Manual Técnico

RatBehavior

Fundación Universitaria Konrad Lorenz
Facultad de Matemáticas e Ingenierías

Manual Técnico del Simulador de Comportamiento para ratas de laboratorio de psicología

Presentado por:
Carlos Eduardo Martínez león

Herramientas usadas: VS Code. XAMPP

26 de noviembre de 2025

Control de Versiones

Versión	Fecha	Descripción	Autor
1.0	26/11/2025	Versión final del manual técnico	C.E. Martínez

Contenido

1. Introducción

El presente manual técnico describe el funcionamiento interno, arquitectura, componentes, base de datos, módulos, lógica de negocio y procesos del sistema **RatBehavior**, una aplicación web diseñada para apoyar prácticas de laboratorio en las que los estudiantes observan videos del comportamiento animal y registran las conductas observadas.

El sistema fue desarrollado empleando:

- **Python 3.x**
- **Flask** como framework backend
- **SQLAlchemy** para el ORM
- **MySQL** montado en **XAMPP**
- **HTML, CSS y Bootstrap** en el frontend
- **Flask-Login** para autenticación
- **PyMySQL** como conector base de datos

El manual ofrece una visión precisa del funcionamiento y es útil para programadores, mantenedores, administradores del sistema y docentes.

2. Alcance

2.1 Descripción General del Alcance

El sistema RatBehavior es una aplicación web desarrollada en Python utilizando el framework Flask, diseñada para apoyar la enseñanza de análisis conductual en entornos académicos. El presente capítulo establece los límites funcionales y no funcionales del sistema, definiendo qué procesos cubre, qué entidades administra, qué interacción habilita y cuáles quedan fuera del alcance.

El sistema opera bajo una arquitectura cliente-servidor, donde el backend procesa las operaciones lógicas y el frontend proporciona la interfaz para los estudiantes y administradores. La persistencia de datos se gestiona mediante MySQL ejecutado localmente por XAMPP.

2.2 Alcance Funcional

El alcance funcional corresponde al conjunto de funcionalidades implementadas dentro del sistema. RatBehavior abarca las siguientes operaciones primarias:

2.2.1 Gestión de Autenticación

- Inicio de sesión mediante credenciales registradas en la tabla usuarios.
- Validación de credenciales utilizando contraseñas cifradas.
- Control de sesiones mediante Flask-Login.
- Cierre de sesión seguro.

2.2.2 Presentación de Laboratorios Conductuales

- Visualización de videos asociados a laboratorios conductuales.
- Selección de un laboratorio específico registrado en la base de datos.
- Relación directa entre videos y conductas permitidas mediante la tabla videos_conductas.

2.2.3 Registro de Observaciones Conductuales

- Registro de las conductas seleccionadas por el estudiante durante la visualización del video.
- Almacenamiento automático en la tabla resultados.
- Validación de consistencia entre el video, las conductas habilitadas y el usuario autenticado.

2.2.4 Generación de Calificaciones

- Procesamiento automatizado del puntaje basado en las conductas seleccionadas.

- Registro de calificación en la tabla correspondiente.
- Presentación del resultado final al usuario en la interfaz.

2.2.5 Administración de Datos

(Para usuarios con rol administrativo)

- Gestión de videos (creación, edición, eliminación).
- Gestión de conductas.
- Gestión de asociaciones video–conducta.
- Consulta de evaluaciones realizadas por estudiantes.

2.3 Alcance No Funcional

Define las características operativas del sistema y las restricciones técnicas aplicadas.

2.3.1 Rendimiento

- Las operaciones de consulta deben ejecutarse en menos de 2 segundos en un entorno local.
- El sistema debe permitir concurrencia de múltiples usuarios sin degradación perceptible.

2.3.2 Seguridad

- Cifrado unidireccional de contraseñas.
- Protección de rutas mediante control de autenticación.
- Integridad referencial estricta en MySQL.
- Restricción de eliminación de datos con dependencias activas.

2.3.3 Compatibilidad

- Compatible con navegadores modernos (Chrome, Edge, Firefox).
- Ejecución del backend exclusivamente en Python 3.x.
- Servidor de base de datos provisto por MySQL/MariaDB en XAMPP.

2.3.4 Mantenibilidad

- Separación entre capas de presentación, lógica y persistencia.
- Código estructurado por módulos en Flask.
- Base de datos normalizada en 3FN.

2.4 Alcance de Datos del Sistema

El alcance de datos corresponde a todas las entidades administradas en MySQL. RatBehavior manipula la siguiente información:

- **usuarios:** credenciales, roles y datos básicos.
- **roles:** permisos de acceso.
- **videos:** enlaces y descripciones.
- **conductas:** catálogos de comportamientos observables.
- **videos_conductas:** asociaciones entre videos y conductas permitidas.
- **resultados:** registros de respuestas seleccionadas por cada estudiante.
- **evaluación final o puntaje:** resultado computado.

El alcance de datos incluye además todas las relaciones primarias y foráneas que garantizan consistencia interna.

2.5 Alcance de Interacción del Usuario

El sistema permite interacciones específicas según el perfil:

Estudiante

- Autenticarse.
- Seleccionar laboratorio.
- Observar el video.
- Registrar conductas.
- Visualizar calificación final.

