Corpus Christi: A Web Application For Course Enrollment

Ross Otto, Josh James, Brandon George, Josiah Keller, Imani Muya,
Jacob Banks, Eliza You, Ryan Jones, Sam Skinner, Lily Pratico

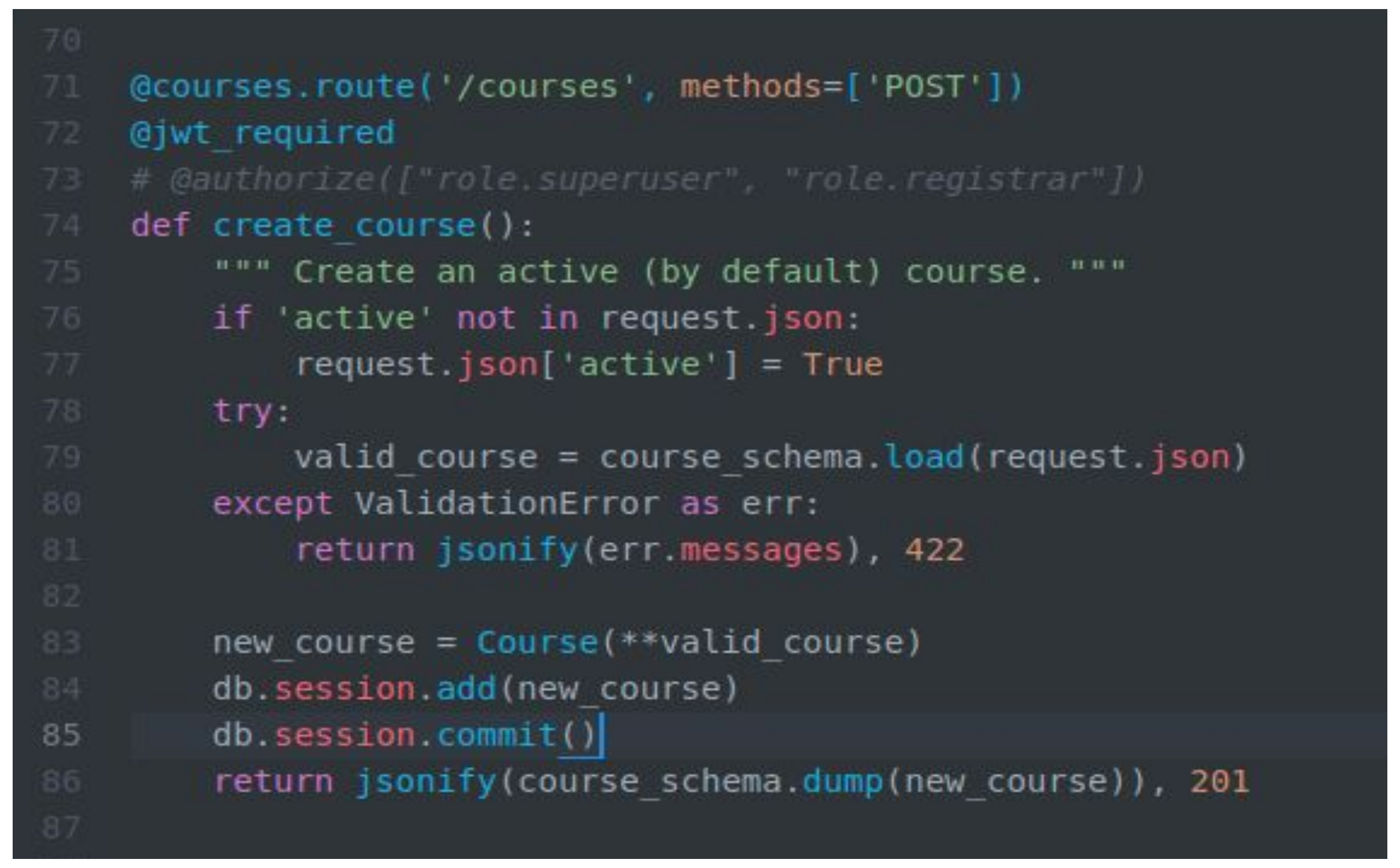
Background

Arco Iglesia Cristiana is a church in Cuenca, Ecuador. Our missions computing group was tasked with creating a web application to service their administrative needs. Our team was responsible for creating a system that would allow different users to create, host, and attend Christian fellowship courses. In addition, diplomas can be awarded to enrolled students after completion of essential courses. The application is able to support various user types (e.g., Admin, Registrar, Public) that interact with the system in different ways.

