

UIPR - Recinto de Arecibo  
Programa de Ciencias de Computadoras

Sistema de Seguimiento de Tickets  
COMP 2052 - Server-Side Web Development and Back-End  
Microservices

Prof. J. Dastas Mendez  
Abimael Santa (R00627260)  
Génesis Ojeda (R00581877)

Repositorio Github:

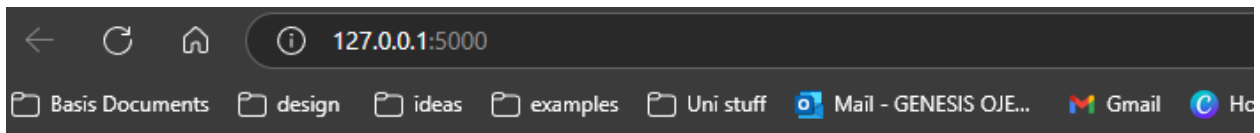
-Abimael: <https://github.com/Cruzade-ab/ProyectoFinalComp2052>

-Génesis: <https://github.com/DevQueenPR/ProyectoFinalComp2052>

## Capturas de pruebas realizadas con REST Client

Dashboard - Usuario, Técnico, Admin

Return: Retorna h1 que especifica que es un modo prueba



### **Corriendo en Modo de Prueba.**

ENDPOINT:

```
@main.route('/') # Ambas rutas llevan al mismo lugar
@main.route('/dashboard')
def index():
    """
    Página de inicio pública (home).
    """
    return '<h1>Corriendo en Modo de Prueba.</h1>'
```

## Creación de un ticket- Técnico o Admin

POST: Se le envían los campos requeridos para la creación de un ticket.

Return: Retorna un mensaje de ticket creado.

```
final_project > pruebas > create.rest > ...
1 | ## Crear un nuevo ticket (POST)
2 |
3 | Send Request
4 | POST http://localhost:5000/tickets
5 | Content-Type: application/json
6 |
7 | {
8 |   "asunto": "Error en servidor de Aguadilla",
9 |   "descripcion": "El sistema está presentando fallos al iniciar sesión.",
10 |   "prioridad": "Alta",
11 |   "estado": "Abierto",
12 |   "usuario_id": 12,
13 |   "tecnico_id": 8
14 | }

1 HTTP/1.1 201 CREATED
2 Server: Werkzeug/3.1.3 Python/3.13.2
3 Date: Fri, 09 May 2025 14:05:40 GMT
4 Content-Type: application/json
5 Content-Length: 44
6 Connection: close
7
8 {
9   "id": 8,
10  "message": "Ticket creado"
11 }
```

Resultado: Visualización de ticket.

```
[
  {
    "asunto": "Error cargando data en la web",
    "descripcion": "No carga correctamente",
    "estado": "Abierto",
    "fecha_creacion": "Wed, 07 May 2025 00:00:00 GMT",
    "id": 2,
    "prioridad": "Alta",
    "tecnico_id": 3,
    "usuario_id": 10
  },
  {
    "asunto": "Problema de red",
    "descripcion": "Problemas de conexión",
    "estado": "En proceso",
    "fecha_creacion": "Wed, 07 May 2025 00:00:00 GMT",
    "id": 3,
    "prioridad": "Baja",
    "tecnico_id": 8,
    "usuario_id": 11
  },
  {
    "asunto": "Problemas en laptop",
    "descripcion": "Problemas en laptop - Ventanilla negra.",
    "estado": "Abierto",
    "fecha_creacion": "Fri, 09 May 2025 08:15:31 GMT",
    "id": 5,
    "prioridad": "Alta",
    "tecnico_id": 13,
    "usuario_id": 11
  },
]
```

```
]
{
  "asunto": "Error en servidor",
  "descripcion": "El sistema está fallando",
  "estado": "Abierto",
  "fecha_creacion": "Fri, 09 May 2025 14:03:49 GMT",
  "id": 7,
  "prioridad": "Alta",
  "tecnico_id": 8,
  "usuario_id": 12
},
{
  "asunto": "Error en servidor de Aguadilla",
  "descripcion": "El sistema está presentando fallos al iniciar sesión.",
  "estado": "Abierto",
  "fecha_creacion": "Fri, 09 May 2025 14:05:40 GMT",
  "id": 8,
  "prioridad": "Alta",
  "tecnico_id": 8,
  "usuario_id": 12
},
{
  "asunto": "Sistema abajo",
  "descripcion": "El sistema está abajo actualmente.",
  "estado": "Abierto",
  "fecha_creacion": "Fri, 09 May 2025 14:09:49 GMT",
  "id": 9,
  "prioridad": "Alta",
  "tecnico_id": 3,
  "usuario_id": 10
}
```