Administrador

- Crear y editar usuarios.
- Registrar videos.
- Registrar conductas.
- Configurar relaciones video–conductas.
- Consultar evaluaciones.

No se contempla la carga de videos desde URL externas mediante la interfaz: solo mediante administrador.

2.6 Límites del Sistema

El sistema excluye explícitamente las siguientes características:

- No realiza análisis automatizado del video usando IA o visión computarizada.
- No incluye módulos de carga masiva de videos o conductas.
- No permite gestión de múltiples instituciones o sedes.
- No funciona en servidores remotos sin reconfiguración adicional.
- No ofrece integración con plataformas externas como Moodle o Google Classroom.
- No genera reportes descargables en PDF de forma nativa.

Estos límites definen el territorio funcional permitido para asegurar estabilidad y simplicidad operativa.

2.7 Interoperabilidad y Dependencias

El funcionamiento del sistema depende de:

- Python 3.x
- Flask, SQLAlchemy, Flask-Login
- XAMPP para MySQL
- Navegador web para acceso al frontend
- Conexión entre backend y base de datos vía protocolo mysql+pymysql

La interoperabilidad se limita al entorno local salvo ajustes adicionales.

2.8 Restricciones Operativas

- El servidor MySQL debe estar activo para iniciar la aplicación.
- No se permite eliminar registros con llaves foráneas activas.
- Solo usuarios autenticados pueden registrar evaluaciones.
- La ejecución depende de la estructura de carpetas del proyecto en Visual Studio Code.

2.9 Supuestos del Sistema

- Los estudiantes tienen conocimientos básicos de navegación web.
- El administrador dispone de acceso a phpMyAdmin.
- Los videos se encuentran previamente verificados y funcionales.
- El sistema opera en un entorno académico controlado.

3. Componentes Técnicos

A continuación, se describen las tecnologías que conforman la arquitectura del sistema RatBehavior, definiendo su función, su interacción con otros componentes y la justificación técnica de su uso.

3.1 Backend

Python 3.13.7

Lenguaje de programación de alto nivel utilizado para implementar toda la lógica del servidor. Ofrece una sintaxis clara, un amplio ecosistema de librerías científicas y frameworks web, así como compatibilidad directa con herramientas de inteligencia artificial y análisis de datos. Su uso en RatBehavior permite una implementación modular, mantenible y extensible.

Flask

Framework web minimalista que permite construir aplicaciones basadas en rutas, controladores y plantillas. Flask brinda:

- Servidor WSGI integrado.
- Manejo flexible de peticiones HTTP.
- Gestión de plantillas con Jinja2.
- Integración sencilla con SQLAlchemy y módulos de autenticación.

Se seleccionó Flask por su ligereza, adaptabilidad y por permitir un control granular de cada componente del sistema.

SQLAlchemy

ORM (Object-Relational Mapper) que estructura las tablas de MySQL como clases de Python, permitiendo manipular los datos mediante objetos en lugar de SQL puro. Con SQLAlchemy se garantiza:

- Abstracción de la base de datos.
- Mapeo limpio entre entidades y relaciones.
- Protección contra inyección SQL.
- Consultas optimizadas mediante el Core y el ORM.

3.2 Base de Datos (BD) MySQL

Motor de base de datos relacional empleado para almacenar la información persistente del sistema. Se utiliza por su estabilidad, soporte de transacciones, múltiples motores de almacenamiento y buena integración con SQLAlchemy. Maneja:

- Usuarios y roles.
- Videos.
- Conductas.
- Resultados de evaluaciones.
- Relaciones entre entidades.

XAMPP (Módulo MySQL/MariaDB)

Paquete que incluye Apache, PHP, Perl y MySQL. En este proyecto se emplea únicamente su servidor **MySQL/MariaDB**, que proporciona:

- Un entorno local reproducible.
- Herramientas de administración integradas (phpMyAdmin).
- Puertos y servicios estándar ampliamente documentados.

XAMPP permite ejecutar la base de datos de forma local sin necesidad de configuraciones complejas.

3.3 Frontend HTML5

Lenguaje de marcado utilizado para estructurar la interfaz del sistema. Permite definir:

- Formularios de acceso.
- Contenedores de video.
- Todos los elementos interactivos visibles para el usuario.

HTML5 aporta compatibilidad con todos los navegadores modernos y soporte nativo para multimedia.

Bootstrap 5

Framework CSS utilizado para mejorar la presentación visual y garantizar diseño responsivo. Su uso facilita:

- Componentes prediseñados (cards, modals, navbars).
- Estilos consistentes en toda la interfaz.

- Adaptación automática a distintas resoluciones (PC, tablet, móvil).

Bootstrap reduce significativamente los tiempos de diseño de interfaz.

3.4 ORM SQLAlchemy ORM

El módulo ORM de SQLAlchemy es el responsable del mapeo objeto-relacional. En RatBehavior se utiliza para:

- Definir las clases Usuarios, Conductas, Videos, Resultados, etc.
- Traducir automáticamente consultas en instrucciones SQL.
- Establecer relaciones uno-a-muchos y muchos-a-muchos.
- Gestionar sesiones transaccionales con la base de datos.

