas E2E

Simulan la experiencia de un usuario real al interactuar con la aplicación web desde el navegador, reproduciendo de forma automatizada los pasos clave de los flujos más relevantes, como el proceso de compra de pasajes o la calificación de viajes. Estas pruebas permiten validar tanto la respuesta visual de la interfaz como la correcta interacción con los servicios backend. Además de verificar que los flujos completos funcionen de extremo a extremo, ayudan a identificar errores que solo se presentan en escenarios integrados. En este proyecto, su uso fue limitado a recorridos funcionales críticos, pero resultaron esenciales para comprobar la coherencia general del sistema desde la perspectiva del usuario.

### 3.3 Pruebas de interfaz

Se realizaron de forma exploratoria tanto sobre la interfaz web como en la aplicación móvil. En el entorno mobile, las pruebas se llevaron a cabo manualmente desde dispositivos físicos, ejecutando flujos funcionales clave como el inicio de sesión, la compra de pasajes y la consulta del historial. Estas pruebas permitieron detectar problemas visuales, errores de navegación y validar la experiencia general del usuario.

## 4. Estrategia de verificación y trazabilidad

Las matrices de verificación y trazabilidad presentadas a continuación permiten visualizar, por un lado, el tipo de prueba unitaria aplicada a cada componente técnico del sistema y, por otro, el tipo de verificación aplicado a cada requisito funcional. En concordancia con la estrategia adoptada, la mayoría de los requerimientos fueron validados mediante pruebas funcionales sobre el backend, ejecutadas sobre servicios y controladores utilizando herramientas como Mockito y JUnit. Los casos que implicaban lógica crítica desde la perspectiva del usuario final

fueron verificados mediante pruebas E2E automatizadas con Playwright en el frontend web. Además, ciertas funcionalidades específicas de la aplicación móvil fueron verificadas manualmente desde dispositivos físicos, enfocándose en recorridos esenciales como la autenticación y la compra de pasajes..

#### 4.1 Verificación unitaria por componente

Componente	Tipo	Cobertura Lineas	Resultado
AsientoService	Servicio	100%	OK ▾
CategoriaService	Servicio	100%	OK ▾
CompraPasajeService	Servicio	100%	OK ▾
EmailService	Servicio	100%	OK ▾
GenerarContraseniaService	Servicio	100%	OK ▾
JwtService	Servicio	100%	OK ▾
LocalidadService	Servicio	100%	OK ▾
OmnibusService	Servicio	100%	OK ▾
TokenService	Servicio	100%	OK ▾
TransactionalService	Servicio	100%	OK ▾
UsuarioService	Servicio	100%	OK ▾
ViajeService	Servicio	80%	OK ▾
AsientoController	Controlador	100%	OK ▾
CategoriaController	Controlador	100%	OK ▾
ComprasController	Controlador	100%	OK ▾
LocalidadController	Controlador	100%	OK ▾
OmnibusController	Controlador	100%	OK ▾
TokenController	Controlador	100%	OK ▾
UsuarioController	Controlador	100%	OK ▾
ViajeController	Controlador	100%	OK ▾

Ver anexo: [7.1](#) y [7.2](#)

## 4.2 Trazabilidad de requisitos

Requisito	Caso	Tipo	Descripción	Resultado
RF-01: Alta de Usuario	T-001	Test Unitario	Verifica lógica de creación y validación sin interfaz.	OK ▾
RF-02: Listado de Usuarios	T-002	E2E Playwright	Requiere interacción con filtros y visualización de datos.	OK ▾
RF-03: Estadísticas de Usuarios	T-003	E2E Playwright	Implica generación de vistas gráficas y descarga desde UI.	OK ▾
RF-04: Alta de Localidad	T-004	Test Unitario	Valida reglas de negocio simples en backend.	OK ▾
RF-05: Alta de Ómnibus	T-005	Test Unitario	Lógica directa de persistencia, sin interfaz dependiente.	OK ▾
RF-06: Reasignación de Viaje	T-006	E2E Playwright	Depende de selección visual y validaciones cruzadas.	OK ▾
RF-07: Cierre de Venta de Pasajes	T-007	Unitario	Lógica compleja en backend.	OK ▾
RF-08: Devolución de Pasaje	T-008	Unitario	Involucra validación contextual y confirmación en la interfaz.	OK ▾
RF-09: Listado de Pasajes Vendidos	T-009	E2E Playwright	Visualización de datos dependientes de filtros desde UI.	OK ▾
RF-10: Compra de Pasaje	T-010	E2E Playwright	Flujo completo validado desde búsqueda hasta confirmación y PDF.	OK ▾
RF-11: Historial de Compras	T-011	Funcional	Requiere filtrado por cliente e interacción de interfaz.	OK ▾
RF-12: Calificación de Viaje	T-012	Funcional	Validación de reglas tras viaje finalizado y uso de formulario UI.	OK ▾
RF-13: Crear Cuenta Cliente	T-013	Unitario + Funcional	Verifica backend + confirmación desde frontend	OK ▾