ENDPOINT:

```
@main.route('/tickets', methods=['POST'])
def crear_ticket():
    """
    Crea un ticket sin validación.
    Espera JSON con 'titulo', 'descripcion' y 'tecnico_id'.
    """
    data = request.get_json()
```

```

if not data:
    return jsonify({'error': 'No input data provided'}), 400

ticket = Ticket(
    asunto=data.get('asunto'),
    descripcion=data.get('descripcion'),
    prioridad=data.get('prioridad'),
    estado=data.get('estado'),
    usuario_id=data.get('usuario_id'),
    tecnico_id=data.get('tecnico_id'),
    fecha_creacion=datetime.now(timezone.utc)
)

db.session.add(ticket)
db.session.commit()

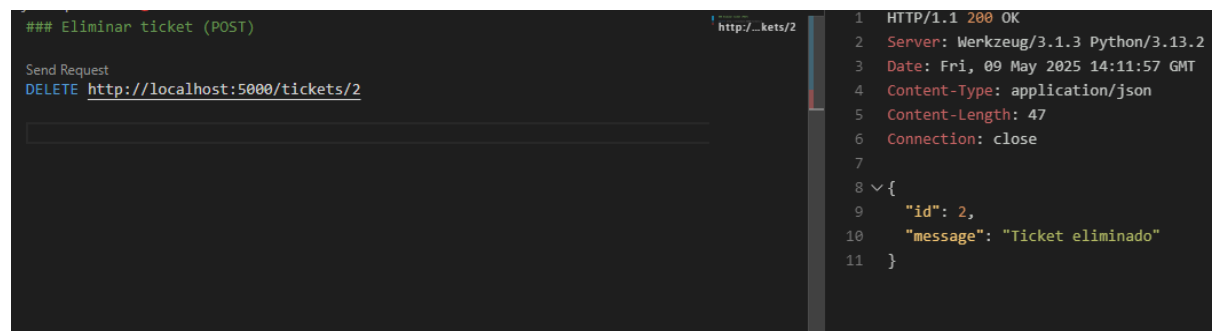
return jsonify({'message': 'Ticket creado', 'id': ticket.id,
'tecnico_id': ticket.tecnico_id}), 201

```

## Borrar un ticket - Técnico o Admin

DELETE: Borra el ticket seleccionado.

Return: Retorna un mensaje de que el ticket fue eliminado



```

### Eliminar ticket (POST)
Send Request
DELETE http://localhost:5000/tickets/2

1 HTTP/1.1 200 OK
2 Server: Werkzeug/3.1.3 Python/3.13.2
3 Date: Fri, 09 May 2025 14:11:57 GMT
4 Content-Type: application/json
5 Content-Length: 47
6 Connection: close
7
8 {
9   "id": 2,
10  "message": "Ticket eliminado"
11 }

```

## Resultado:

```
[
  {
    "asunto": "Problema de red",
    "descripcion": "Problemas de conexión",
    "estado": "En proceso",
    "fecha_creacion": "Wed, 07 May 2025 00:00:00 GMT",
    "id": 3,
    "prioridad": "Baja",
    "tecnico_id": 8,
    "usuario_id": 11
  },
  {
    "asunto": "Problemas en laptop",
    "descripcion": "Problemas en laptop - Ventanilla negra.",
    "estado": "Abierto",
    "fecha_creacion": "Fri, 09 May 2025 08:15:31 GMT",
    "id": 5,
    "prioridad": "Alta",
    "tecnico_id": 13,
    "usuario_id": 11
  },
  {
    "asunto": "Error en servidor",
    "descripcion": "El sistema está fallando",
    "estado": "Abierto",
    "fecha_creacion": "Fri, 09 May 2025 14:03:49 GMT",
    "id": 7,
    "prioridad": "Alta",
    "tecnico_id": 8,
    "usuario_id": 12
  },
]
```

```
{
  "asunto": "Error en servidor de Aguadilla",
  "descripcion": "El sistema está presentando fallos al iniciar sesión.",
  "estado": "Abierto",
  "fecha_creacion": "Fri, 09 May 2025 14:05:40 GMT",
  "id": 8,
  "prioridad": "Alta",
  "tecnico_id": 8,
  "usuario_id": 12
},
{
  "asunto": "Sistema abajo",
  "descripcion": "El sistema está abajo actualmente.",
  "estado": "Abierto",
  "fecha_creacion": "Fri, 09 May 2025 14:09:49 GMT",
  "id": 9,
  "prioridad": "Alta",
  "tecnico_id": 3,
  "usuario_id": 10
}
]
```

## ENDPOINT:

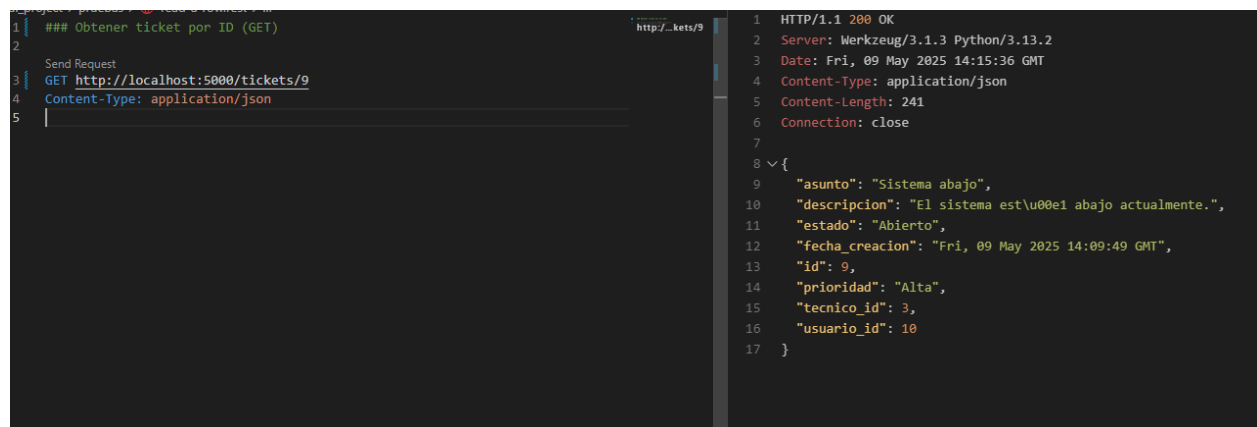
```
@main.route('/tickets/<int:id>', methods=['DELETE'])
def eliminar_ticket(id):
    """
    Elimina un ticket sin validación de permisos.
    """
    ticket = Ticket.query.get_or_404(id)
    db.session.delete(ticket)
    db.session.commit()
```

```
return jsonify({'message': 'Ticket eliminado', 'id': ticket.id}), 200
```

