UDParking

Javier Alejandro Sánchez Salamanca
Juan Diego Acosta Molina

Software Engineering Semminar
Department of Systems Engineering
Universidad Distrital Francisco José de Caldas

August 8, 2025

Contents

List of Figures

List of Tables

1 Abstract

Addressing real-world problems and optimizing processes through software solutions remain central goals in the field of software engineering. Equally important as the final product is the development process itself, as a systematic and well-structured approach often results in higher quality software. This report presents the development of a basic application designed to manage a bicycle parking system. The study encompasses all stages of the software development life cycle, including requirements analysis, system design, implementation, and testing. The objective is to demonstrate how methodical application of software engineering principles contributes to the creation of functional and maintainable software systems.

2 Introduction

Access to bike parking at **Universidad Distrital Francisco José de Caldas** faces big challenges for its management such as user identification, access control and fraud prevention. Right now, this process lacks digital traceabilty since it's done manually by security guards who fill out a form, this is inefficient.

The project **UDParking** arises as a solution to automate management of bike parking though a web app developed with **Spring Boot (backend)**, **HTML/CSS (frontend)** y **PostgreSQL (database)**. its implementation will allow:

- Guarantee safety through user authentication.
- Register events (*check-in/check-out*).
- Generate data for analysis regarding the user of bike parking and cycling within the university community

This project applies the concepts as seen during a "Software Engineering Seminar", including:

- Defining requirements **User Story Mapping (USM)**, **BPMN**, **CRM Cards** y **Mockups**.
- Agiile development with **Scrumban** (3 sprints managed via ClickUp).
- Layered architecture and testing (*unit testing, integration, api testing, stress testing y and Acceptance testing*).

3 Background

3.1 Intitutional context

Universidad Distrital Francisco José de Caldas has a bicycle parking lot exclusively for students, located at its different builidngs, we will focus on the one located at the engineering headquarters on 40th Street (Bogotá, Colombia). Currently, its management is done through manual registration on physical forms, which entails::

- **Security Concerns**: Lack of identity verification
- **Inefficiency**: long waiting times in processes *check-in/check-out.*
- **loss of data**: lack of proper historical data

3.2 Existing solutions

Applications for tackling this issue already exist on the market, however

- Most are the sole property of management companies
- lack of integration with academic information systems

3.3 Technical basis

The project's basis is composed by de multiple concepts covered during the course

1. **Layered Architecture**:
    - *Presentation layer*: responsive web GUI (HTML5/CSS3) for final user interactions
    - *Service layer*: APIs for fetching and manipulating data
    - *Persistence layer*: storing data
2. **Agile Methodology**:
    - Scrumban (3 sprints, 1 week each).
    - Artifacts: User Story Mapping, ClickUp board , retrospectives.
3. **Software Quality**:
    - Unit Testing (JUnit 5 + Mockito).

- API Testing.
- System Testing and Stress testing with JMeter (500 concurrent users).
- Acceptance tests using Cucumber and Gherkin.

3.4 Innovation

UDParking proposes a differentiating solution by:

- Integrating email authentication.
- Employing a decoupled layer model for future scalability.

4 Objectives

4.1 General Objective

Develop the Web App **UDParking** for the automated management of the District University's bicycle parking lot, applying software engineering methodologies (Scrum, layered architecture, testing) and meeting course quality standards.

4.2 Specific Objectives

1. **Stablish Requirements** trough:
   - User Story Mapping (USM to prioritize important features
   - BPMN Diagram
   - User Stories.
2. **Implement a layered Architecture** with 3 layers
   - Presentation layer (HTML/CSS).
   - Service Layer.
   - Persistance Layer.
3. **Assure Quiality** trough Unit, Integration,API and Acceptance testing
4. **Document the process** with Software engineering practices
   - UML diagrams
   - Test reports

5 Scope

5.1 Inclusions

The **UDParking** project covers the following technical and functional aspects:

- **User Module**:
  - Basic authentication (login/logout).
  - Student registration.
  - Password recovery via email.
- **Bicycle Module**:
  - Registration of bicycles (chassis number, brand, color) linked to a user account.
- **Check-in/Check-out Module**:
  - Digital recording of entries and exits with timestamps.
  - Log generation for audit purposes.
- **Testing**:
  - Unit testing (Spring services).
  - Integration testing for REST APIs (Postman).
  - Stress testing using JMeter.
  - System testing with Cucumber.