			con notificación.	
RF-14: Editar Datos Personales	T-014	Funcional	Requiere interacción directa con formulario y respuesta inmediata.	OK ▾
RF-15: Inicio de Sesión + OTP	T-015	Funcional	Flujo compuesto con validación visual y backend coordinado.	OK ▾
RF-16: Recuperar Contraseña	T-016	Funcional	Implica recuperación, correo y cambio interactivo desde frontend.	OK ▾
RF-17: Cierre de Sesión	T-017	Funcional	Acción directa en interfaz, validación de redirección y fin de sesión.	OK ▾

[Ver anexo: 7.3](#)

### 4.3 Cobertura de Pruebas

- Casos de uso críticos: 100 %
- Funcionalidades implementadas: 100 %
- Cobertura de código: 97%

## 5. Incidencias encontradas

A continuación se enumeran las incidencias identificadas durante la etapa de testing. Cabe destacar que otras inconsistencias fueron detectadas en etapas anteriores y corregidas de forma oportuna por el equipo de desarrollo, sin requerir su registro formal.

ID	Descripción	Riesgo	Estado
BUG-01	"Listado de viajes" queda bajo "Pasajes" y no "Viajes"	Bajo ▾	Corregido ▾
BUG-02	Tipeo en mensaje en "LocalidadController"	Bajo ▾	Corregido ▾
BUG-03	Tipeo en filtro en "Listado de viajes"	Bajo ▾	Corregido ▾
BUG-04	Cuando creo un viaje me lo crea para el día anterior al seleccionado.	Alto ▾	Corregido ▾

BUG-05	El frontend no mostraba las localidades inactivas.	Medio ▾	Corregido ▾
BUG-06	Iconos sin descripción en “Listado de ómnibus”.	Bajo ▾	Corregido ▾
BUG-09	App mobile crashea cuando se abre el calendario.	Alto ▾	Corregido ▾
BUG-10	Se podía modificar el campo email en el perfil del usuario.	Medio ▾	Corregido ▾
BUG-11	En el campo de calificación aparecen todas, no solamente la del usuario.	Medio ▾	Corregido ▾
BUG-12	Luego de realizada la compra, redireccionaba a localhost.	Medio ▾	Corregido ▾
BUG-13	Si se recarga la página se cierra la sesión.	Medio ▾	Corregido ▾
BUG-14	Compras realizadas no aparecen en “Mis viajes”.	Alto ▾	Corregido ▾
BUG-15	Se pueden devolver pasajes pasados.	Medio ▾	Corregido ▾















## 6. Conclusiones

El proceso de verificación permitió validar la funcionalidad, estabilidad e integridad del sistema desarrollado, a partir de una estrategia combinada de pruebas unitarias, funcionales y de extremo a extremo. Se logró cubrir la totalidad de los casos de uso críticos mediante pruebas automatizadas sobre el backend y pruebas E2E representativas desde la interfaz web, complementadas con pruebas manuales en dispositivos móviles. Las incidencias detectadas fueron corregidas oportunamente y no se identificaron fallos graves tras su resolución.













## 7. Anexos

### 7.1 Cobertura tests unitarios de servicios

Element ^	Class, %	Method, %
✓  com.example.Login.service	100% (15/15)	95% (105/110)
 AsientoService	100% (1/1)	100% (3/3)
 CategoriaService	100% (1/1)	100% (1/1)
 CerrarViajeService	100% (1/1)	100% (1/1)
 CompraPasajeService	100% (2/2)	100% (9/9)
 EmailService	100% (1/1)	100% (1/1)
 GenerarContraseniaService	100% (1/1)	100% (3/3)
 JwtService	100% (1/1)	100% (6/6)
 LocalidadService	100% (1/1)	100% (7/7)
 OmnibusService	100% (1/1)	100% (11/11)
 TokenService	100% (1/1)	100% (3/3)
 TransactionalService	100% (1/1)	100% (1/1)
 UsuarioService	100% (2/2)	100% (38/38)
 ViajeService	100% (1/1)	80% (21/26)

