🧪 Información de Pruebas

Sistema de Gestión de Proyectos - Guía Completa para Testing

🚀 Acceso Rápido

Aplicación Web:
`http://localhost:5050`

Login:
`http://localhost:5050/auth/login`

⏱️ **Estado del Sistema:**
`http://localhost:5050/health`

🔐 Credenciales de Prueba

⚠️ **IMPORTANTE:** Estas son credenciales de desarrollo. NO usar en producción.

Email	Contraseña	Rol	Nombre
`admin@sistema.local`	`Maho#2024`	SUPERADMIN	Admin Sistema
`administrador@sistema.local`	`Admin#2024`	ADMIN	Administrador L1
`control@sistema.local`	`Control#2024`	CONTROL	Control de Proyectos
`usuario@sistema.local`	`Usuario#2024`	USUARIO	Usuario Operativo
`solicitante@sistema.local`	`Solicit#2024`	ADMIN	Solicitante Externo
`lector@sistema.local`	`Lector#2024`	LECTOR	Lector del Sistema

🗄️ Configuración de Base de Datos

Conexión Principal (Docker)

Host	`localhost`
Puerto	`3308` (mapeado a 3306 en contenedor)
Usuario	`proyectos_admin`

Contraseña	`Maho#2024`
Base de Datos	`proyectosDB`

Herramientas de Acceso

MySQL Workbench: Conéctate a `localhost:3308` con los datos anteriores

phpMyAdmin (si está configurado): `http://localhost:8080`

Línea de comandos:

```
mysql -h localhost -P 3308 -u proyectos_admin -p proyectosDB
```

Dentro del Contenedor Docker

Host: `mysql_db`

Puerto: `3306` (puerto interno)

🐳 Comandos Docker Esenciales

Gestión Básica

Iniciar:

```
docker-compose up -d
```

Detener:

```
docker-compose down
```

Reconstruir (después de cambios):

```
docker-compose up --build
```

Ver logs en tiempo real:

```
docker-compose logs -f proyectos_app
```

Migraciones de BD

Crear migración:

```
docker-compose exec proyectos_app flask db migrate -m "descripción"
```

Aplicar migraciones:

```
docker-compose exec proyectos_app flask db upgrade
```

Ver historial:

```
docker-compose exec proyectos_app flask db history
```

Acceso al Contenedor

Shell Python (Flask):

```
docker-compose exec proyectos_app flask shell
```

Bash en el contenedor:

```
docker-compose exec proyectos_app bash
```

Bash en MySQL:

```
docker-compose exec mysql_db bash
```

Endpoints Principales

Autenticación

- `GET/POST /auth/login` - Inicio de sesión
- `GET /auth/logout` - Cerrar sesión
- `GET /auth/register` - Registro (si está habilitado)

Dashboard

- `GET /dashboard` - Panel principal
- `GET /health` - Estado del sistema

Gestión de Permisos

- `GET/POST /permissions` - Gestión de permisos (SUPERADMIN)
- `GET /permissions/pages` - Lista de páginas
- `POST /permissions/assign` - Asignar permisos

Administración

- `GET /admin/backup/list` - Lista de backups
- `POST /admin/backup/create` - Crear backup
- `GET /admin/backup/stats` - Estadísticas

⚙️ Variables de Entorno Clave

Archivos de configuración:
- `.env` - Configuración para Docker
- `.env.local` - Configuración para desarrollo local (sobrescribe .env)

Variables Importantes

FLASK_ENV	`development` o `production`
SECRET_KEY	Clave para sesiones (cambiar en producción)
DATABASE_URL	`mysql://user:pass@host:port/db`
MYSQL_HOST	`mysql_db` (Docker) o `localhost` (local)
MYSQL_PORT	`3306` (Docker) o `3308` (local)

📊 Estructura de Base de Datos

Tablas Principales

trabajador - Usuarios del sistema (con UserMixin)

requerimiento - Solicitudes de trabajo

proyecto - Proyectos activos

area, sector, recinto - Estructura organizacional

especialidad - Especialidades de trabajo

user_page_permissions - Permisos por página (granular)

custom_role - Roles personalizados

Relaciones Clave

Trabajador → CustomRole: Asignación de roles personalizados

Requerimiento → Trabajador: M2M con tabla puente

Proyecto → Requerimiento: 1 a muchos

Trabajador → UserPagePermission: Permisos granulares

📐 Patrones y Convenciones de Código

Sistema de Permisos (OBLIGATORIO)

```
# Verificar permisos en endpoints if not (current_user.is_superadmin()
or current_user.has_page_permission('/ruta')): flash('No tiene
```

```
permisos', 'error') return redirect(url_for('main.dashboard'))
```