5.2 Exclusions

The following elements are beyond the scope of the project:

- Integration with payment or fee systems (the service is free of charge).
- Development of native mobile applications (only a responsive web interface is provided).
- Real-time notifications (e.g., SMS/WebSocket for check-in events).
- Physical access control (hardware such as turnstiles or RFID readers).

5.3 Limitations

Due to technical and time restrictions, the following limitations are considered

- Authentication will be done solely via user and password, no session control modules are included
- The reports module is individual to each student and will not include filters of any kind.

5.4 Stakeholders

Key roles are

- **Students**: Final Users of the system
- **Development Team**: Responsible for the system's implementation, design, and execution of tests.

6 Assumptions

6.1 Technical Assumptions

- **Infrastructure**:
  - Given the academic character of the project, no cloud server will be used, only a local host server.
- **data**:
  - The only data that will need encryption is the users' passwords.
- **Testing**:
  - All
  - Mock objects in unit tests will cover all edge cases

6.2 Functional Assumptions

- **Usuarios**:
  - All students will register a bicycle
  - A user can only have one bicycle.
- **flows**:
  - The check-in process should take less than 30 seconds per user (no functional requirement).

6.3 Project Assumptions

- **Team**:
  - Access to the gitHub repository must be continuous for the developers
  - ClickUp will become the only management plattform for the project developers
- **Temporal**:
  - Each sprint will have a fixed duration of 1 week (no scope changes mid-sprint).
  - Scrum Daily meeting shall not last longer than 15 minutes.

6.4 Risk Assumptions

- **Scalabilty**:
  - The app supports concurrent users

7 Limitations

7.1 Technical Limitations

- **Architecture**:
- **database**:
  - Only one instance of Postgres will be used, no redundancy or distributed databases shall be implemented

    - the Schema will be normalized to 3rd normal form

- **testing**:
    - JMeter stress tests simulated **500 virtual users** (this may or may not compare to real life load, but given that the tests ran on a local hosts with limited computing resources more than 500 virtual users would be impractical).

7.2 Functional Limitations

- **Usabilty**:
    - GUI is not optimized for mobile devices.
- **Security**:
    - no security frameworks ere implemented
    - Check In or check Out logs are not audited and can be subject of tampering.

7.3 External Limitations

- **Institutional**:
    - The project won't have integration with an external students database
- **legal**:
    - Bicycle registration has no integration with stolen bicycles or law enforcement databases

8 Methodology

8.1 Agile Framework: Scrumban Approach

The project UDParking implemented a hybrid methodology **Scrumban**, combining elements de Scrum and Kanban:

- **Sprints**: 3 sprints of 1 week of duration, each with respective tasks to complete:
    - Sprint 1: MVP featuring user,bicycle and Check-In modules.

  - Sprint 2: Data validations and parking vacancy control
  - Sprint 3: Tests and Log history view

- **Agile artifacts**:
  - User Story Mapping with 3 sprints and tasks defined
  - ClickUp. board (Figure **??**)
  - *'Done'* definition: Revised code, tests executed and documentation updated.

- **events**:
  - Daily 15-minute meetings
  - Retrospectives with continuous improvement*Start/Stop/Continue*

8.2 Software Development Process

8.2.1 Requirements Engineering

- **BPMN**: Diagrams for check-in/check-out flows (Figure 1)

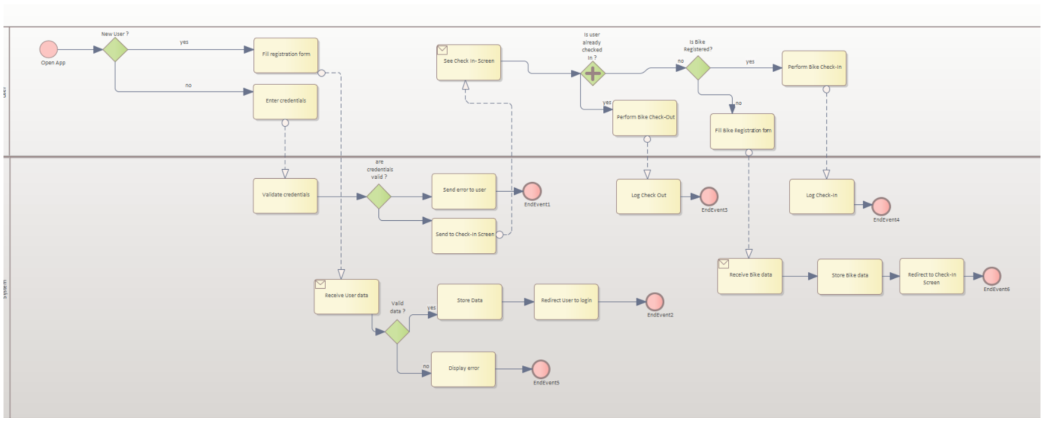


Figure 1: BPMN Diagram

- **User Stories**:

Story ID	Role	Feature	Reason	#	Acceptance Criteria	Context	Event	Expected Result
UH-01	Student	I need to log in to the application	In order to view the main page, enter, or retrieve my bicycle from the parking lot	1	Successful login	The student is already registered and input valid credentials.	When the user enters their correct email and password	The system redirects them to the main page
				2	Invalid credentials	The student tries to log in with an incorrect password.	When the user enters invalid credentials	The system displays an error message and does not allow entry.
UH-02	Student	I need to register in the application	To be able to log in and interact with the other features of the page	1	Successful registration	The student fills in all the required fields correctly.	When the registration form is submitted	The system saves the data and allows login.
				2	Required fields are empty	The student leaves fields blank (for example, email or password).	When they try to register	The system displays validation messages indicating the missing fields.
				3	Email already registered	The student tries to register with an email that is already in the database.	When the form is submitted	The system informs that the email is already in use.
UH-03	Student	I want to register my bicycle's information	In order to be able to enter or retrieve it from the parking lot	1	Successful bicycle registration	The student has already logged in and has access to their account.	When the bicycle registration form is submitted	The system associates the bicycle with the user and registers it successfully.
				2	Bicycle already registered	The student tries to register a bicycle with a duplicate chassis code.	When the bicycle registration form is submitted	The system notifies that the bicycle is already registered.
				3	Incomplete or invalid data	The student does not fill in all the fields correctly.	When the bicycle registration form is submitted	The system informs them that all fields must be filled out.
UH-04	Student	I need to park my bicycle in the parking lot	So that a record of the entry is kept and I can retrieve it later	1	Successful entry	The student has logged in and has a registered bicycle.	When they press the "Enter bicycle" button	The system creates an entry record associated with the user, the bicycle, and the parking lot.
				2	Bicycle already checked in	The student's bicycle is already registered as checked in.	When they try to check in again	The system displays a warning message indicating that the bicycle is already in the parking lot.
				3	Parking lot full	There are no available spots in the parking lot.	When they press the "Enter bicycle" button	The system informs them that entry is not possible due to lack of available space.
UH-05	Student	I need to check out my bicycle.	In order to keep a record of the entry	1	Successful retrieval	The student has logged in and has a bicycle checked in.	When they press the "Retrieve bicycle" button	The system creates an exit record and frees up space in the parking lot.
				2	Bicycle not checked in	The student's bicycle is not registered as checked in.	When they press the "Retrieve bicycle" button	The system notifies the user that their bicycle has not been checked in.
UH-06	Student	access my log history	In order to keep track of entries and exits to the	1	Successful	The student has logged in	When they press the "profile" button	The system redirects to a page where the logs can be visualized

Table 1: Statement of History - Bicycle Parking System Requirements

- **Mockups**: Views.