### 7.2 Cobertura tests unitarios de controladores

Element ^	Class, %	Method, %
✓  com.example.Login.controller	100% (10/10)	100% (73/73)
 AsientoController	100% (1/1)	100% (2/2)
 CategoriaController	100% (1/1)	100% (1/1)
 CerrarViajeController	100% (1/1)	100% (1/1)
 ComprasController	100% (1/1)	100% (3/3)
 LocalidadController	100% (1/1)	100% (6/6)
 OmnibusController	100% (1/1)	100% (10/10)
 TokenController	100% (1/1)	100% (2/2)
 UsuarioController	100% (2/2)	100% (30/30)
 ViajeController	100% (1/1)	100% (18/18)

## 7.3 Cobertura de Requisitos

```
test('Listado de Usuarios', async ({ page }) => {
  await page.goto('https://panel.tecnobus.uy');

  await expect(page.getByRole('link', { name: 'Image' })).toBeVisible();
  await expect(page.getByRole('link', { name: 'Administración' })).toBeVisible();
  await expect(page.getByRole('link', { name: 'Estadísticas' })).toBeVisible();
  await expect(page.getByRole('link', { name: 'Usuario' })).toBeVisible();
  await expect(page.getByText('admin', { exact: true })).toBeVisible();
  await expect(page.locator('div').filter({ hasText: /^a$/ })).toBeVisible();

  await page.getByRole('link', { name: 'Administración' }).click();
  await page.getByRole('link', { name: 'Listado usuarios' }).click();
  await expect(page.locator('div').filter({ hasText: 'Listado de' }).nth(2)).toBeVisible();
  await page.getByRole('button', { name: 'Choose' }).click();
  await page.getByText('50').click();
  await expect(page.getByRole('cell', { name: 'Jubilado', exact: true })).toBeVisible();
});
```

```
test('Editar datos personales', async ({ page }) => {
  await page.goto('https://tecnobus.uy');

  //Acceso al perfil de usuario
  await page.locator('div').filter({ hasText: /^A$/ }).click();
  await page.getByRole('link', { name: 'Perfil de Usuario' }).click();
  await expect(page.locator('div').filter({ hasText: 'Mi PerfilNombreApellidoEmailCedulaFecha de nacimientoDesactivar mi' })).nth(3)).toBeVisible();

  //Modifica Nombre
  await page.locator('div').filter({ hasText: /^Nombre$/ }).locator('i').click();
  await page.getByRole('textbox', { name: 'Nombre' }).click();
  await page.getByRole('textbox', { name: 'Nombre' }).fill('Alexa');
  await page.locator('div').filter({ hasText: /^Nombre$/ }).locator('i').click();

  //Modifica Apellido
  await page.locator('div').filter({ hasText: /^Apellido$/ }).locator('i').click();
  await page.getByRole('textbox', { name: 'Apellido' }).click();
  await page.getByRole('textbox', { name: 'Apellido' }).fill('Test2');
  await page.locator('div').filter({ hasText: /^Apellido$/ }).locator('i').click();

  //No tiene para modificar el Email
  await expect(page.locator('div').filter({ hasText: /^Email$/ }).locator('i')).toHaveCount(0);

  // Chequea que los campos están visibles y Guarda
  await expect(page.getByText('Desactivar mi cuenta')).toBeVisible();
  await expect(page.getByRole('button', { name: 'Guardar' })).toBeVisible();
  await expect(page.getByRole('link', { name: '¿Olvidaste tu contraseña?' })).toBeVisible();
  await page.getByRole('button', { name: 'Guardar' }).click();

  // Verifica que se muestre el mensaje de éxito
  await expect(page.locator('div').filter({ hasText: /^ÉxitoPerfil actualizado correctamente$/ })).nth(3)).toBeVisible();
  await expect(page.getByRole('alert')).toContainText('Éxito');
  await expect(page.getByRole('alert')).toContainText('Perfil actualizado correctamente');

  // Devuelve los cambios a los valores originales
  await page.locator('div').filter({ hasText: /^Nombre$/ }).locator('i').click();
  await page.getByRole('textbox', { name: 'Nombre' }).click();
  await page.getByRole('textbox', { name: 'Nombre' }).fill('Alex');
  await page.locator('div').filter({ hasText: /^Nombre$/ }).locator('i').click();
  await page.locator('div').filter({ hasText: /^Apellido$/ }).locator('i').click();
  await page.getByRole('textbox', { name: 'Apellido' }).click();
  await page.getByRole('textbox', { name: 'Apellido' }).fill('Test');
  await page.locator('div').filter({ hasText: /^Apellido$/ }).locator('i').click();
  await page.getByRole('button', { name: 'Guardar' }).click();
});
```

TESTING ...