Technology Stack

User Interface

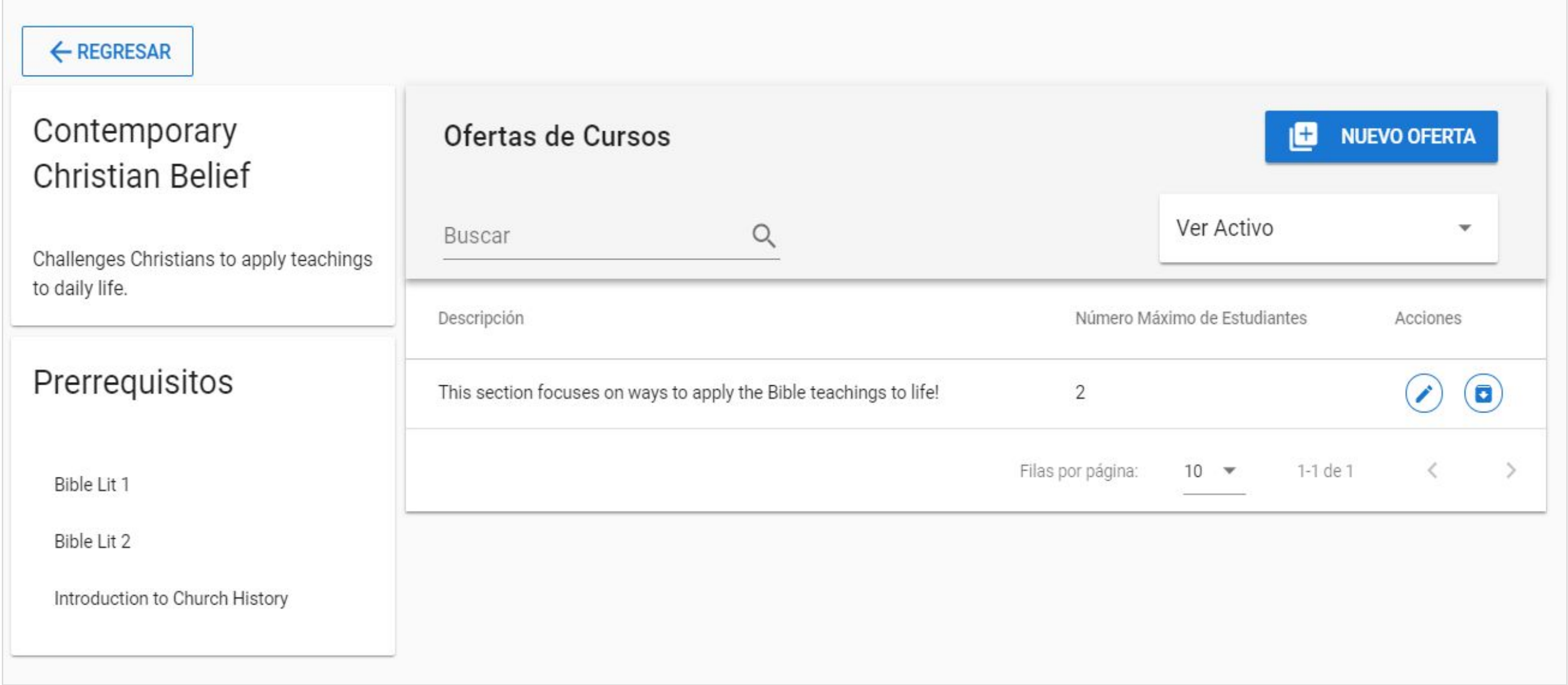
The UI group focused on the Administrative side of course creation. We wanted to design an intuitive interface to create, update, and archive Courses, Course Offerings (sections), and create Class Meetings, which are used to track attendance. We have given teachers the ability to edit their rosters by adding students from existing accounts or confirming self-registered students, and created a transcript page for students to track completed and ongoing diplomas. Our UI team worked closely with the other UI teams to ensure a consistent theme to improve ease of use.



```python

@courses.route('/courses', methods=['POST'])
@jwt_required
# @authorize(["role.superuser", "role.registrar"])
def create_course():
    """ Create an active (by default) course. """
    if 'active' not in request.json:
        request.json['active'] = True
    try:
        valid_course = course_schema.load(request.json)
    except ValidationError as err:
        return jsonify(err.messages), 422

    new_course = Course(**valid_course)
    db.session.add(new_course)
    db.session.commit()
    return jsonify(course_schema.dump(new_course)), 201