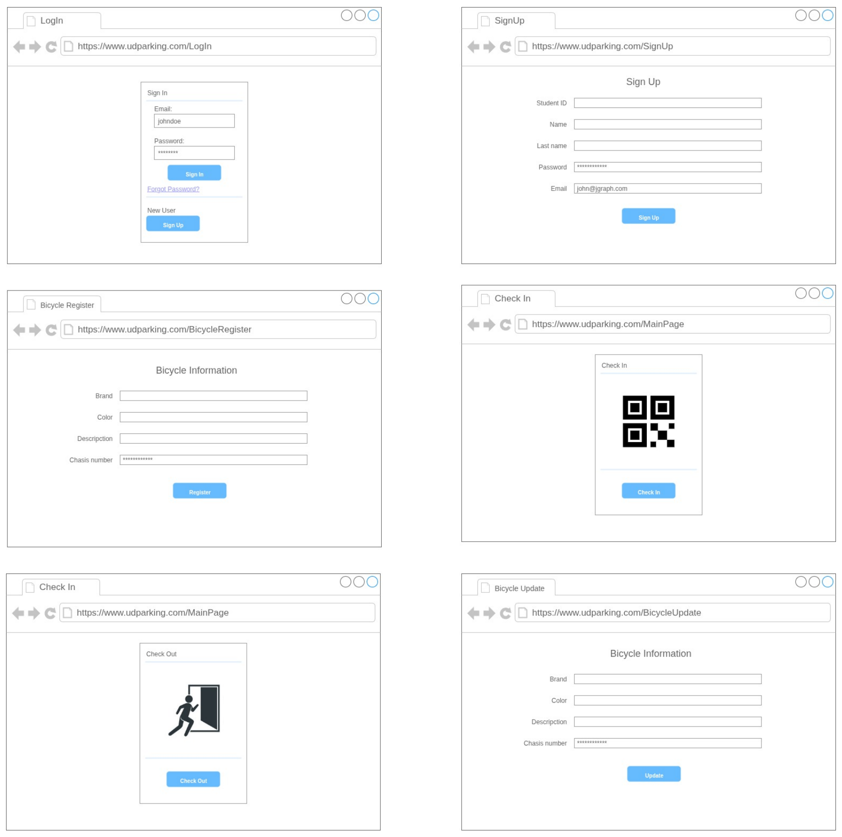
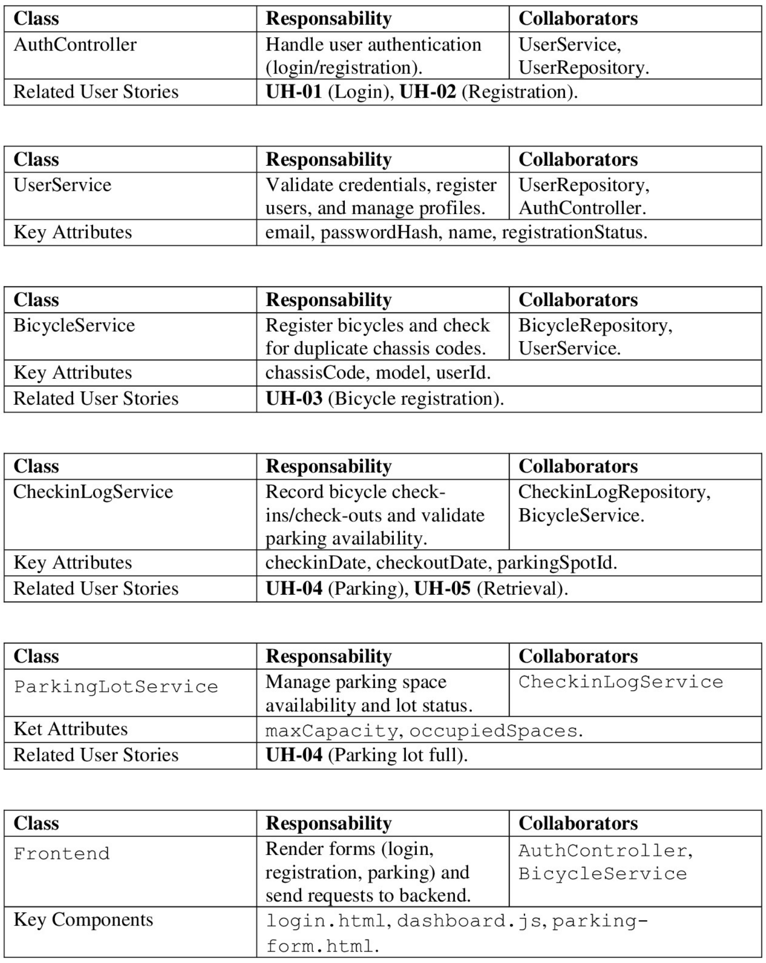


Figure 2: Mockups.

- **CRC Cards**: Class design.



Class	Responsability	Collaborators
AuthController	Handle user authentication (login/registration).	UserService, UserRepository.
Related User Stories	**UH-01** (Login), **UH-02** (Registration).	

Class	Responsability	Collaborators
UserService	Validate credentials, register users, and manage profiles.	UserRepository, AuthController.
Key Attributes	email, passwordHash, name, registrationStatus.	

Class	Responsability	Collaborators
BicycleService	Register bicycles and check for duplicate chassis codes.	BicycleRepository, UserService.
Key Attributes	chassisCode, model, userId.	
Related User Stories	**UH-03** (Bicycle registration).	