TEST EXPLORER

Filter (e.g. text, !exclude, @tag)

1/1

tests 89.5s

- Admin.spec.js 20.5s
  - Listado de Usuarios 10.8s
  - Estadísticas de Usuarios 9.6s
- Cliente.spec.js 45.0s
  - Compra Pasaje 23.7s
  - Editar datos personales 8.1s
  - Historial de compras 6.4s
  - Calificar viaje 6.8s
- Vendedor.spec.js 24.1s
  - Reasignacion omnibus-viaje 12.1s
  - Listado de Pasajes Vendidos 12.0s

JS Admin.spec.js X JS Vendedor.spec.js JS Cliente.spec.js

```
tests > JS Admin.spec.js > ...
1 import { test, expect } from '@playwright/test';
2
3 // Usar sesión guardada
4 test.use({ storageState: 'authAdmin.json' });
5
6 > test('Listado de Usuarios', async ({ page }) => { ...
22 });
23
24
25 > test('Estadísticas de Usuarios', async ({ page }) => { ...
51 });
52 |
```

TESTING ...

TEST EXPLORER

Filter (e.g. text, !exclude, @tag)

1/1

tests 89.5s

- Admin.spec.js 20.5s
  - Listado de Usuarios 10.8s
  - Estadísticas de Usuarios 9.6s
- Cliente.spec.js 45.0s
  - Compra Pasaje 23.7s
  - Editar datos personales 8.1s
  - Historial de compras 6.4s
  - Calificar viaje 6.8s
- Vendedor.spec.js 24.1s
  - Reasignacion omnibus-viaje 12.1s
  - Listado de Pasajes Vendidos 12.0s

JS Admin.spec.js JS Vendedor.spec.js X JS Cliente.spec.js

```
tests > JS Vendedor.spec.js > ...
1 import { test, expect } from '@playwright/test';
2
3 // Usar sesión guardada
4 test.use({ storageState: 'authVendedor.json' });
5
6 > test('Reasignacion omnibus-viaje', async ({ page }) => { ...
39 });
40
41
42
43 > test('Listado de Pasajes Vendidos', async ({ page }) => { ...
75 });
76
77
78 |
```

TESTING ...

TEST EXPLORER

Filter (e.g. text, !exclude, @tag)