## Get Ticket By ID - Técnico, Admin

GET: Trae los datos asignados a ese ticket

Return: Retorna un json con los atributos del ticket



The screenshot shows a web browser window with the address bar displaying 'http://localhost:5000/tickets/9'. The page content shows the details of a ticket with ID 9. The response is a JSON object with the following fields: asunto, descripcion, estado, fecha\_creacion, id, prioridad, tecnico\_id, and usuario\_id.

```
1 ## Obtener ticket por ID (GET)
2
3 Send Request
4 GET http://localhost:5000/tickets/9
5 Content-Type: application/json
6
7
8 HTTP/1.1 200 OK
9 Server: Werkzeug/3.1.3 Python/3.13.2
10 Date: Fri, 09 May 2025 14:15:36 GMT
11 Content-Type: application/json
12 Content-Length: 241
13 Connection: close
14
15 {
16   "asunto": "Sistema abajo",
17   "descripcion": "El sistema est\u00e1 abajo actualmente.",
18   "estado": "Abierto",
19   "fecha_creacion": "Fri, 09 May 2025 14:09:49 GMT",
20   "id": 9,
21   "prioridad": "Alta",
22   "tecnico_id": 3,
23   "usuario_id": 10
24 }
```

## ENDPOINT:

```
@main.route('/tickets/<int:id>', methods=['GET'])
def listar_un_ticket(id):
    """
    Retorna un solo ticket por su ID (JSON).
    """
    ticket = Ticket.query.get_or_404(id)


    data = {
        'id': ticket.id,
        'asunto': ticket.asunto,
        'descripcion': ticket.descripcion,
        'prioridad': ticket.prioridad,
        'estado': ticket.estado,
        'usuario_id': ticket.usuario_id,
        'tecnico_id': ticket.tecnico_id,
        'fecha_creacion': ticket.fecha_creacion
    }
```

```
return jsonify(data), 200
```

## Get all Tickets - Técnico, Admin, User

## GET: Trae los datos de todos los tickets

**Return:** Retorna un json con los todos los tickets y sus atributos

```
final_project > pruebas >  read.rest > ...
1  ### Obtener todos los tickets (GET)
2
   Send Request
3  GET http://localhost:5000/tickets
4  Content-Type: application/json
5  
```

## Resultado:

```
{
  "asunto": "Error en servidor de Aguadilla",
  "descripcion": "El sistema está presentando fallos al iniciar sesión.",
  "estado": "Abierto",
  "fecha_creacion": "Fri, 09 May 2025 14:05:40 GMT",
  "id": 8,
  "prioridad": "Alta",
  "tecnico_id": 8,
  "usuario_id": 12
},
{
  "asunto": "Sistema abajo",
  "descripcion": "El sistema está presentando fallos al iniciar sesión.",
  "estado": "Abierto",
  "fecha_creacion": "Fri, 09 May 2025 14:09:49 GMT",
  "id": 9,
  "prioridad": "Alta",
  "tecnico_id": 3,
  "usuario_id": 10
}
```

```
localhost:5000/tickets
Basis Documents  design  ideas  examples  Uni stuff  Mail - GENESIS O
pretty-print ☒
{
  "asunto": "Problema de red",
  "descripcion": "Problemas de conexión",
  "estado": "En proceso",
  "fecha_creacion": "Wed, 07 May 2025 00:00:00 GMT",
  "id": 3,
  "prioridad": "Baja",
  "tecnico_id": 8,
  "usuario_id": 11
},
{
  "asunto": "Problemas en laptop",
  "descripcion": "Problemas en laptop - Ventanilla negra.",
  "estado": "Abierto",
  "fecha_creacion": "Fri, 09 May 2025 08:15:31 GMT",
  "id": 5,
  "prioridad": "Alta",
  "tecnico_id": 13,
  "usuario_id": 11
},
{
  "asunto": "Error en servidor",
  "descripcion": "El sistema está fallando",
  "estado": "Abierto",
  "fecha_creacion": "Fri, 09 May 2025 14:03:49 GMT",
  "id": 7,
  "prioridad": "Alta",
  "tecnico_id": 8,
  "usuario_id": 12
},
}
```