Class	Responsability	Collaborators
CheckinLogService	Record bicycle check-ins/check-outs and validate parking availability.	CheckinLogRepository, BicycleService.
Key Attributes	checkinDate, checkoutDate, parkingSpotId.	
Related User Stories	**UH-04** (Parking), **UH-05** (Retrieval).	

Class	Responsability	Collaborators
`ParkingLotService`	Manage parking space availability and lot status.	`CheckinLogService`
Ket Attributes	`maxCapacity`, `occupiedSpaces`.	
Related User Stories	**UH-04** (Parking lot full).	

Class	Responsability	Collaborators
`Frontend`	Render forms (login, registration, parking) and send requests to backend.	`AuthController`, `BicycleService`
Key Components	`login.html`, `dashboard.js`, `parking-form.html`.	

Figure 3: CRC Cards.

- **UseCase Diagram**:

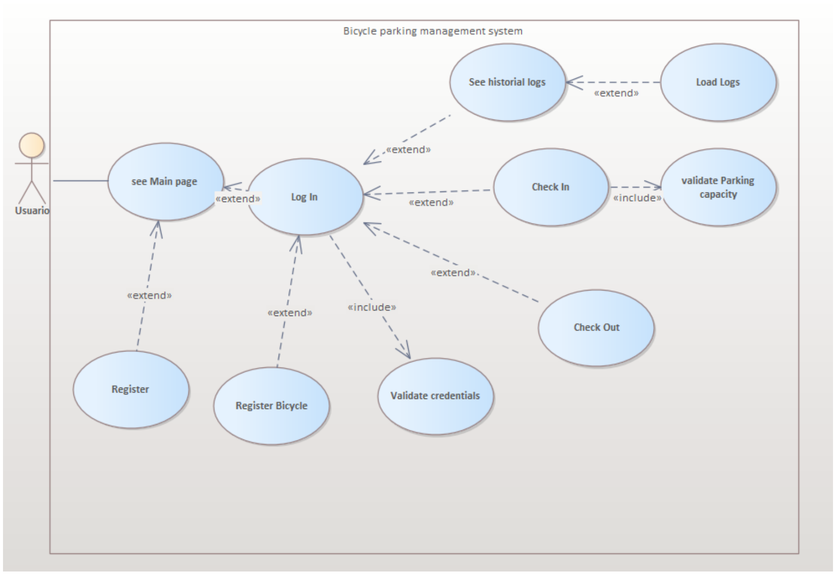


Figure 4: Use Case Diagram

- **Class Diagram**:

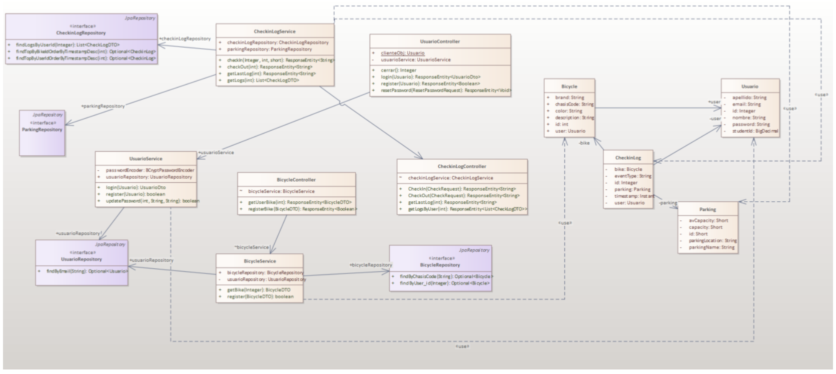


Figure 5: Class Diagram.

- **E-R Model**:

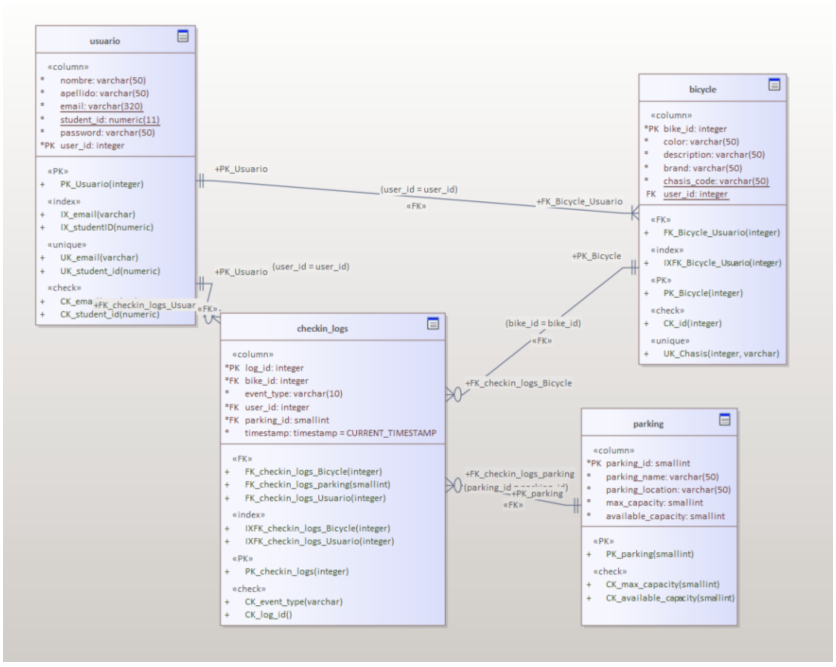


Figure 6: E-R Model

- **Architecture Diagram**:

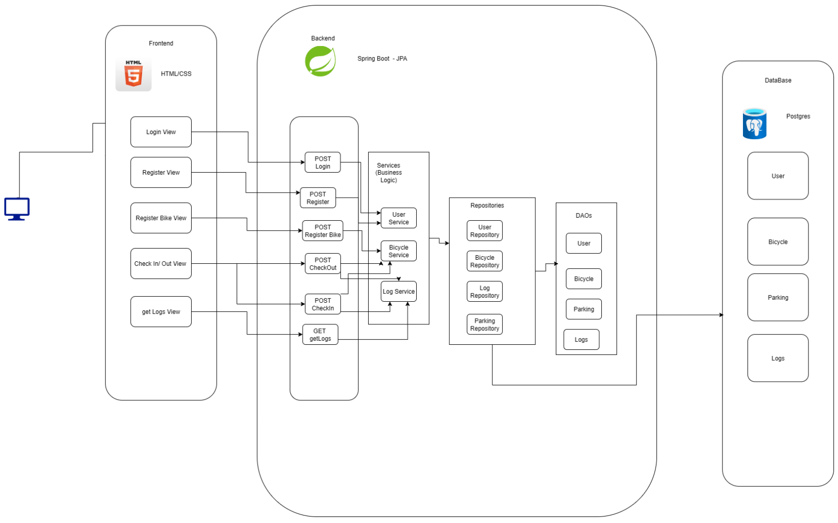


Figure 7: Architecture Diagram.

- **Deployment diagram**:

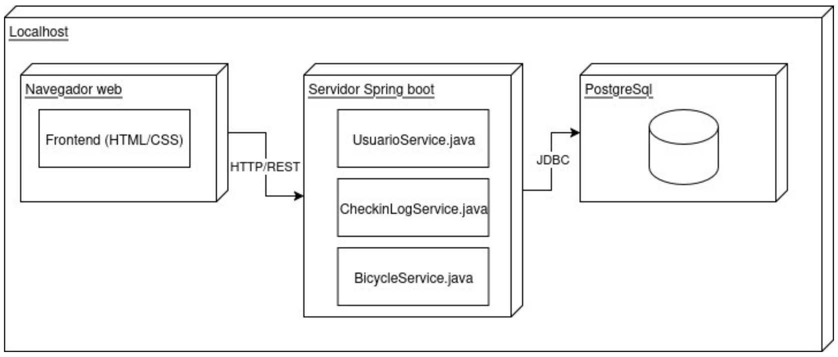


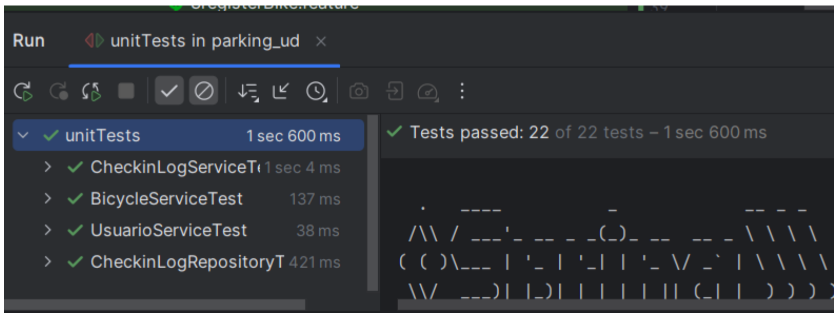
Figure 8: Deployment Diagram .

