**Aplicación Web de Recetas Médicas**

Documento de entrega: diseño, BD, normalización, UML, SCRUM, scripts y evidencias.

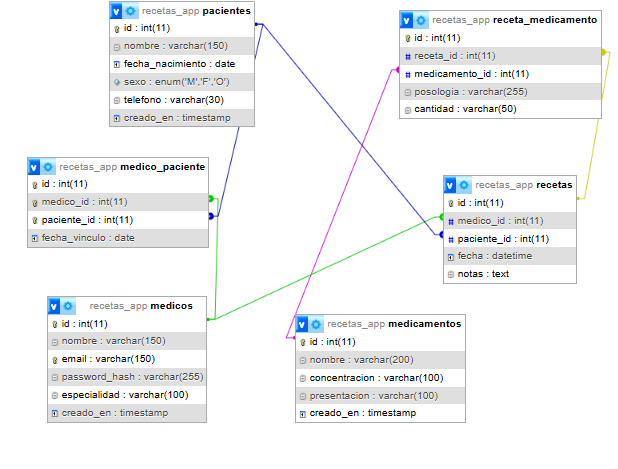
**1. Resumen del proyecto**

Aplicación web que permite a médicos autenticarse y emitir recetas a pacientes, con impresión en papel tamaño carta. Desplegable en servidor Apache con base de datos MySQL. Se siguen buenas prácticas de normalización (hasta BCNF), control de versiones con Git y planificación ágil (SCRUM).

**2. Modelo relacional (ER) — Resumen**

Entidades principales: medicos, pacientes, recetas, medicamentos, receta\_medicamento (tabla intermedia con posología). Relaciones:

* medicos 1 — N recetas (un médico emite muchas recetas)
* pacientes 1 — N recetas (un paciente puede tener muchas recetas)
* medicos N — M pacientes (se modela si hace falta: medico\_paciente)
* recetas N — M medicamentos a través de receta\_medicamento

**Diagrama ER **

**3. Script SQL (creación de tablas + datos de prueba)**

-- Base: recetas\_app

CREATE DATABASE IF NOT EXISTS recetas\_app DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci;

USE recetas\_app;

-- Tabla medicos

CREATE TABLE medicos (

id INT AUTO\_INCREMENT PRIMARY KEY,

nombre VARCHAR(150) NOT NULL,

email VARCHAR(150) NOT NULL UNIQUE,

password\_hash VARCHAR(255) NOT NULL,

especialidad VARCHAR(100),

creado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

) ENGINE=InnoDB;

-- Tabla pacientes

CREATE TABLE pacientes (

id INT AUTO\_INCREMENT PRIMARY KEY,

nombre VARCHAR(150) NOT NULL,

fecha\_nacimiento DATE,

sexo ENUM('M','F','O') DEFAULT 'O',

telefono VARCHAR(30),

creado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

) ENGINE=InnoDB;

-- Tabla medicamentos

CREATE TABLE medicamentos (

id INT AUTO\_INCREMENT PRIMARY KEY,

nombre VARCHAR(200) NOT NULL,

concentracion VARCHAR(100),

presentacion VARCHAR(100),

creado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

) ENGINE=InnoDB;

-- Tabla recetas

CREATE TABLE recetas (

id INT AUTO\_INCREMENT PRIMARY KEY,

medico\_id INT NOT NULL,

paciente\_id INT NOT NULL,

fecha DATETIME DEFAULT CURRENT\_TIMESTAMP,

notas TEXT,

FOREIGN KEY (medico\_id) REFERENCES medicos(id) ON DELETE RESTRICT ON UPDATE CASCADE,

FOREIGN KEY (paciente\_id) REFERENCES pacientes(id) ON DELETE RESTRICT ON UPDATE CASCADE

) ENGINE=InnoDB;

-- Tabla receta\_medicamento (N-M con posologia)

CREATE TABLE receta\_medicamento (

id INT AUTO\_INCREMENT PRIMARY KEY,

receta\_id INT NOT NULL,

medicamento\_id INT NOT NULL,

posologia VARCHAR(255) NOT NULL,

cantidad VARCHAR(50),

FOREIGN KEY (receta\_id) REFERENCES recetas(id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (medicamento\_id) REFERENCES medicamentos(id) ON DELETE RESTRICT ON UPDATE CASCADE

) ENGINE=InnoDB;

-- Tabla medico\_paciente (opcional: historial/relacion N-M)