## ENDPOINT:

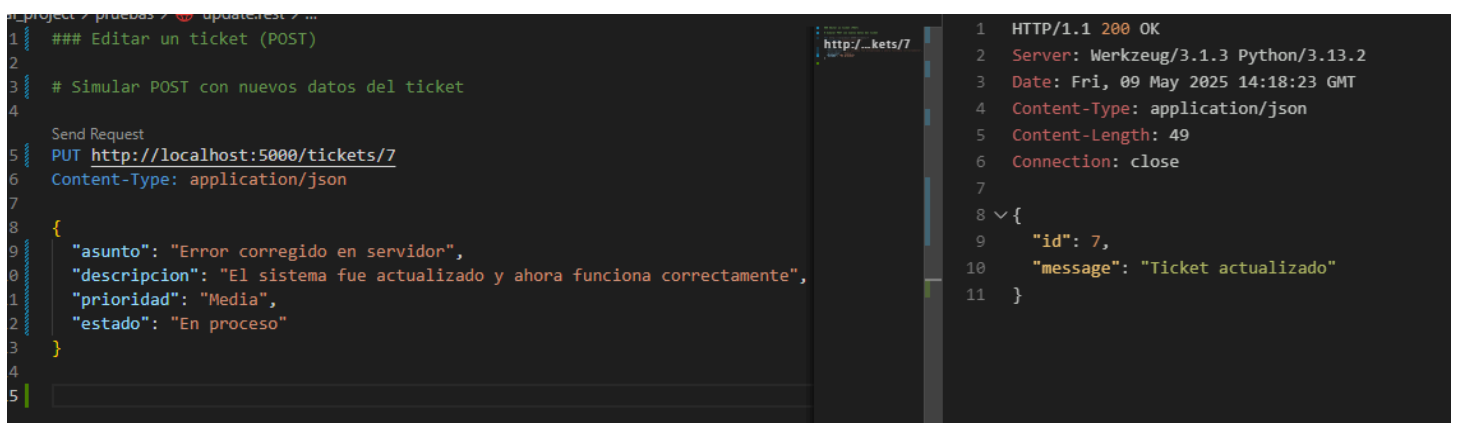
```
@main.route('/tickets', methods=['GET'])
def listar_tickets():
    """
    Retorna una lista de tickets (JSON).
    """
    tickets = Ticket.query.all()

    data = [
        {
            'id': ticket.id,
            'asunto': ticket.asunto,
            'descripcion': ticket.descripcion,
            'prioridad': ticket.prioridad,
            'estado': ticket.estado,
            'usuario_id': ticket.usuario_id,
            'tecnico_id': ticket.tecnico_id,
            'fecha_creacion': ticket.fecha_creacion
        }
        for ticket in tickets
    ]
    return jsonify(data), 200
```

## Update Ticket - Técnico, Admin

PUT: Actualiza los datos del ticket seleccionado

Return: Retorna un json con el mensaje de confirmación de que el ticket fue actualizado.



```
1 ## Editar un ticket (POST)
2
3 # Simular POST con nuevos datos del ticket
4
5 Send Request
6 PUT http://localhost:5000/tickets/7
7 Content-Type: application/json
8
9 {
10   "asunto": "Error corregido en servidor",
11   "descripcion": "El sistema fue actualizado y ahora funciona correctamente",
12   "prioridad": "Media",
13   "estado": "En proceso"
14 }
15
16 HTTP/1.1 200 OK
17 Server: Werkzeug/3.1.3 Python/3.13.2
18 Date: Fri, 09 May 2025 14:18:23 GMT
19 Content-Type: application/json
20 Content-Length: 49
21 Connection: close
22
23 {
24   "id": 7,
25   "message": "Ticket actualizado"
26 }
```



## Resultado:

```
{
  "asunto": "Error corregido en servidor",
  "descripcion": "El sistema fue actualizado y ahora funciona correctamente",
  "estado": "En proceso",
  "fecha_creacion": "Fri, 09 May 2025 14:03:49 GMT",
  "id": 7,
  "prioridad": "Media",
  "tecnico_id": 8,
  "usuario_id": 12
}
```

```
[
  {
    "asunto": "Problema de red",
    "descripcion": "Problemas de conexión",
    "estado": "En proceso",
    "fecha_creacion": "Wed, 07 May 2025 00:00:00 GMT",
    "id": 3,
    "prioridad": "Baja",
    "tecnico_id": 8,
    "usuario_id": 11
  },
  {
    "asunto": "Problemas en laptop",
    "descripcion": "Problemas en laptop - Ventanilla negra.",
    "estado": "Abierto",
    "fecha_creacion": "Fri, 09 May 2025 08:15:31 GMT",
    "id": 5,
    "prioridad": "Alta",
    "tecnico_id": 13,
    "usuario_id": 11
  },
  {
    "asunto": "Error corregido en servidor",
    "descripcion": "El sistema fue actualizado y ahora funciona correctamente",
    "estado": "En proceso",
    "fecha_creacion": "Fri, 09 May 2025 14:03:49 GMT",
    "id": 7,
    "prioridad": "Media",
    "tecnico_id": 8,
    "usuario_id": 12
  },
  {
    "asunto": "Error en servidor de Aguadilla",
    "descripcion": "El sistema está presentando fallos al iniciar sesión.",
    "estado": "Abierto",
    "fecha_creacion": "Fri, 09 May 2025 14:05:40 GMT",
    "id": 8,
    "prioridad": "Alta",
    "tecnico_id": 8,
    "usuario_id": 12
  },
  {
    "asunto": "Sistema abajo",
    "descripcion": "El sistema está abajo actualmente.",
    "estado": "Abierto",
    "fecha_creacion": "Fri, 09 May 2025 14:09:49 GMT",
    "id": 9,
    "prioridad": "Alta",
    "tecnico_id": 3,
    "usuario_id": 10
  }
]
```