1/1

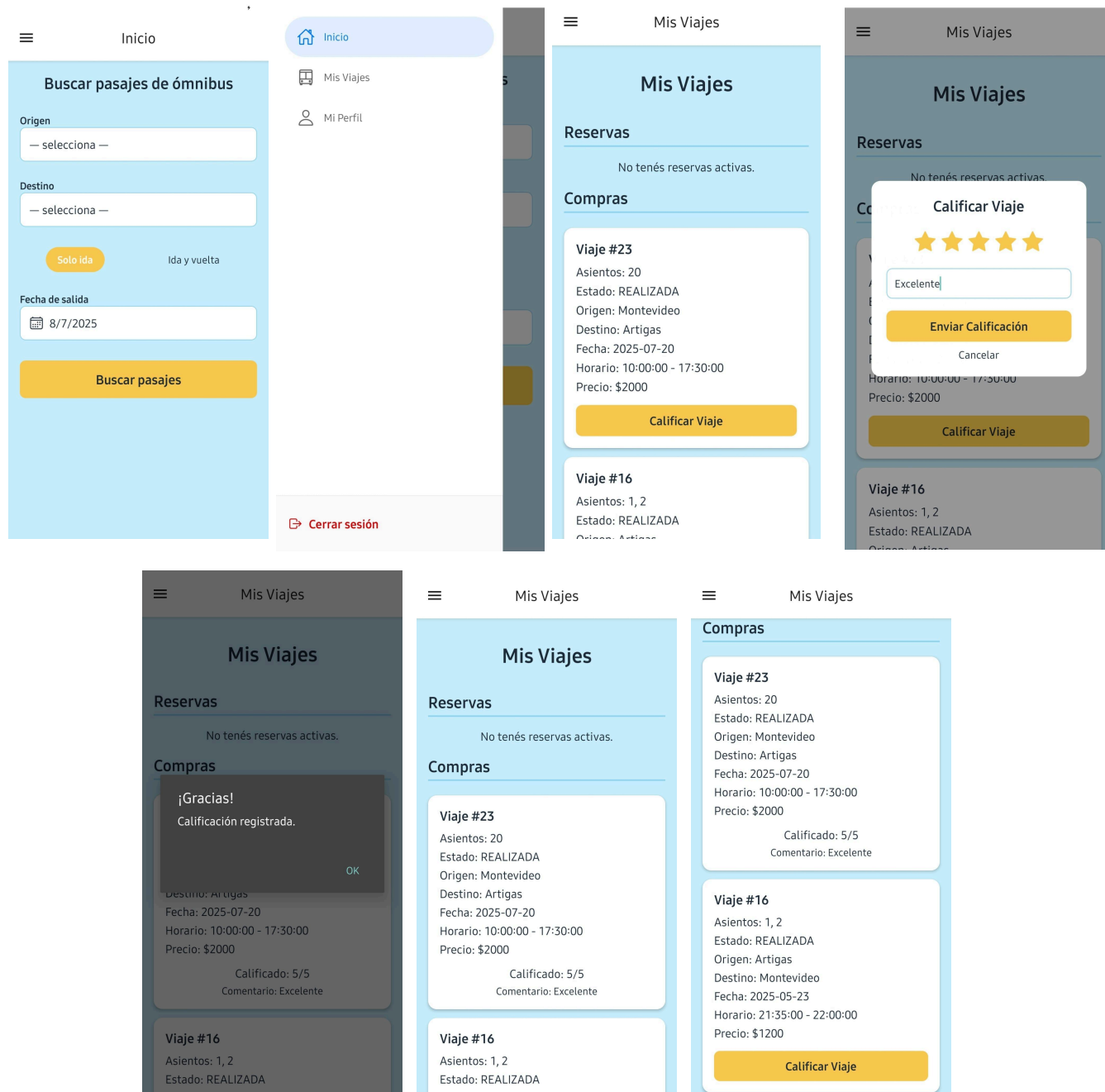
tests 89.5s

- Admin.spec.js 20.5s
  - Listado de Usuarios 10.8s
  - Estadísticas de Usuarios 9.6s
- Cliente.spec.js 45.0s
  - Compra Pasaje 23.7s
  - Editar datos personales 8.1s
  - Historial de compras 6.4s
  - Calificar viaje 6.8s
- Vendedor.spec.js 24.1s
  - Reasignacion omnibus-viaje 12.1s
  - Listado de Pasajes Vendidos 12.0s

JS Admin.spec.js JS Vendedor.spec.js JS Cliente.spec.js X

```
tests > JS Cliente.spec.js > ...
1 import { test, expect } from '@playwright/test';
2
3 // Usar sesión guardada
4 test.use({ storageState: 'authCliente.json' });
5
6 > test('Compra Pasaje', async ({ page }) => { ...
64 });
65
66
67 > test('Editar datos personales', async ({ page }) => { ...
112 });
113
114
115 > test('Historial de compras', async ({ page }) => { ...
140 });
141
142
143 > test('Calificar viaje', async ({ page }) => { ...
185 });
186 |
```

## 7.3.1 RF-11: Calificación de Viaje y RF-12: Historial de Compras



## 7.3.2 RF-14: Editar Datos Personales

The screenshots illustrate the user interface for editing personal data in a mobile application. The process begins on the 'Inicio' (Home) screen, where users can search for bus tickets by origin, destination, and date. The 'Mi Perfil' (My Profile) screen allows users to edit their personal information, including name, last name, email, ID number, and date of birth. A success dialog box confirms the update, and a 'Guardar' (Save) button is present at the bottom of the profile screen.