Permite trabajar la base de datos como si fuera parte natural del código Python.

3.5 Autenticación Flask-Login

Extensión dedicada a gestionar la autenticación de usuarios. Proporciona funcionalidades como:

- Manejo de sesiones mediante cookies seguras.
- Decoradores para proteger rutas (“@login_required”).
- Recuperación del usuario actual en cada petición.
- Integración con SQLAlchemy para cargar usuarios desde la BD.

Es la pieza clave que permite controlar el flujo de acceso al sistema.

3.6 Migraciones Alembic

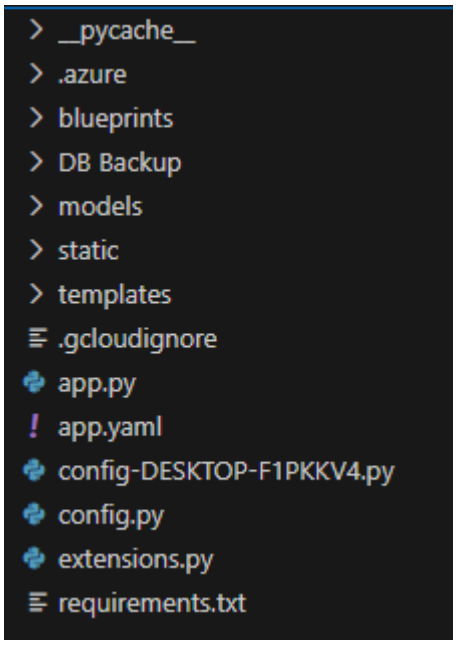
Herramienta de migraciones integrada con SQLAlchemy. Su propósito es:

- Aplicar cambios estructurales a la base de datos (crear tablas, modificar columnas, relaciones, etc.) sin afectar los datos existentes.
- Llevar un control de versiones del esquema.
- Mantener coherencia entre los modelos Python y las tablas MySQL.

En entornos educativos es útil para distribuir actualizaciones del sistema sin reinstalar la base de datos.

4. Arquitectura general del proyecto

```
├── app.py                    # Punto de entrada de la aplicación Flask
├── config.py                 # Configuración del sistema (MySQL, SECRET_KEY)
├── extensions.py             # Inicialización de SQLAlchemy, LoginManager
│
├── blueprints/
│   ├── auth.py               # Login, logout, autenticación
│   ├── main.py               # Pantalla de inicio, selección de test/lab
│   ├── laboratorio.py        # Carga de videos y conductas
│   ├── preguntas.py          # Gestión de preguntas y opciones
│   ├── evaluacion.py         # Registro de respuestas y cálculo de puntajes
│   └── admin.py              # Gestión de usuarios/roles/videos
│
├── models/
│   ├── usuarios.py
│   ├── roles.py
│   ├── videos.py
│   ├── conductas.py
│   ├── videos_conductas.py
│   ├── evaluaciones.py
│   ├── resultados.py
│   ├── tv_test.py
│   ├── tv_pregunta.py
│   ├── tv_opcion.py
│   ├── tv_usuario.py
│   └── tv_resultado.py
│
├── templates/                # HTML + Jinja2
│   ├── login.html
│   ├── index.html
│   ├── laboratorio.html
│   ├── preguntas.html
│   ├── opciones.html
│   ├── evaluar.html
│   └── resultados.html
│
├── static/                   # CSS, JS
├── DB Backup/                # Respaldos .sql
└── requirements.txt
```



La arquitectura del sistema RatBehavior responde a un modelo **cliente–servidor** estructurado en capas. Esta organización permite separar responsabilidades, facilitar el mantenimiento, mejorar la escalabilidad y garantizar una interacción coherente entre los componentes que conforman la aplicación.

4.1 Visión General de la Arquitectura

RatBehavior se compone de tres capas principales:

1. **Capa de Presentación (Frontend)**
2. **Capa Lógica o de Aplicación (Backend)**
3. **Capa de Persistencia (Base de Datos MySQL)**

Estas capas se comunican mediante protocolos estándar web (HTTP/HTTPS), consultas ORM gestionadas por SQLAlchemy y un motor de base de datos relacional.

4.2 Capa de Presentación (Frontend)

La capa de presentación corresponde a la interfaz con la que interactúa el usuario a través del navegador web. Está construida mediante:

- **HTML5** para la estructura de las páginas.
- **CSS/Bootstrap 5** para el diseño visual y comportamiento responsivo.
- **Jinja2** como motor de plantillas para integrar datos dinámicos provenientes del backend.

Responsabilidades de la Capa de Presentación:

- Capturar credenciales del usuario.
- Visualizar el catálogo de videos y laboratorios disponibles.
- Reproducir el video asignado a la práctica.
- Presentar los formularios de selección de conductas.
- Mostrar la calificación final y los resultados almacenados.

Comunicación con el Backend:

El navegador envía solicitudes HTTP (métodos GET/POST) a las rutas definidas en Flask.
Los datos ingresados por el usuario se envían al servidor para su procesamiento y persistencia.