```

Database

To store data and create the relationships between tables in the database, we used SQLAlchemy, an Object-Relational Mapper. Upon completing the database, we were tasked with populating the tables with information that would be similar to what would be found in most use-cases. To help better support quality assurance, we created command line tools to ensure some consistent entries across multiple machines or when the tables were dropped and recreated.



← REGRESAR

Contemporary Christian Belief

Challenges Christians to apply teachings to daily life.

Prerrequisitos

- Bible Lit 1
- Bible Lit 2
- Introduction to Church History

Ofertas de Cursos

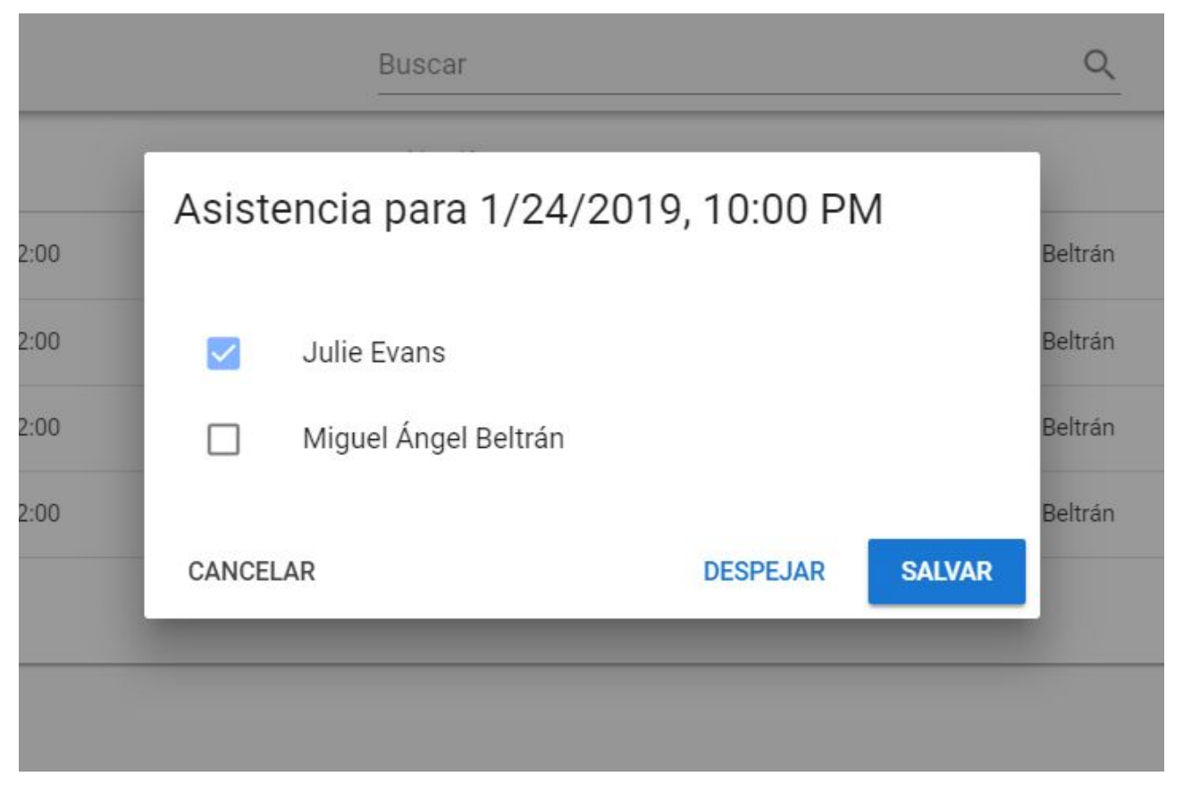
NUEVO OFERTA

Buscar

Ver Activo

Descripción	Número Máximo de Estudiantes	Acciones
This section focuses on ways to apply the Bible teachings to life!	2	