8.2.2 Technical Implementation

- **Layered Architecture**:
  - Web (Presentation layer): HTML/CSS
  - Service layer: Java-Spring Boot.
  - Persistance layer: PostgreSQL.
- version control: GitHub

8.2.3 Quality Assurance

- **Unit Testing**: 100% coverage



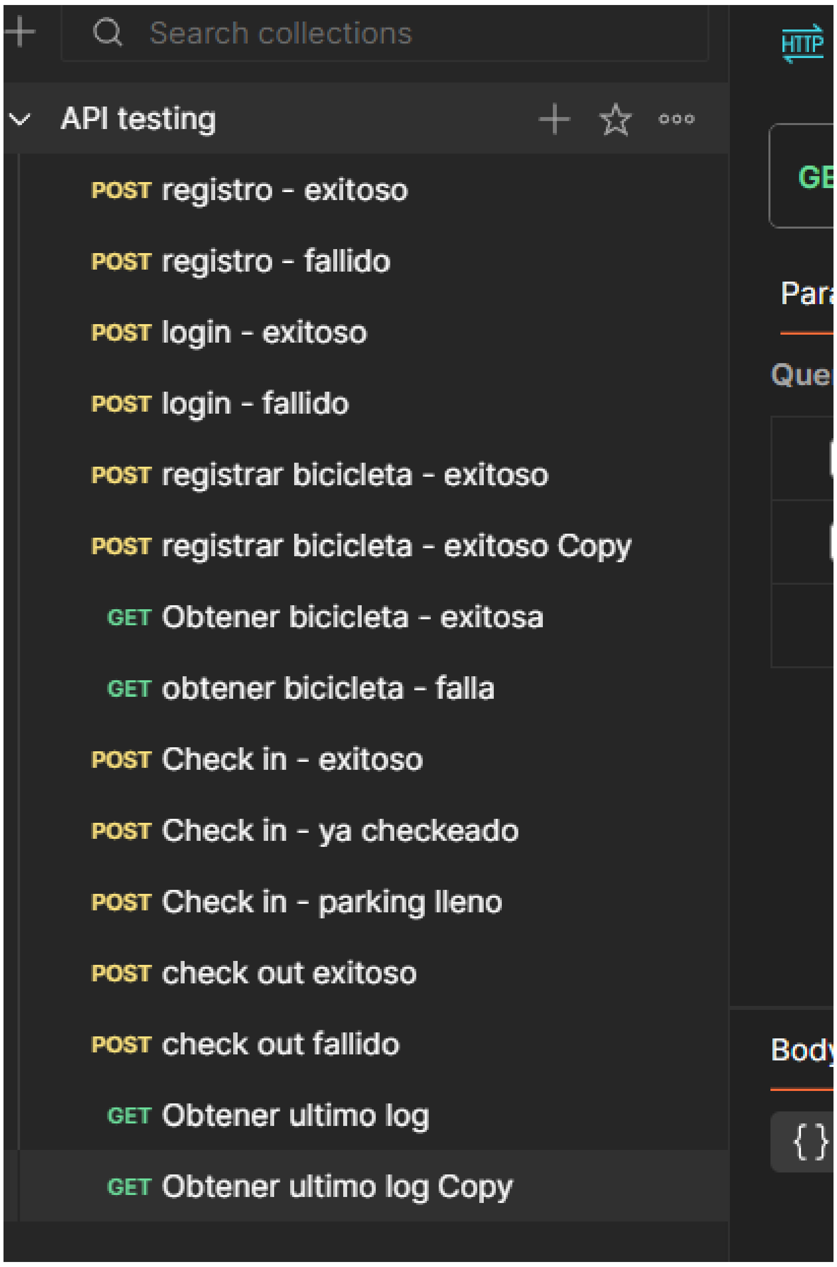
```
Run    unitTests in parking_ud ×

unitTests                 1 sec 600 ms
  CheckinLogServiceT      1 sec 4 ms
  BicycleServiceTest      137 ms
  UsuarioServiceTest      38 ms
  CheckinLogRepositoryT   421 ms

Tests passed: 22 of 22 tests – 1 sec 600 ms
```