CREATE TABLE medico\_paciente (

id INT AUTO\_INCREMENT PRIMARY KEY,

medico\_id INT NOT NULL,

paciente\_id INT NOT NULL,

fecha\_vinculo DATE DEFAULT (CURRENT\_DATE),

FOREIGN KEY (medico\_id) REFERENCES medicos(id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (paciente\_id) REFERENCES pacientes(id) ON DELETE CASCADE ON UPDATE CASCADE,

UNIQUE KEY uniq\_med\_pac (medico\_id, paciente\_id)

) ENGINE=InnoDB;

-- Datos de prueba mínimos

INSERT INTO medicos (nombre, email, password\_hash, especialidad) VALUES

('Dr. José López','jose.lopez@example.com', '$2y$12$EXAMPLEHASH', 'Medicina General');

INSERT INTO pacientes (nombre, fecha\_nacimiento, sexo, telefono) VALUES

('María Pérez','1985-04-12','F','5512345678'),

('Carlos Sánchez','1990-08-30','M','5598765432');

INSERT INTO medicamentos (nombre, concentracion, presentacion) VALUES

('Paracetamol','500 mg','Caja con 20 tabletas'),

('Amoxicilina','500 mg','Caja con 12 cápsulas');

-- Crear receta y relacionarla

INSERT INTO recetas (medico\_id, paciente\_id, notas) VALUES (1, 1, 'Consulta por dolor de cabeza recurrente');

INSERT INTO receta\_medicamento (receta\_id, medicamento\_id, posologia, cantidad) VALUES (1, 1, 'Cada 8 horas por 3 días', '12 tabletas');

**4. Normalización y justificación hasta BCNF**

**Primera forma normal (1FN)**: todas las tablas tienen atributos atómicos; la lista de medicamentos no se guarda en un campo repetible, sino en la tabla receta\_medicamento — cumple 1FN.

**Segunda forma normal (2FN)**: no hay dependencias parciales en tablas con claves compuestas (las tablas con claves naturales compuestas fueron evitadas; receta\_medicamento tiene su propia PK id y FKs), por lo que 2FN se satisface.

**Tercera forma normal (3FN)**: cada atributo no clave es funcionalmente dependiente de la clave primaria completa y no hay dependencias transitivas significativas (ej.: datos del médico/ paciente no se duplican en recetas).

**BCNF**: para cada dependencia funcional X -> Y, X debe ser superclave. Las relaciones que diseñamos (medicos, pacientes, medicamentos, recetas, receta\_medicamento) mantienen que las claves determinantes son PKs (o FKs que apuntan a PKs). No existen dependencias funcionales donde una columna no clave determine otras columnas no clave. Por lo tanto, el diseño cumple BCNF.

Nota: si se añadieran campos derivados (ej. edad en pacientes) habría que eliminarlos o marcarlos como calculables para preservar BCNF.

**5. Modelado UML**

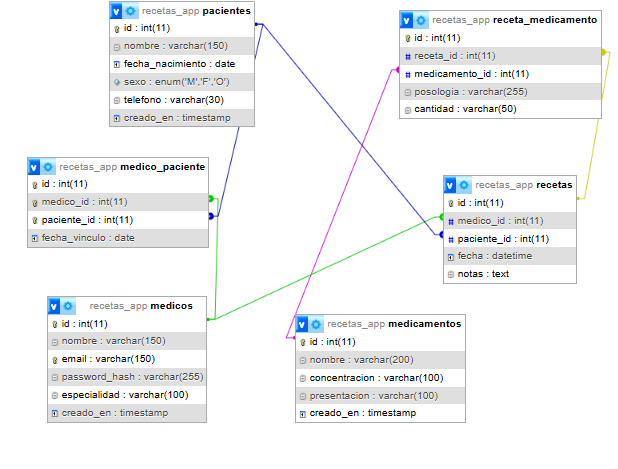
**5.1 Diagrama de casos de uso (texto)**

Actores: Médico, Sistema (autenticación, impresión)

Casos:

* Iniciar sesión
* Buscar/seleccionar paciente
* Crear nueva receta
* Agregar medicamento a receta
* Guardar receta
* Imprimir receta (formato carta)

**5.2 Diagrama de clases**



Medico "1" -- "\*" Receta

Paciente "1" -- "\*" Receta

Receta "1" -- "\*" RecetaMedicamento

Medicamento "1" -- "\*" RecetaMedicamento

**5.3 Diagrama de secuencia (emisión de receta)**

actor Medico

participant Web as WebApp

participant Server

participant DB as MySQL

Medico->>Web: Inicia sesión (email/password)

Web->>Server: POST /login

Server->>DB: SELECT medicos WHERE email=?