Filas por página: 10 1-1 de 1



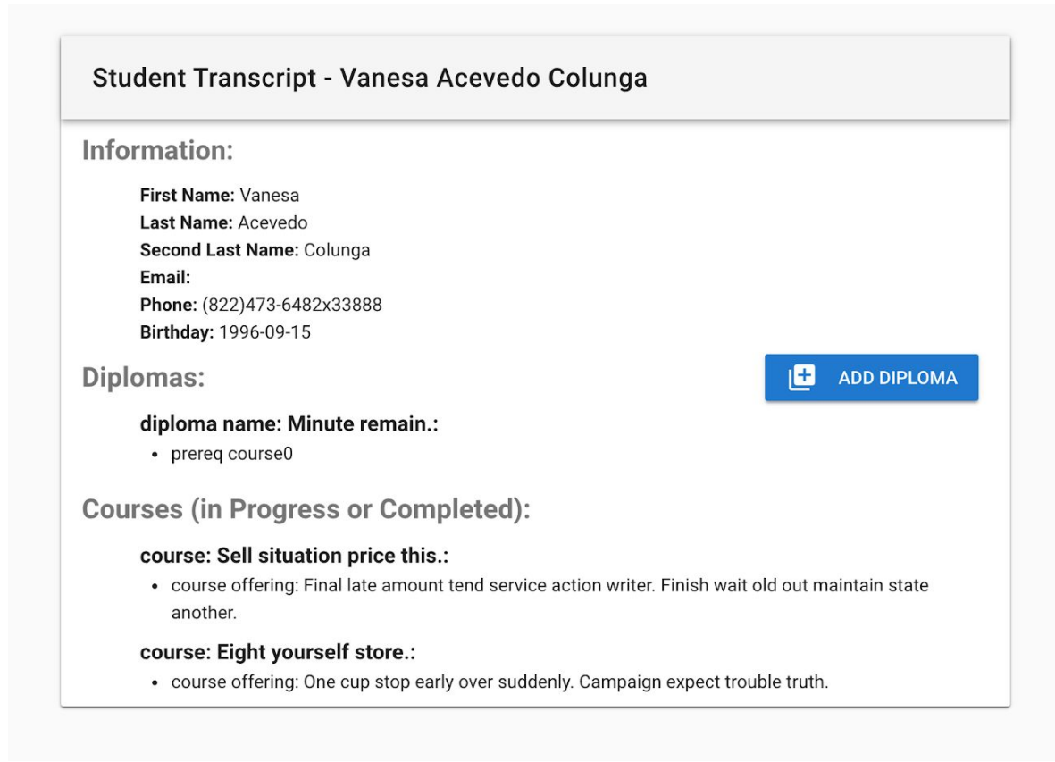
Buscar

Asistencia para 1/24/2019, 10:00 PM

☑ Julie Evans

☐ Miguel Ángel Beltrán

CANCELAR DESPEJAR SALVAR



Student Transcript - Vanesa Acevedo Colunga

Information:

First Name: Vanesa
Last Name: Acevedo
Second Last Name: Colunga
Email:
Phone: (822)473-6482x33888
Birthday: 1996-09-15

Diplomas: ADD DIPLOMA

diploma name: Minute remain.:

- prereq course0

Courses (in Progress or Completed):

course: Sell situation price this.:

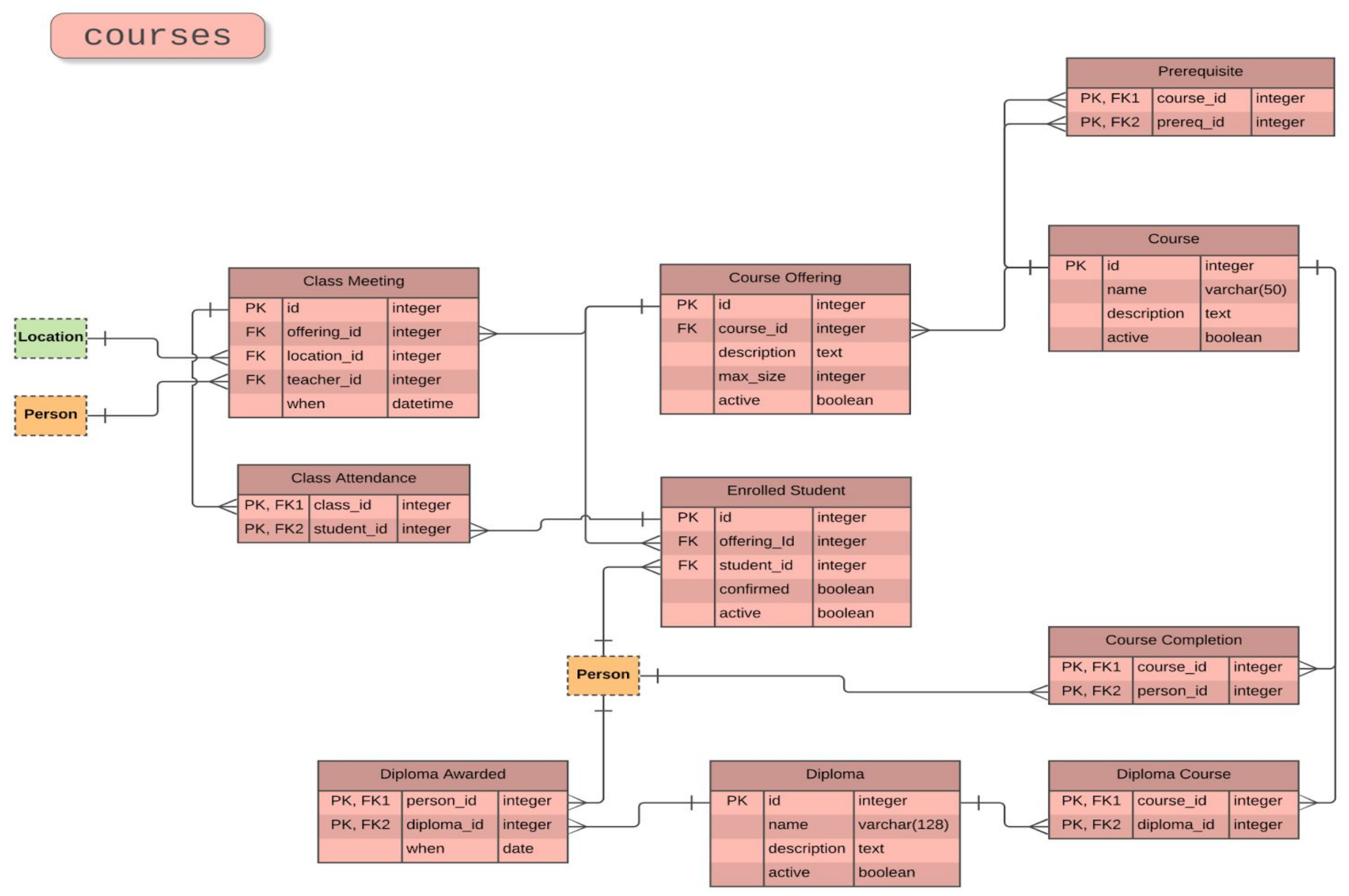
- course offering: Final late amount tend service action writer. Finish wait old out maintain state another.

course: Eight yourself store.:

- course offering: One cup stop early over suddenly. Campaign expect trouble truth.

API

On the API side, we used Python, SQL-Alchemy, and Flask to create the routes that the UI team would call to access information in the database. We were heavily involved in communicating with the database and user interface groups to ensure code quality and code integrity. We met regularly with other API team members to identify and apply the best CRUD practices for our project.



courses

Prerequisite

PK, FK1	course_id	integer
PK, FK2	prereq_id	integer

Course

PK	id	integer
	name	varchar(50)
	description	text
	active	boolean

Class Meeting

PK	id	integer
FK	offering_id	integer
FK	location_id	integer
FK	teacher_id	integer
	when	datetime

Course Offering

PK	id	integer
FK	course_id	integer
	description	text
	max_size	integer
	active	boolean

Class Attendance

PK, FK1	class_id	integer
PK, FK2	student_id	integer

Enrolled Student

PK	id	integer
FK	offering_Id	integer
FK	student_id	integer
	confirmed	boolean
	active	boolean