## ENDPOINT:

```
@main.route('/tickets/<int:id>', methods=['PUT'])
def actualizar_ticket(id):
    """
    Actualiza un ticket sin validación de usuario o permisos.
    """
    ticket = Ticket.query.get_or_404(id)
    data = request.get_json()
```

```
ticket.asunto = data.get('asunto', ticket.asunto)
ticket.descripcion = data.get('descripcion', ticket.descripcion)
ticket.prioridad = data.get('prioridad', ticket.prioridad)
ticket.estado = data.get('estado', ticket.estado)
ticket.usuario_id = data.get('usuario_id', ticket.usuario_id)
ticket.tecnico_id = data.get('tecnico_id', ticket.tecnico_id)
ticket.fecha_creacion = data.get('fecha_creacion',
ticket.fecha_creacion)

    db.session.commit()

    return jsonify({'message': 'Ticket actualizado', 'id': ticket.id}),
200
```

# Código de los Endpoints Principales

1. “/”: Endpoint raíz que muestra home de la pagina

```
@main.route('/')
def index():
    """
    Página de inicio pública (home).
    """
    return render_template('index.html')
```

2. “/Dashboard”: Panel Principal que muestra los tickets en el sistema considerando su rol como usuario

```
@main.route('/dashboard')
@login_required
def dashboard():
    """
    Panel principal del usuario.
    """
    if current_user.role.name == 'Admin':
        tickets = Ticket.query.all()
    elif current_user.role.name == 'Técnico':
        tickets = Ticket.query.filter_by(tecnico_id=current_user.id).all()
    else:
        tickets = Ticket.query.filter_by(usuario_id=current_user.id).all()

    return render_template('dashboard.html', tickets=tickets)
```

3. “/tickets”: Ruta que contiene la función para devolver y listar los tickets

```

@main.route('/tickets', methods=['GET'])
def listar_tickets():
    """
    Retorna una lista de tickets en formato JSON.
    """
    try:
        # Obtener todos los tickets de la base de datos
        tickets = Ticket.query.all()

        # Formatear los datos en una lista de diccionarios
        data = [
            {
                'id': ticket.id,
                'asunto': ticket.asunto,
                'descripcion': ticket.descripcion,
                'prioridad': ticket.prioridad,
                'estado': ticket.estado,
                'usuario_id': ticket.usuario_id,
                'tecnico_id': ticket.tecnico_id,
                'fecha_creacion': ticket.fecha_creacion
            }
            for ticket in tickets
        ]

        # Retorna respuesta
        return ({'tickets': data}), 200

    except Exception as e:
        # Manejo de errores
        return ({'error': str(e)}), 500

```

4. "Tickets"[POST]: Ruta que maneja el formulario para la creación del ticket en el sistema, considerando los diferentes roles

```

@main.route('/tickets', methods=['GET', 'POST'])
@login_required
def tickets():
    form = TicketsForm()

    usuarios = User.query.filter(User.role_id == 2).all()

```

```

form.usuario_id.choices = [(u.id, u.username) for u in usuarios]

if current_user.role.name == 'Admin':
    tecnicos = User.query.filter(User.role_id == 3).all()
    form.tecnico_id.choices = [(t.id, t.username) for t in tecnicos]
else:
    # Si es técnico, su ID se asigna automáticamente
    form.tecnico_id.choices = [(current_user.id,
current_user.username)]
    form.tecnico_id.data = current_user.id

if form.validate_on_submit():
    ticket = Ticket(
        asunto=form.asunto.data,
        descripcion=form.descripcion.data,
        prioridad=form.prioridad.data,
        estado=form.estado.data,
        usuario_id=form.usuario_id.data,
        tecnico_id=form.tecnico_id.data
    )

    db.session.add(ticket)
    db.session.commit()

    flash("Ticket creado exitosamente.", "success")
    return redirect(url_for('main.dashboard'))
else:

    flash("Error en el formulario. Verifica los datos.", "danger")

return render_template('ticket_form.html', form=form)

```

5. “/tickets/id/editar”: Ruta que maneja el formulario para la data de un ticket existente y lo actualiza, Considerando a su vez los distintos roles != usuarios

```

@main.route('/tickets/<int:id>/editar', methods=['GET', 'POST'])

```

```

@login_required
def editar_ticket(id):
    """
    Permite editar un ticket existente. Solo si es admin o el técnico
    dueño.
    """
    ticket = Ticket.query.get_or_404(id)

    # Validación de permisos
    if current_user.role.name not in ['Admin', 'Técnico'] or (
        ticket.tecnico_id != current_user.id and current_user.role.name !=
'Admin'):
        flash('No tienes permiso para editar este ticket.')
        return redirect(url_for('main.dashboard'))

    form = TicketsForm(obj=ticket)

    # Cargar usuarios
    usuarios = User.query.filter(User.role_id == 2).all()
    form.usuario_id.choices = [(u.id, u.username) for u in usuarios]

    # Cargar técnicos
    if current_user.role.name == 'Admin':
        tecnicos = User.query.filter(User.role_id == 3).all()
        form.tecnico_id.choices = [(t.id, t.username) for t in tecnicos]
    else:
        form.tecnico_id.choices = [(ticket.tecnico.id,
ticket.tecnico.username)]

    if form.validate_on_submit():
        ticket.asunto = form.asunto.data
        ticket.descripcion = form.descripcion.data
        ticket.prioridad = form.prioridad.data
        ticket.estado = form.estado.data
        ticket.usuario_id = form.usuario_id.data

    # Solo admins pueden cambiar el técnico
    if current_user.role.name == 'Admin':
        ticket.tecnico_id = form.tecnico_id.data

```

```

        db.session.commit()
        flash("Ticket actualizado correctamente.")
        return redirect(url_for('main.dashboard'))

    return render_template('ticket_form.html', form=form, editar=True)