DB-->>Server: datos medico

Server-->>Web: sesión iniciada

Medico->>Web: Solicita nueva receta (selecciona paciente)

Web->>Server: GET /pacientes/{id}

Server->>DB: SELECT paciente

DB-->>Server: datos paciente

Medico->>Web: Agrega medicamento(s) y posología

Web->>Server: POST /recetas {medico\_id, paciente\_id, medicamentos[]}

Server->>DB: INSERT receta -> obtiene id

Server->>DB: INSERT receta\_medicamento (varias filas)

DB-->>Server: OK

Server-->>Web: retorna receta creada

Medico->>Web: Solicita impresión

Web->>Server: GET /recetas/{id}/print

Server->>DB: SELECT datos receta y medicamentos

DB-->>Server: datos

Server-->>Web: HTML para imprimir (formato carta)

**6. Arquitectura y tecnologías sugeridas**

* Back-end: PHP 8.x con PDO (o Node.js/Express si prefieres). Se proponen rutas simples: /login, /pacientes, /recetas, /recetas/{id}/print.
* Base de datos: MySQL 8.x.
* Servidor web: Apache (mod\_php o PHP-FPM + proxy).
* Control de versiones: Git (repositorio con ramas main y develop).
* Front-end: HTML5, CSS (Bootstrap o Tailwind), Javascript mínimo.
* Autenticación: sesiones PHP + password\_hash/password\_verify (bcrypt).

**7. Ejemplos de código (snippets)**

**7.1 Conexión PDO (config.php)**

<?php

// config.php

$dsn = 'mysql:host=localhost;dbname=recetas\_app;charset=utf8mb4';

$user = 'root';

$pass = 'tu\_password';

$options = [PDO::ATTR\_ERRMODE=>PDO::ERRMODE\_EXCEPTION, PDO::ATTR\_DEFAULT\_FETCH\_MODE=>PDO::FETCH\_ASSOC];

$pdo = new PDO($dsn, $user, $pass, $options);

**7.2 Login (login.php) — ejemplo simplificado**

<?php

// login.php (POST email,password)

require 'config.php';

$email = $\_POST['email'];

$pwd = $\_POST['password'];

$stmt = $pdo->prepare('SELECT id, password\_hash, nombre FROM medicos WHERE email=?');

$stmt->execute([$email]);

$user = $stmt->fetch();

if($user && password\_verify($pwd, $user['password\_hash'])){

session\_start();

$\_SESSION['medico\_id'] = $user['id'];

$\_SESSION['medico\_nombre'] = $user['nombre'];

header('Location: /dashboard.php');

exit;

} else {

echo 'Credenciales inválidas';

}

**7.3 Guardar receta (guardar\_receta.php)**

<?php

require 'config.php';

session\_start();

$medico\_id = $\_SESSION['medico\_id'];

$paciente\_id = $\_POST['paciente\_id'];

$notas = $\_POST['notas'];

$meds = $\_POST['medicamentos']; // esperado: array de objetos {medicamento\_id, posologia, cantidad}

$pdo->beginTransaction();

$stmt = $pdo->prepare('INSERT INTO recetas (medico\_id, paciente\_id, notas) VALUES (?,?,?)');

$stmt->execute([$medico\_id, $paciente\_id, $notas]);

$receta\_id = $pdo->lastInsertId();

$stmt2 = $pdo->prepare('INSERT INTO receta\_medicamento (receta\_id, medicamento\_id, posologia, cantidad) VALUES (?,?,?,?)');

foreach($meds as $m){

$stmt2->execute([$receta\_id, $m['medicamento\_id'], $m['posologia'], $m['cantidad']]);

}

$pdo->commit();

header('Location: /receta\_ver.php?id='.$receta\_id);

**7.4 Plantilla para imprimir (receta\_print.php)**

<?php

require 'config.php';

$id = intval($\_GET['id']);

$stmt = $pdo->prepare('SELECT r.\*, m.nombre as medico\_nombre, p.nombre as paciente\_nombre FROM recetas r JOIN medicos m ON r.medico\_id=m.id JOIN pacientes p ON r.paciente\_id=p.id WHERE r.id=?');

$stmt->execute([$id]);

$rec = $stmt->fetch();

$stmt = $pdo->prepare('SELECT rm.\*, md.nombre FROM receta\_medicamento rm JOIN medicamentos md ON rm.medicamento\_id = md.id WHERE rm.receta\_id=?');

$stmt->execute([$id]);

$items = $stmt->fetchAll();

?>

<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Receta #<?php echo $rec['id'];?></title>

<style>

@page { size: A4; }

body { font-family: Arial, sans-serif; margin: 20mm; }

.header{ text-align:left; }

</style>

</head>

<body onload="window.print()">

<div class="header">

<h2>Receta médica</h2>

<p><strong>Medico:</strong> <?php echo htmlspecialchars($rec['medico\_nombre']);?></p>