Course Completion

PK, FK1	course_id	integer
PK, FK2	person_id	integer

Diploma Awarded

PK, FK1	person_id	integer
PK, FK2	diploma_id	integer
	when	date

Diploma

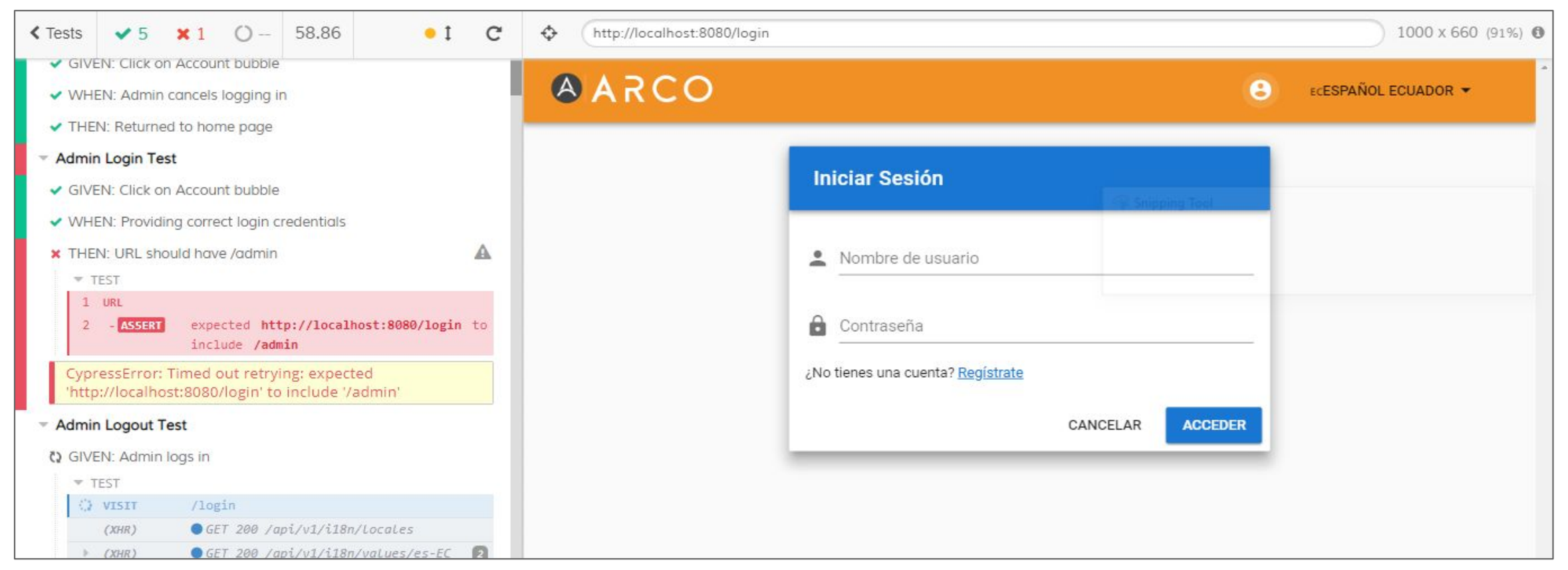
PK	id	integer
	name	varchar(128)
	description	text
	active	boolean

Diploma Course

PK, FK1	course_id	integer
PK, FK2	diploma_id	integer

Testing

For quality assurance testing, we used Pytest for the API endpoints with an extension for coverage testing and Cypress for end-to-end UI testing. QA was tasked with being a resource to the whole team and to confirm all aspects of our code worked exactly as intended.



Tests ✓ 5 ✗ 1 ○ -- 58.86

http://localhost:8080/login 1000 x 660 (91%)

```
✓ GIVEN: Click on Account bubble
✓ WHEN: Admin cancels logging in
✓ THEN: Returned to home page
Admin Login Test
✓ GIVEN: Click on Account bubble
✓ WHEN: Providing correct login credentials
✗ THEN: URL should have /admin
  TEST
  1  URL
  2  - ASSERT  expected http://localhost:8080/login to include /admin
  CypressError: Timed out retrying: expected 'http://localhost:8080/login' to include '/admin'
Admin Logout Test
  GIVEN: Admin logs in
  TEST
  VISIT  /login
  (XHR)  GET 200 /api/v1/i18n/locales
  (XHR)  GET 200 /api/v1/i18n/values/es-EC
```