4.3 Capa de Lógica de Negocio (Backend)

Implementada en **Python utilizando Flask**, esta capa contiene la lógica central de la aplicación. Es la responsable de procesar solicitudes, validar datos, aplicar reglas del negocio y comunicarse con la base de datos a través del ORM.

Componentes Principales:

4.3.1 Flask (Servidor Web WSGI)

Gestiona:

- rutas HTTP,
- sesiones,
- respuestas dinámicas,
- interacción con el frontend,
- manejo de estado del usuario.

4.3.2 SQLAlchemy (ORM)

Permite mapear las tablas como clases de Python:

- Usuarios
- Videos
- Conductas
- Videos_conductas
- Resultados

SQLAlchemy abstrae las operaciones SQL, creando una capa limpia entre el backend y la base de datos.

4.3.3 Flask-Login (Autenticación y Sesiones)

Se encarga de:

- validar credenciales,
- mantener sesión activa,
- proteger rutas para usuarios autenticados,
- administrar cookies seguras.

4.3.4 Controladores (Views)

Cada vista representa un endpoint que ejecuta acciones específicas:

- /login
- /logout
- /videos
- /evaluar/<id_video>
- /resultado

Estos controladores reciben solicitudes, procesan información y devuelven respuestas renderizadas con plantillas HTML.

4.3.5 Lógica de Evaluación

Contiene los algoritmos que:

- validan las conductas seleccionadas,
- comparan las respuestas del estudiante,
- calculan puntajes,
- generan el resultado final.

4.4 Capa de Persistencia (MySQL + XAMPP)

La información del sistema se almacena en un servidor **MySQL/MariaDB** provisto por XAMPP.
El modelo de datos está normalizado y utiliza claves primarias y foráneas para garantizar integridad referencial.

Tablas Principales:

- usuarios
- roles
- videos
- conductas
- videos_conductas
- resultados

Responsabilidades:

- Almacenar credenciales y roles.
- Mantener catálogo de videos experimentales.
- Registrar asociaciones entre videos y conductas permitidas.
- Guardar respuestas y calificaciones.
- Evitar inconsistencias mediante restricciones de integridad.

Comunicación Backend ↔ MySQL

El backend utiliza el driver **PyMySQL** bajo el protocolo mysql+pymysql, gestionado desde SQLAlchemy.

4.5 Flujo General de la Arquitectura

1. **El usuario ingresa al navegador** y accede a la URL de la aplicación.
2. La **capa de presentación** envía credenciales al backend.
3. Flask valida la autenticación con SQLAlchemy y la base de datos.
4. El usuario selecciona un laboratorio; el backend consulta MySQL.

5. El frontend muestra el video y las conductas cargadas desde la base.
6. El usuario marca las conductas observadas; el backend procesa la evaluación.
7. El resultado se guarda en MySQL.
8. El sistema muestra la calificación final al usuario.

4.6 Arquitectura Física

El sistema está diseñado para ejecutarse de forma local con:

- **Backend Python en Visual Studio Code**
- **Servidor MySQL (XAMPP) en el mismo equipo**
- **Navegador web como cliente**

Este despliegue reduce dependencias externas y facilita las prácticas académicas.

4.7 Arquitectura Lógica

El proyecto sigue el patrón:

Modelo (ORM)

Representación interna de entidades y relaciones.

Vista (HTML+Bootstrap)

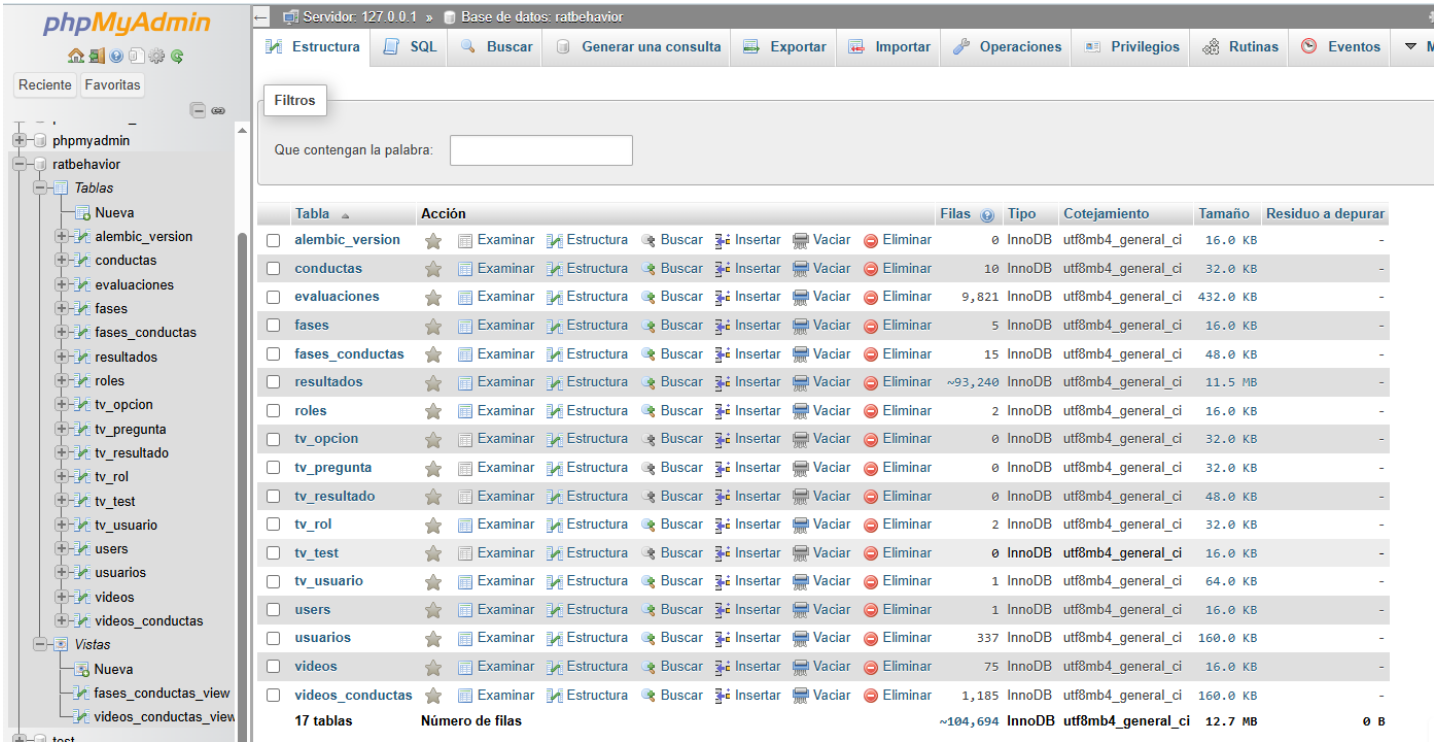
Interfaz final mostrada al usuario.