```

4. “/tickets/id/eliminar”: Ruta que maneja el formulario para la data de un ticket existente y lo elimina.

```

@main.route('/tickets/<int:id>/eliminar', methods=['POST'])
@login_required
def eliminar_ticket(id):
    """
    Elimina un ticket si el usuario es admin o su tecnico creador.
    """
    ticket = Ticket.query.get_or_404(id)

    if current_user.role.name not in ['Admin', 'Técnico'] or (
        ticket.tecnico_id != current_user.id and current_user.role.name !=
'Admin'):
        flash('You do not have permission to delete this course.') # Traducido
        return redirect(url_for('main.dashboard'))

    db.session.delete(ticket)
    db.session.commit()
    flash("Ticket deleted successfully.") # Traducido
    return redirect(url_for('main.dashboard'))

```

6. “/Usuarios”: Ruta que devuelve lista de usuarios. Considerando el rol necesario de Admin.

```

@main.route('/usuarios')
@login_required
def listar_usuarios():
    if current_user.role.name != 'Admin':

```

```

        flash("You do not have permission to view this page.")
        return redirect(url_for('main.dashboard'))

    # Obtener instancias completas de usuarios con sus roles (no usar
    .add_columns)
    usuarios = User.query.join(User.role).all()

    return render_template('usuarios.html', usuarios=usuarios)

```

7. “/usuarios/id/editar”: Ruta que maneja el formulario para actualización de un usuario. Considerando el rol necesario de Admin para esta

```

@main.route('/usuarios/<int:id>/editar', methods=['GET', 'POST'])
@login_required
def editar_usuario(id):
    if current_user.role.name != 'Admin':
        flash("No tienes permiso para editar usuarios.")
        return redirect(url_for('main.dashboard'))

    usuario = User.query.get_or_404(id)
    form = UserEditForm(obj=usuario)

    # choices: (id, nombre)
    form.role.choices = [(r.id, r.name) for r in Role.query.all()]

    if form.validate_on_submit():
        usuario.username = form.username.data
        usuario.email = form.email.data
        usuario.role_id = form.role.data # ahora es un int
        db.session.commit()
        flash("Usuario actualizado correctamente.")
        return redirect(url_for('main.listar_usuarios'))

    # Establecer el valor actual del select
    form.role.data = usuario.role_id

```



```
return render_template('usuario_form.html', form=form, editar=True)
```

8. '/usuarios/id/eliminar': Ruta que recibe el id de un usuario a eliminar

```
@main.route('/usuarios/<int:id>/eliminar', methods=['POST'])
@login_required
def eliminar_usuario(id):
    if current_user.role.name != 'Admin':
        flash("No tienes permiso para eliminar usuarios.")
        return redirect(url_for('main.dashboard'))

    usuario = User.query.get_or_404(id)
    db.session.delete(usuario)
    db.session.commit()
    flash("Usuario eliminado correctamente.")
    return redirect(url_for('main.listar_usuarios'))
```

9. '/login': Ruta que maneja el formulario, obteniendo los credenciales e inicia sesión.

```
@auth.route('/login', methods=['GET', 'POST'])
def login():
    print("Hola mundo")
    """
    Inicia sesión de un usuario existente si las credenciales son válidas.
    """
    form = LoginForm()

    if form.is_submitted(): #verifica si el formulario fue enviado
        if form.validate_on_submit():
            user = User.query.filter_by(email=form.email.data).first()
            print(f"Form submitted with email: {form.email.data}")

            if user and user.check_password(form.password.data):
                login_user(user)
                return redirect(url_for('main.dashboard'))

            flash('Email o contraseña incorrectos.', 'danger')
        else:
            flash('Formulario inválido. Verifica tus datos.', 'danger')
```

```
print("Renderizando log in")
return render_template('login.html', form=form)
```

10. /register: Ruta que maneja formulario de crear usuario y lo crea en la base de datos:

```
@auth.route('/register', methods=['GET', 'POST'])
def register():
    """
    Registra un nuevo usuario y lo asocia por defecto al rol "User".
    """
    form = RegisterForm()

    # Procesa el formulario si fue enviado correctamente
    if form.validate_on_submit():
        # Buscar el rol por nombre seleccionado
        role = Role.query.filter_by(name=form.role.data).first() # Puedes
        renombrar esto a 'Student' si cambias toda la app a inglés

        # Crea el usuario con datos del formulario
        user = User(
            username=form.username.data,
            email=form.email.data,
            role=role
        )
        user.set_password(form.password.data)
        print(f"user {user}")

        # Guarda en la base de datos
        db.session.add(user)
        db.session.commit()

        # Muestra mensaje de éxito
        print('User registered successfully.')
        flash('User registered successfully.')
        return redirect(url_for('auth.login'))
```

```
# Renderiza el formulario de registro
return render_template('register.html', form=form)
```

#### 11. “/logout”: Ruta para manejar el cierre de sesión

```
@auth.route('/logout')
def logout():
    """
    Cierra sesión del usuario actual y redirige al login.
    """
    logout_user()
    return redirect(url_for('auth.login'))
```

# Screenshots del Flujo de la Aplicación

## Registro de Usuario

127.0.0.1:5000/register

Tickets

127.0.0.1:5000/register

Username

RamontEagle

Email

skysision@gmail.com

Password

\*\*\*\*\*

Confirm password

\*\*\*\*\*

Role

Usuario

Register

Already have an account? [Log in here.](#)

Formulario para el registro de Usuario

127.0.0.1:5000/login

Tickets

127.0.0.1:5000/login

User registered successfully.