**Screenshot 1: Inicio**

Buscar pasajes de ómnibus

Origen  
Artigas, Artigas

Destino  
Montevideo, Montevideo

Solo ida Ida y vuelta

Fecha de salida  
30/7/2025

Buscar pasajes

**Screenshot 2: Mi Perfil**

Mis Viajes

Mi Perfil

**Screenshot 3: Mi Perfil**

Nombre Alexander

Apellido RodriguezGmail

Email alexander87rodriguez@gmail.com

Cédula CI

Fecha de nacimiento 2000-12-22

Desactivar mi cuenta

Guardar

**Screenshot 4: Mi Perfil**

Nombre AlexanderPropio

Apellido RodriguezGmail

Email alexander87rodriguez@gmail.com

Cédula CI

Fecha de nacimiento 2000-12-22

Desactivar mi cuenta

Guardar

**Screenshot 5: Mi Perfil**

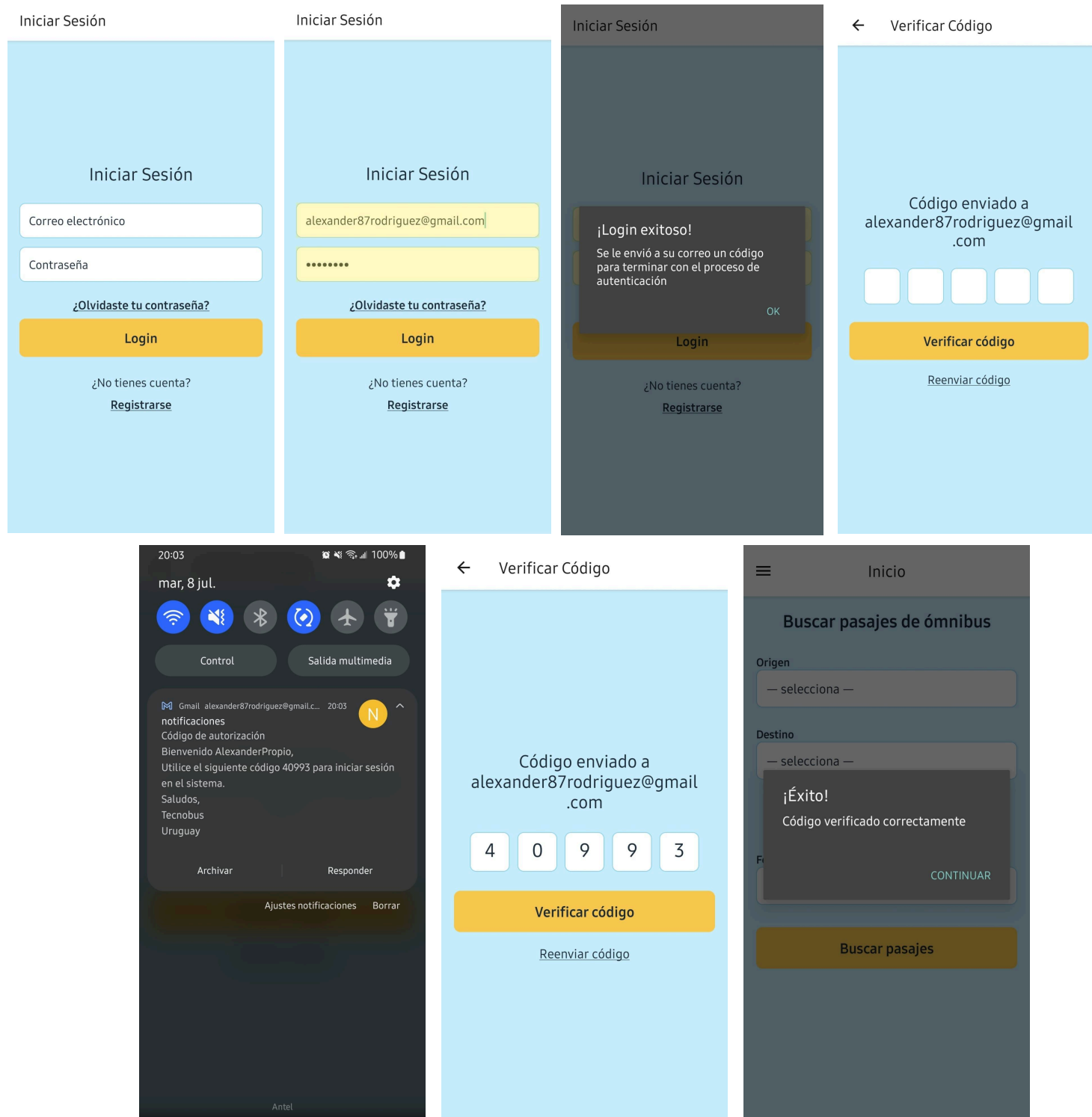
Éxito  
Perfil actualizado correctamente