Controlador (Rutas en Flask)

Procesos que coordinan entradas, validan datos y devuelven respuestas.

Este enfoque es equivalente al patrón **MVC simplificado**, adecuado para aplicaciones Flask.

5. Base de Datos MySQL (XAMPP)



5.1 Tablas principales del sistema original

5.1.1. usuarios

Guarda los usuarios que ingresan al sistema.

Campo	Tipo	Descripción
id	int	PK
email	varchar	Usuario de acceso
password	varchar	Hash
nombres	varchar	Nombre del usuario
apellidos	varchar	Apellidos
idrol	int	FK → roles.id

5.1.2. roles

Define permisos del sistema.

Rol	Uso
Administrador	Puede gestionar toda la app
Estudiante	Solo realiza evaluaciones

5.1.3. videos

Videos de comportamiento (YouTube).

Campo	Tipo
id	int
urlvideo	varchar
descripcion	varchar

5.1.4. conductas

Conductas que el estudiante puede seleccionar.

5.1.5. videos_conductas

Relación N:N entre videos y conductas.

5.1.6. evaluaciones

Registro la fase o tipo de evaluación.

5.1.7. resultados

Guarda cada conducta seleccionada.

Este suele ser el más grande (en tu captura hay **93,240 filas**).

5.2 NUEVO BLOQUE: Tablas del módulo de TEST (TV)

Estas permiten crear **exámenes tipo test** con preguntas y opciones.

5.2.1. tv_test

Define cada test.

| Campo | Tipo | Descripción |
| id | int | PK |

| nombre | varchar | Nombre del test |
| descripcion | text | Detalles del test |

5.2.2. tv_pregunta

Preguntas del test.

Campo	Tipo
id	int
idtest	FK → tv_test.id
enunciado	Pregunta

5.2.3. tv_opcion

Opciones de cada pregunta.

Campo	Tipo
id	int
idpregunta	FK → tv_pregunta.id
texto	Opción
correcta	boolean

5.2.4. tv_usuario

Qué usuario hizo qué test.

5.2.5. tv_resultado

Respuestas del usuario.

Tu captura muestra **más de 104,694 registros**, indicando uso real del sistema.

5.2.6. tv_rol

Define roles para el módulo TV (si aplican).

6. Instalación

El sistema RatBehavior está diseñado para ejecutarse de forma local utilizando Python, Flask y MySQL a través de XAMPP. Para realizar la instalación en un entorno estándar, siga los pasos descritos a continuación:

6.1 Instalar Python

- Descargue Python 3.12 o superior desde: https://www.python.org
- Durante la instalación, active la opción **"Agregar Python a PATH"**.

6.2 Instalar XAMPP

- Descargue XAMPP desde: https://www.apachefriends.org
- Instale y abra el **Panel de control XAMPP**.
- Active el servicio **MySQL**.

6.3 Configuración de la Base de Datos

1. Inicie **phpMyAdmin** desde el panel de XAMPP.
2. Cree la base de datos con el nombre:
   RatBehavior
3. Importe el archivo SQL del proyecto, si lo tiene, o genere las tablas mediante SQLAlchemy.