ARCO ECESPAÑOL ECUADOR

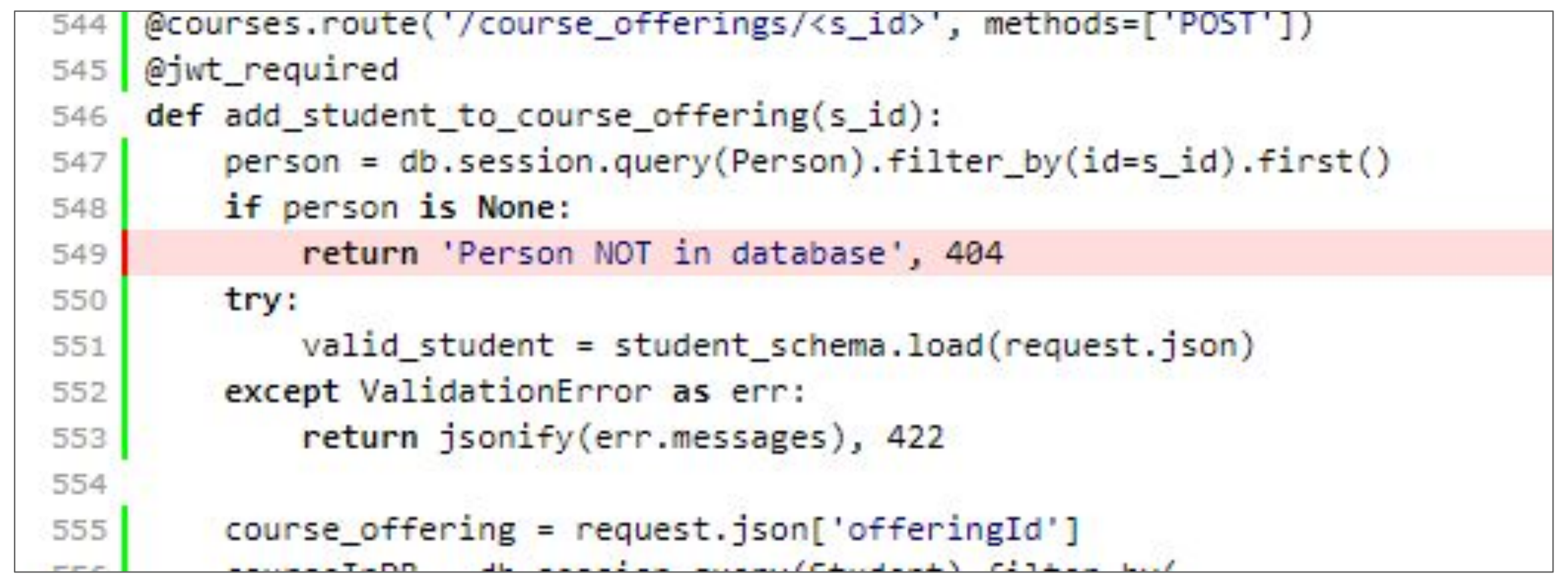
Iniciar Sesión

Nombre de usuario

Contraseña

¿No tienes una cuenta? Regístrate

CANCELAR ACCEDER



```python
@courses.route('/course_offerings/<s_id>', methods=['POST'])
@jwt_required
def add_student_to_course_offering(s_id):
    person = db.session.query(Person).filter_by(id=s_id).first()
    if person is None:
        return 'Person NOT in database', 404
    try:
        valid_student = student_schema.load(request.json)
    except ValidationError as err:
        return jsonify(err.messages), 422

    course_offering = request.json['offeringId']
```

Further Work

We were able to complete a large portion of the courses user stories. There are some UI pages and API endpoints that still need to be completed to add full functionality to the site. Future additions would include a more complete Diplomas page, more information listed on course pages, and the ability to upload images to different course pages.

Acknowledgements

Our team would like to thank Dr. Tom and Mrs. Darci Nurkkala as well as Dr. Ken Kiers for their support throughout this project.

Department of Computer Science & Engineering
Taylor University, Upland, IN

ARCO