User Login

Email

skysision@gmail.com

Password

\*\*\*\*\*

Login

Don't have an account? [Register here.](#)

Mensaje de éxito y credenciales para login

127.0.0.1:5000/dashboard

Tickets

127.0.0.1:5000/dashboard

Ticket Management

Asunto	Description	Prioridad	Estado	Fecha Creación	Técnico Asignado	Usuario Asignado	Actions
You do not have permission to create, update or delete tickets.							

Dashboard Change Password Logout

Main Dashboard- Usuario

127.0.0.1:5000/change-password

Tickets

127.0.0.1:5000/change-password

Change Password

Current password

\*\*\*\*\*

New password

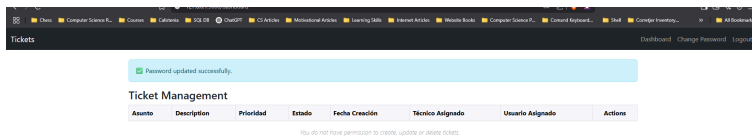
\*\*\*\*\*

Confirm new password

\*\*\*\*\*

Update Password

Formulario para cambiar password de usuario

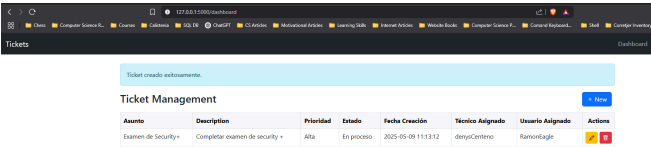


Mensaje de éxito de password cambiado

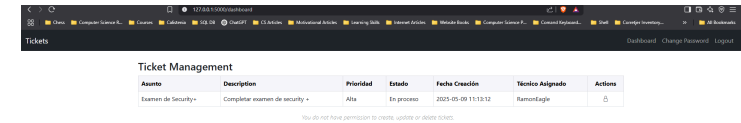
## Registro Usuario Rol: Técnico

Mensaje de éxito y credenciales para login

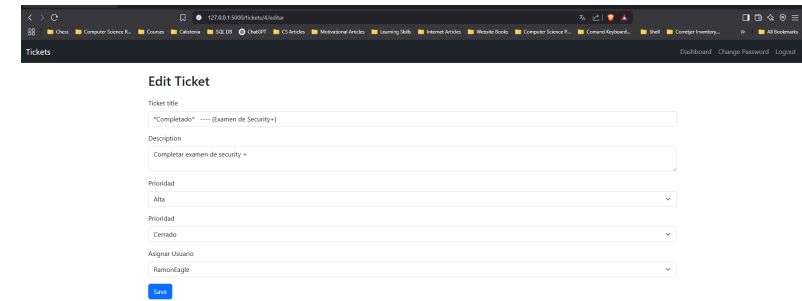
Creación de ticket por el técnico Denys Centeno



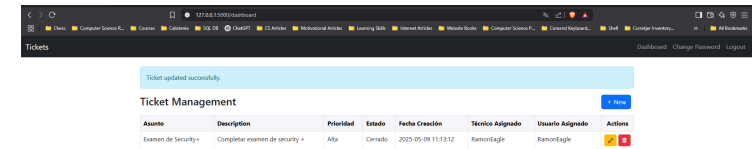
Mensaje de éxito de creación del ticket



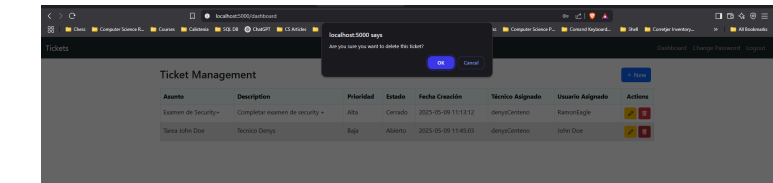
Cuenta de usuario Ramón Eagle donde se visualiza el ticket creado por técnico