6.4 Instalar el Proyecto

1. Obtenga la carpeta del proyecto RatBehavior.
2. Ábrala en Visual Studio Code o cualquier editor.
3. Instale las dependencias ejecutando en la terminal:

   pip install -r requirements.txt

6.5 Configurar la conexión a MySQL

En el archivo *config.py*, asegúrese de que la cadena de conexión sea:

```
SQLALCHEMY_DATABASE_URI = "mysql+pymysql://root:@127.0.0.1:3306/ratbehavior"
```

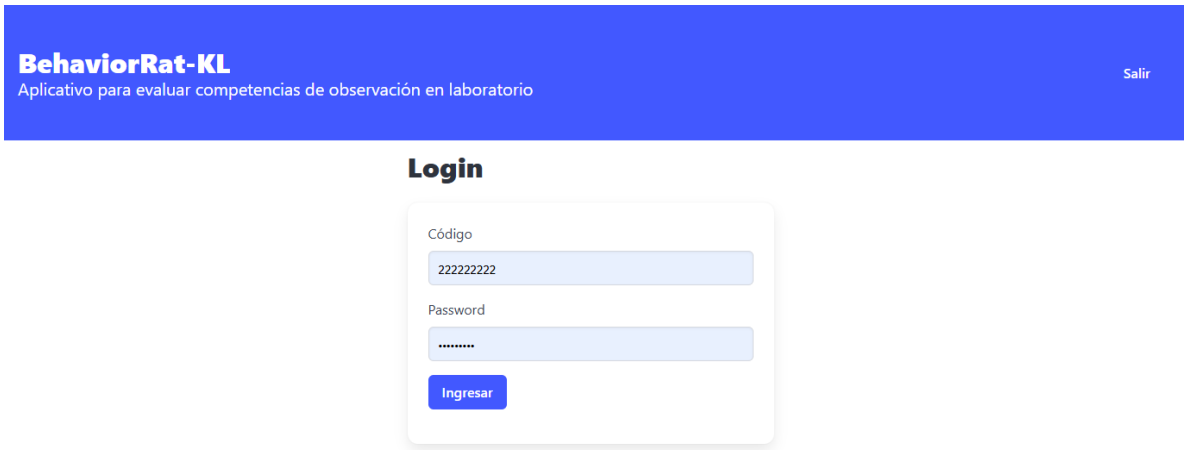
7. Flujo Interno del Sistema

```
PS C:\Users\User\OneDrive - Fundación Universitaria Konrad Lorenz\Escritorio\Entrega Profe Andrés> & C:/Users/User/AppData/Local/Programs/Python/Python
313/python.exe "c:/Users/User/OneDrive - Fundación Universitaria Konrad Lorenz/Escritorio/Entrega Profe Andrés/app.py"
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 358-743-553
```