<p><strong>Paciente:</strong> <?php echo htmlspecialchars($rec['paciente\_nombre']);?></p>

<p><strong>Fecha:</strong> <?php echo $rec['fecha'];?></p>

</div>

<hr>

<h3>Medicamentos</h3>

<ul>

<?php foreach($items as $it): ?>

<li><?php echo htmlspecialchars($it['nombre']).' - '.$it['posologia'].' ('.$it['cantidad'].')';?></li>

<?php endforeach; ?>

</ul>

<hr>

<p>Notas: <?php echo nl2br(htmlspecialchars($rec['notas']));?></p>

</body>

</html>

**8. Despliegue en Apache (resumen de pasos)**

1. Instalar Apache + PHP + MySQL (XAMPP o LAMP stack en Linux).
2. Copiar la carpeta recetas\_app a /var/www/html/recetas\_app o C:\xampp\htdocs\recetas\_app.
3. Crear la base de datos con el script sql/creacion\_recetas\_app.sql.
4. Configurar config.php con credenciales correctas.
5. Asegurar permisos adecuados (propietario www-data en Linux).
6. Habilitar mod\_rewrite si usas URLs amigables.
7. Probar http://localhost/recetas\_app/login.php.

**9. Control de versiones (Git)**

Ejemplo de commits mínimos (mensajes claros):

* feat: crear esquema inicial de base de datos y script SQL
* feat: modelo back-end (config + conexión PDO)
* feat: login y manejo de sesiones para medicos
* feat: CRUD pacientes y medicamentos
* feat: crear y guardar recetas y relacionar medicamentos
* feat: plantilla de impresión en formato carta
* chore: añadir README y guías de despliegue

Mantén ramas: main (producción), develop (integración), feature/\* para nuevas características.

**10. Planeación SCRUM**

**Product Backlog (elementos principales)**

1. Autenticación de médicos
2. CRUD de pacientes
3. CRUD de medicamentos
4. Emisión de receta (crear, añadir medicamentos, guardar)
5. Impresión de receta (formato carta)
6. Interfaz de búsqueda/selección paciente
7. Seguridad: hashes de contraseña, validación de entradas
8. Tests básicos y pruebas manuales
9. Despliegue en Apache
10. Documentación y presentaciones

**Sprints propuestos (2 semanas cada uno)**

**Sprint 1 (2 semanas)**

* Tareas: configurar proyecto, BD, script SQL; conexión PDO; login médico; basic UI.
* Entregable: login funcional + base de datos creada.

**Sprint 2 (2 semanas)**

* Tareas: CRUD pacientes y medicamentos; pruebas unitarias básicas; UI de listados.
* Entregable: gestión de pacientes y medicamentos.

**Sprint 3 (2 semanas)**

* Tareas: implementación de creación de recetas, tabla intermedia; impresión.
* Entregable: flujo completo para crear e imprimir receta.

**Sprint 4 (1 semana)**

* Tareas: pruebas finales, documentación, preparación de entrega, capturas de pantalla.
* Entregable: proyecto desplegado y documentación lista.

**Tablero Kanban (ejemplo)**

| **Backlog** | **En progreso** | **En revisión** | **Hecho** |
| --- | --- | --- | --- |
| Autenticación | CRUD Pacientes | Revisión de pruebas | Despliegue en Apache |
| CRUD Medicamentos | Implementar impresión | Documentación | Login funcional |

**11. Estrategia de pruebas / Evidencias**

* Hacer pruebas manuales: crear médico, crear paciente, crear receta, imprimir.
* Tomar capturas: login.png, crear\_receta.png, receta\_print.png.
* Recolectar logs de servidor Apache y errores PHP si aparecen.

**Incluir en la entrega:** carpeta evidencias/ con las capturas y un README\_evidencias.md describiendo pasos.

**12. Checklist de entrega**

* Script SQL (incluido)
* Diagramas UML (incluidos)
* Código fuente básico (snippets incluidos)
* Instrucciones de despliegue (incluidas)
* Planificación SCRUM y tablero (incluidos)
* Capturas de pantalla (el estudiante debe generarlas en su entorno)
* Registro Git (logs y mensajes de commit)

**13. Sugerencias finales y mejoras opcionales**

* Añadir roles (admin, auditor) y permisos.
* Auditar cambios con tabla receta\_historial para trazabilidad.
* Implementar APIs REST para integración con apps móviles.
* Añadir validaciones y saneamiento exhaustivo (OWASP recommendations).