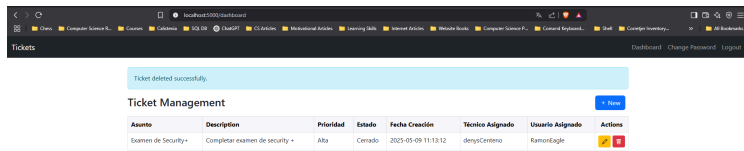
Edición de ticket con cuenta de técnico



Ticket guardado luego de la edición



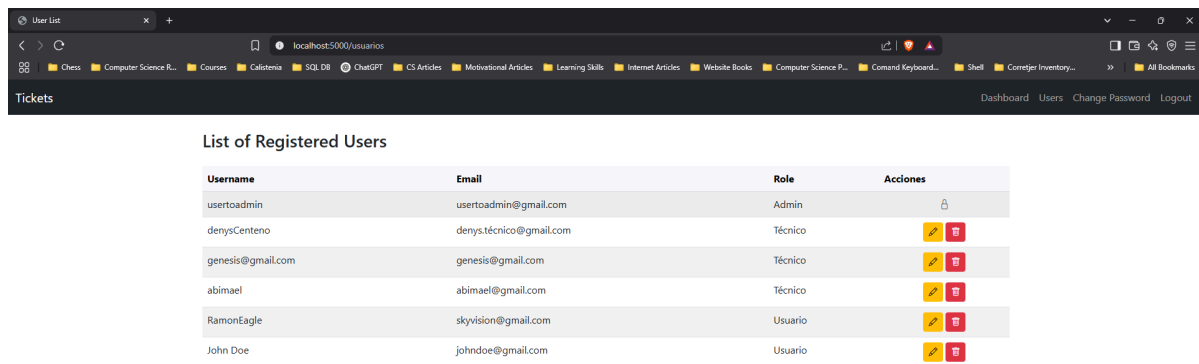
Alerta de mensaje de Eliminación



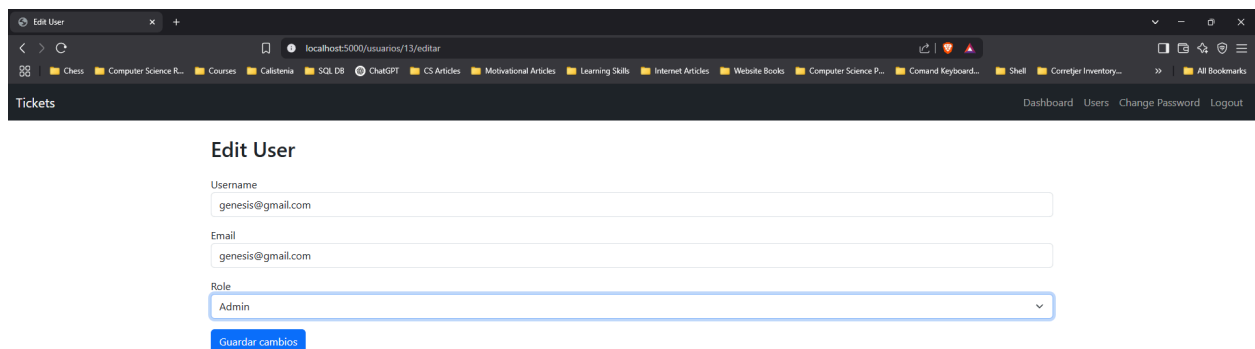
Eliminación de ticket

# Registro Admin

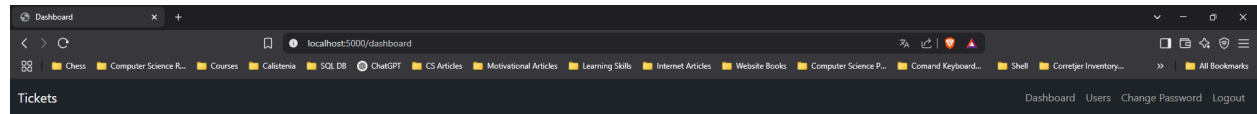
Lista de usuarios, técnicos y admins registrados



Edición de un usuario, técnico, admin desde cuenta de Admin



Creación de Ticket creado desde cuenta de admin



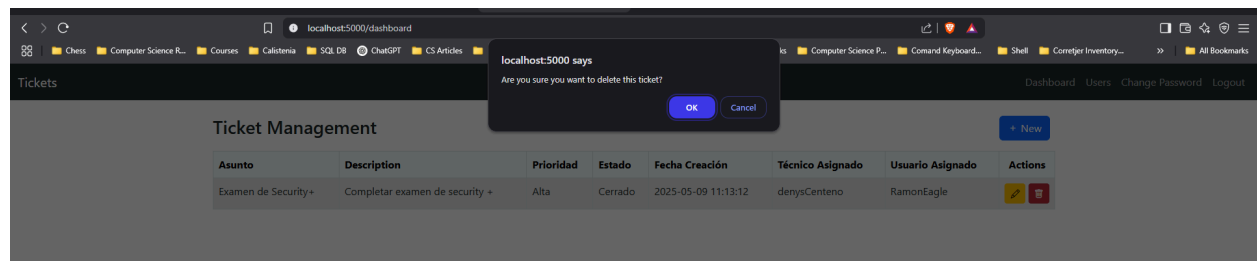
Ticket creado exitosamente.

### Ticket Management

+ New


Asunto	Description	Prioridad	Estado	Fecha Creación	Técnico Asignado	Usuario Asignado	Actions
Examen de Security+	Completar examen de security +	Alta	Cerrado	2025-05-09 11:13:12	RamonEagle	RamonEagle	 
*Completa la Información para la tarea asignada de John Doe *	Defina los parametros de la tarea para John Doe	Alta	Abierto	2025-05-09 11:43:53	John Doe	John Doe	 

### Mensaje de confirmación para borrar un ticket

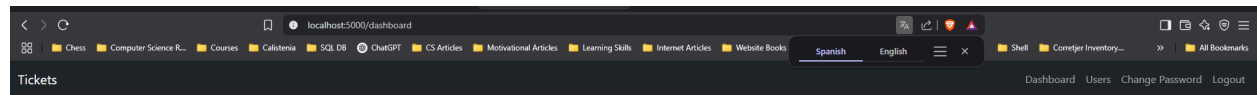


### Ticket Management

+ New

Asunto	Description	Prioridad	Estado	Fecha Creación	Técnico Asignado	Usuario Asignado	Actions
Examen de Security+	Completar examen de security +	Alta	Cerrado	2025-05-09 11:13:12	denysCenteno	RamonEagle	 

### Eliminación de ticket desde cuenta de admin



Ticket deleted successfully.

### Ticket Management

+ New

Asunto	Description	Prioridad	Estado	Fecha Creación	Técnico Asignado	Usuario Asignado	Actions
--------	-------------	-----------	--------	----------------	------------------	------------------	---------