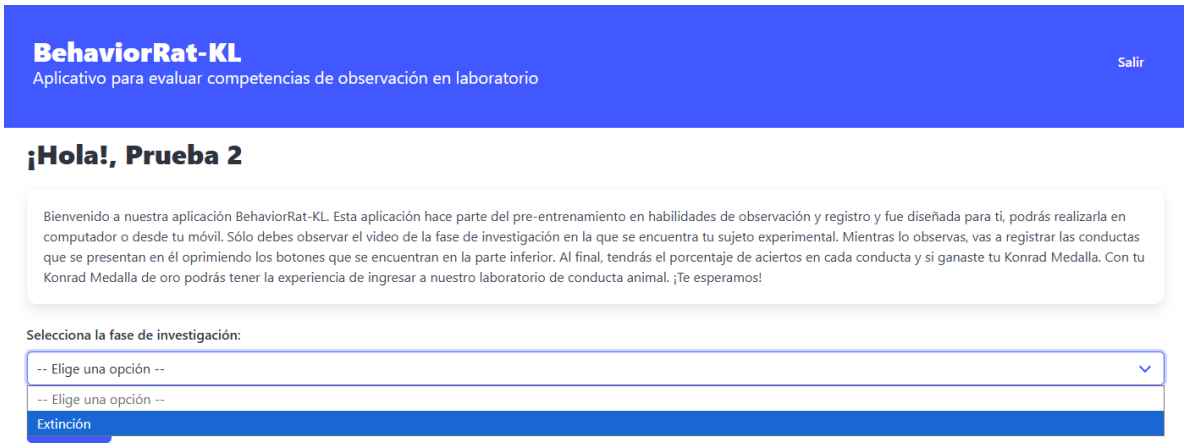
7.1 Se inicia con Python./app.py

7.2 Running on http://127.0.0.1:5000 para acceder a la pagina

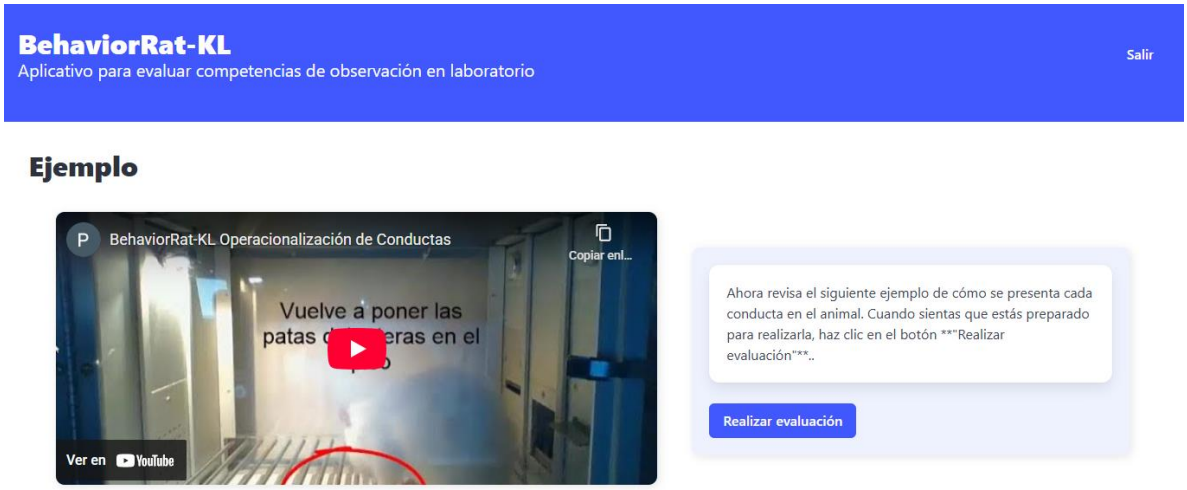
7.3 Se accede a la plataforma mediante usuario y contraseña



7.4 se selecciona el ejercicio a usar

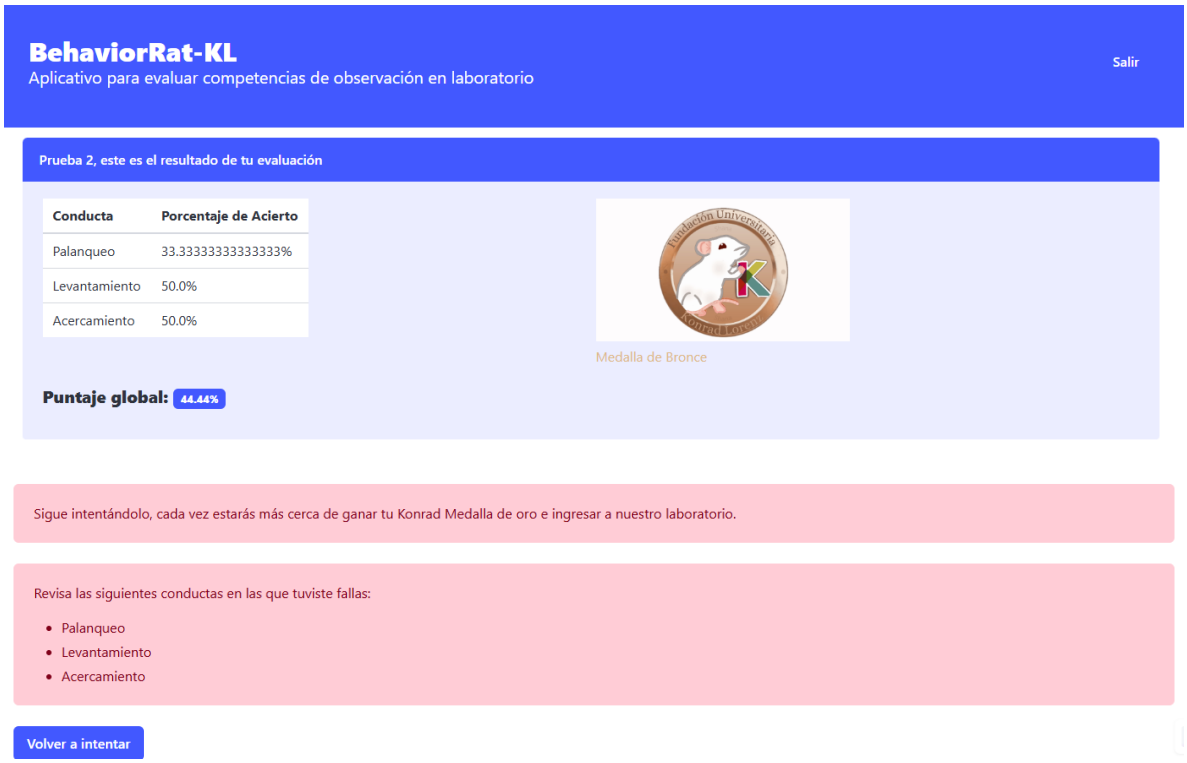


7.5 se debe seleccionar realizar prueba



7.6 al reproducir el video aparecen las 3 opciones para seleccionar en el trascurso del video.

8.7 al final da la calificación de acuerdo a lo seleccionado



8. Requisitos de Hardware y Software

Hardware

- Procesador Intel Core i3 o superior.
- Memoria RAM: **4 GB mínimo**, 8 GB recomendado.
- Espacio en discoteca: mínimo 2 GB libres.
- Resolución de pantalla recomendada: 1366×768 o superior.

Software

- Sistema operativo Windows 10 o Windows 11.
- Python 3.10 o superior.
- XAMPP con MySQL/MariaDB.
- Navegador actualizado (Chrome, Edge, Firefox).
- Visual Studio Code (opcional para desarrollo).
- PIP actualizado para la instalación de dependencias.