Estructura de Rutas

- `@login_required` - Obligatorio en todas las rutas protegidas
- `Flask-WTF CSRF` - Protección habilitada globalmente
- Blueprints por módulo: `auth_bp`, `admin_bp`, `permissions_bp`

Modelos (SQLAlchemy 2.0)

- Usar `TimestampMixin` para `created_at` y `updated_at`
- Implementar `UserMixin` en modelos de usuario
- Usar Enums para valores fijos: `UserRole.SUPERADMIN`

Formularios

- Usar `FlaskForm` para toda validación del lado servidor
- Validadores personalizados: `validates_rut()`, `validates_hexcolor()`
- Protección CSRF automática en `app/templates/bases/base.html`

🧪 Testing

Ejecutar Tests

Todos los tests:

```
pytest
```

Tests específicos:

```
pytest tests/test_permissions.py -v
```

Con cobertura:

```
pytest --cov=app
```

Archivos de Test

- `tests/test_permissions.py` - Tests del sistema de permisos
- `tests/test_endpoints.py` - Tests de rutas
- `tests/conftest.py` - Fixtures y configuración

🛠️ Scripts y Utilidades

Scripts Disponibles

manage.py - Gestión de base de datos y datos iniciales

init_app.py - Inicialización con wait_for_db

restore_backup_direct.py - Restauración de backups

diagnose_mysql_cleanup.py - Diagnóstico de MySQL

Ejecutar Scripts Locales

```
# Dentro del contenedor docker-compose exec proyectos_app python
manage.py seed-data # O en PowerShell local (requiere venv) python
manage.py seed-data
```

🔧 Troubleshooting Común

Error 400 (Bad Request)

Causas típicas:
• Token CSRF inválido o expirado
• Sesión expirada (timeout: 1 hora)
• Datos JSON con Content-Type incorrecto
• Formulario con datos malformados

Error 404 en `.well-known/appspecific/com.chrome.devtools.json`

✅ **Es normal.** Chrome DevTools lo solicita automáticamente. Solo genera WARNING en logs.

Problemas de Permisos

Verifica:
• Tabla `user_page_permissions` tiene entrada
• `current_user.is_active == True`
• Método `has_page_permission()` retorna correcto

Errores de Conexión a BD

```
# Ver estado del contenedor MySQL docker-compose ps # Ver logs de MySQL
docker-compose logs mysql_db # Reiniciar MySQL docker-compose restart
mysql_db
```

Limpiar y Reiniciar Todo

```
# Detener y eliminar volúmenes docker-compose down -v # Reconstruir sin
caché docker-compose build --no-cache # Levantar nuevamente docker-
compose up -d
```

📁 Directorios Importantes

/app	Código principal de la aplicación Flask
/app/models.py	Modelos SQLAlchemy (~1750 líneas)
/app/controllers	Blueprints separados por módulo
/app/routes	Rutas específicas (auth, admin, permissions)
/app/templates	Templates Jinja2
/app/static	CSS, JS, imágenes
/migrations	Historial de migraciones de BD
/tests	Archivos de test con pytest
/backups	Backups de base de datos

📚 Recursos Útiles

Documentación local: `/DOCS` - Incluye instrucciones detalladas

Manuales: `/00 MANUALES` - Comandos, terminal, documentación

Credenciales: `/USUARIOS de prueba.md`

.github/copilot-instructions.md: Instrucciones para AI/Copilot con arquitectura completa

✅ Checklist para Empezar a Probar

1. ✓ Docker Desktop abierto y en ejecución
2. ✓ `docker-compose up -d` completado exitosamente
3. ✓ Verificar `docker-compose ps` - todos los contenedores "Up"
4. ✓ Acceder a `http://localhost:5050/health` - debe retornar OK
5. ✓ Ir a `http://localhost:5050/auth/login`
6. ✓ Usar credenciales SUPERADMIN: `admin@sistema.local` / `Maho#2024`
7. ✓ Verificar que el dashboard se carga correctamente
8. ✓ Revisar logs: `docker-compose logs -f proyectos_app`
9. ✓ Conectar a BD con MySQL Workbench (localhost:3308)
10. ✓ ¡Listo para hacer pruebas!