OK

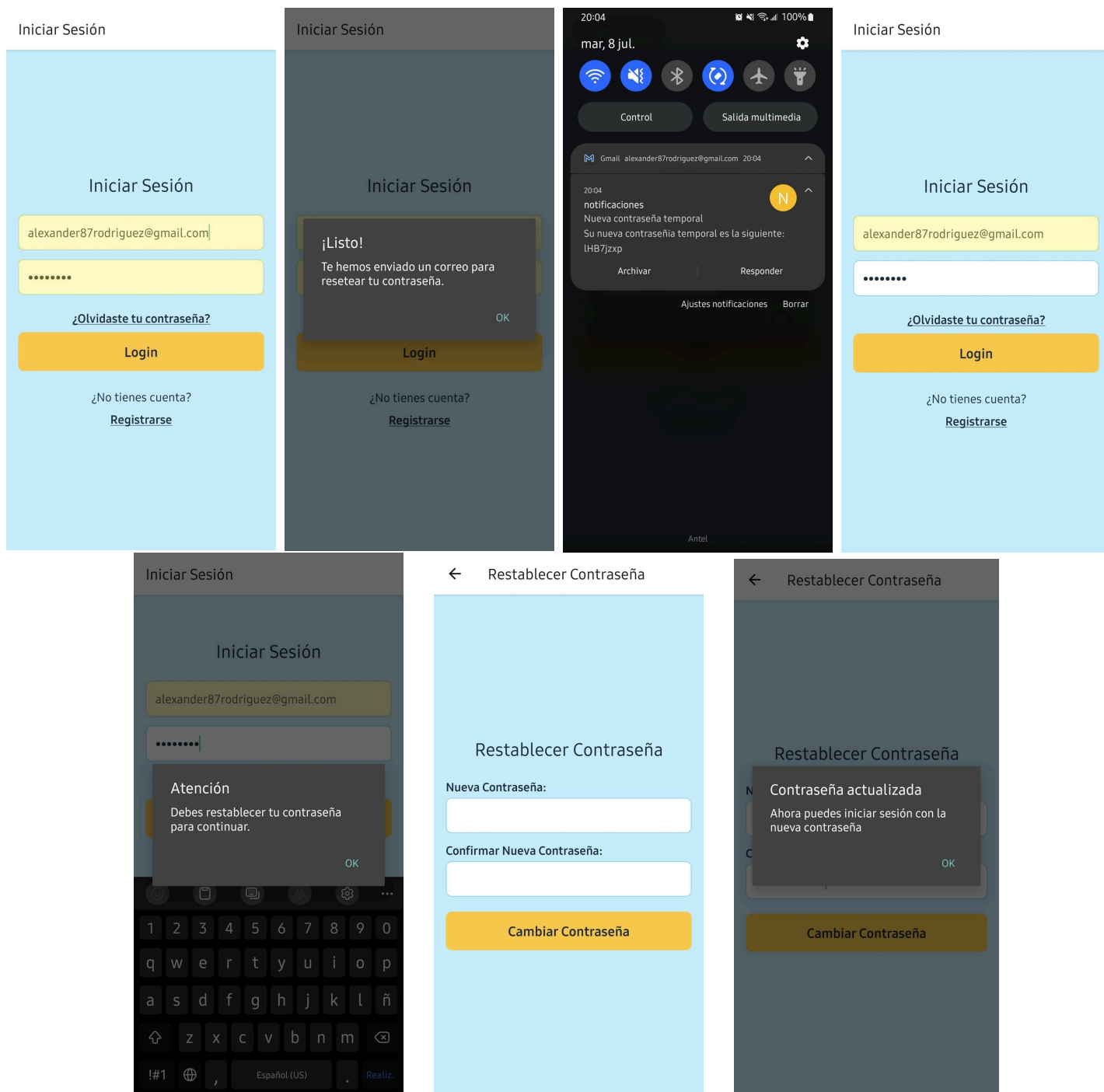
Desactivar mi cuenta

Guardar

### 7.3.3 RF-15: Inicio de sesión + OTP



## 7.3.4 RF-16: Recuperar Contraseña



### 7.3.5 RF-17: Cierre de Sesión