9. Manual de pruebas

El siguiente conjunto de pruebas garantiza el correcto funcionamiento del sistema RatBehavior.

9.1 Pruebas de Autenticación

- Usuario y contraseña correctos → ingreso exitoso.
- Contraseña incorrecta → acceso denegado.
- Intención de acceder a rutas protegidas sin sesión → redirección al login.

9.2 Pruebas de Carga de Videos

- Seleccione un laboratorio debe mostrar el vídeo correspondiente.
- El vídeo debe coincidir con los datos registrados en la tabla videos.
- La reproducción debe iniciarse sin errores.

9.3 Pruebas de Registro de Conductas

- Durante el video, al seleccionar una conducta, esta debe registrarse correctamente.
- Validar en la tabla resultados:
    - id_usuario

- id_video
- id_conducta
- momento/minuto
  se registren adecuadamente.

9.4 Pruebas de Cálculo de Calificación

- Selecciones correctas → calificación alta.
- Selecciones incorrectas → calificación baja.
- El puntaje debe mostrarse y almacenarse correctamente.

9.5 Pruebas del Módulo Administrador

- Crear un vídeo → debe aparecer en el listado.
- Editar un vídeo → cambios reflejados.
- Eliminar un vídeo → solo si no tiene registros dependientes.
- Crear conductas y asociarlas a videos.
- Visualizar evaluaciones de estudiantes.

9.6 Pruebas de Integridad de la Base de Datos

- Evitar eliminar videos con registros en resultados (restricción de FK).
- Los roles y usuarios deben mantener coherencia referencial.
- Las asociaciones de video–conducta deben cargarse sin errores.

10. Glosario Técnico

Administrador:
Usuario con permisos avanzados para gestionar videos, conductas, usuarios y evaluaciones dentro del sistema.

Alembic:
Herramienta de migraciones utilizada junto a SQLAlchemy para aplicar cambios estructurales a la base de datos sin afectar los datos existentes.

Apache / XAMPP:
entorno local que incluye MySQL/MariaDB y herramientas de administración como phpMyAdmin, utilizadas para ejecutar la base de datos del sistema.

Autenticación:
Proceso mediante el cual un usuario ingresa al sistema usando credenciales registradas, gestionado mediante Flask-Login.

Backend:
Capa lógica del sistema implementado en Python y Flask, encargada de procesar solicitudes, validar datos y comunicarse con la base de datos.

Bootstrap 5:
Framework CSS que define el diseño responsivo y visual del frontend del sistema.

Conductas:
Comportamientos observables que el estudiante selecciona durante la visualización del video y que son almacenados en la base de datos.

Controlador (View):
Función en Flask que recibe una petición HTTP, ejecuta la lógica asociada y devuelve una respuesta al usuario.

Datos Persistentes:
Información almacenada en MySQL que permanece disponible entre sesiones, como usuarios, videos, conductas y resultados.

Evaluación:
Proceso mediante el cual el sistema calcula la calificación del estudiante según las conductas seleccionadas durante un video.

Flask:
Framework web ligero utilizado para implementar el backend, manejar rutas y procesar solicitudes HTTP.

Flask-Login:
Extensión de Flask usada para gestionar sesiones, validar usuarios y proteger rutas del sistema.
Frontend:
Interfaz visual que interactúa directamente con el usuario y que está construida con HTML5, CSS y Bootstrap.
HTML5:
Lenguaje de marcado utilizado para crear las vistas del sistema y estructurar la interfaz de usuario.
Jinja2:
Motor de plantillas utilizado por Flask para integrar datos del backend dentro de páginas HTML dinámicas.
MySQL / MariaDB:
Motor de base de datos relacional utilizado para almacenar usuarios, vídeos, conductas, resultados y configuraciones del sistema.
ORM (Object-Relational Mapper):
Mecanismo que permite mapear tablas de la base de datos a clases de Python. En este sistema, es gestionado por SQLAlchemy.
phpMyAdmin:
Interfaz gráfica incluida en XAMPP para administrar MySQL de forma visual (creación de tablas, registros, consultas, etc.).
Python 3.x:
Lenguaje de programación utilizado para desarrollar la lógica del sistema, incluyendo las rutas, controladores y modelos.
Resultados:
Registros generados por los estudiantes durante una evaluación, que incluyen video, conducta y minuto de ocurrencia.
Roles:
Clasificación de permisos del sistema (Administrador o Estudiante), almacenada en la base de datos.
SQLAlchemy:
ORM que permite definir modelos, realizar consultas y administrar la base de datos desde Python sin escribir SQL directo.
Tabla:
Estructura dentro de MySQL que almacena entidades como usuarios, vídeos, conductas o resultados.
Usuario:
Persona registrada en el sistema que puede ser estudiante o administrador y que accede mediante autenticación.
Vídeo:
Elemento multimedia utilizado en la evaluación conductual. Está asociado a conductas mediante la tabla videos_conductas.
videos_conductas:
Tabla intermedia que define qué conductas están permitidas para cada video.

WSGI:
Interfaz estándar utilizada por Flask para comunicarse con el servidor web interno y gestionar solicitudes HTTP.