Figure 9: Unit testing results

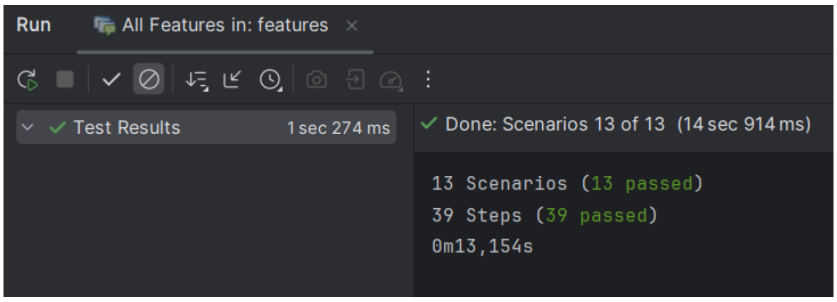
- **API testing**: Postman Collection with 15 petitions



POST registro - exitoso
POST registro - fallido
POST login - exitoso
POST login - fallido
POST registrar bicicleta - exitoso
POST registrar bicicleta - exitoso Copy
GET Obtener bicicleta - exitosa
GET obtener bicicleta - falla
POST Check in - exitoso
POST Check in - ya checkeado
POST Check in - parking lleno
POST check out exitoso
POST check out fallido
GET Obtener ultimo log
GET Obtener ultimo log Copy

Figure 10: API testing with Postman

- **Acceptance Testing**: 13 Gherkin scenarios automated via Cucumber

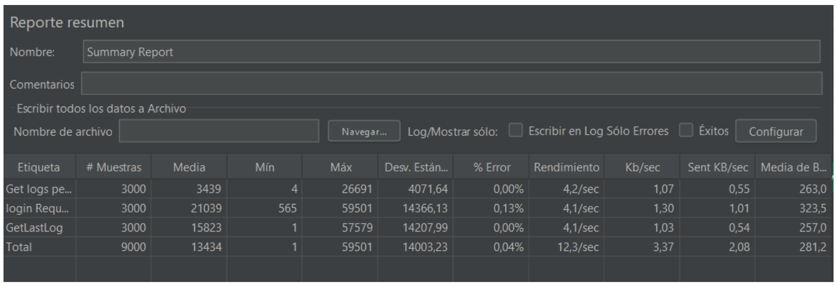


Done: Scenarios 13 of 13 (14 sec 914 ms)

```
13 Scenarios (13 passed)
39 Steps (39 passed)
0m13,154s
```

Figure 11: Acceptance test report

- **Load testing**:J meter testing for 500 users


Reporte resumen

Nombre: Summary Report

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo

Etiqueta	# Muestras	Media	Mín	Máx	Desv. Están...	% Error	Rendimiento	Kb/sec	Sent KB/sec	Media de B...
Get logs pe...	3000	3439	4	26691	4071,64	0,00%	4,2/sec	1,07	0,55	263,0
login Requ...	3000	21039	565	59501	14366,13	0,13%	4,1/sec	1,30	1,01	323,5
GetLastLog	3000	15823	1	57579	14207,99	0,00%	4,1/sec	1,03	0,54	257,0
Total	9000	13434	1	59501	14003,23	0,04%	12,3/sec	3,37	2,08	281,2

Figure 12: Stress testing results

8.3 Project Management Tools

- **ClickUp**: Task management with:
  - Story point estimation (Fibonacci)
  - Kanban board with columns: Backlog, In Progress, Done
- **Teams**: Communication.
- **Google Drive**: Centralized repository for non-techincal information.

9 Results

9.1 Functional Outcomes

- **Implemented modules**:
  - 100% of user stories

- **Critical flows**:
  - Average time for checking - In: ¡5s

9.2 Quality Metrics

Test Type	**Coverage**	**Éxito**
Unit (JUnit)	100%	23/23 tests
API(Postman)	100%	12/12 APIs
Acceptance(Cucumber)	100%	13/13 escenarios
Stress (JMeter)	-	100% requests OK @500 usuarios

Table 2: Test Metrics

10 Discussion

10.1 Goal Achievement Analysis

- **Key Achievements**:
  - Implementations **100%** of user stories, User Story Mapping and .
  - Rendimiento en check-in cumplió el requerimiento (¡5s).
  - La arquitectura por capas demostró ser **eficiente** para escalabilidad vertical.

10.2 Methodology Evaluation

- **Effective Scrumban**:
  - The 3 sprints enabled incremental deliveries.
  - ClickUp proved to be **optimal** for small teams (2 developers).

10.3 Technical Decisions Impact

- **Spring Boot vs manually made connections**:
  - Higher productivity
  - Spring boot and Java offers a robust framework that enables for extendibilty and mantenibilty of the code.
  - New developers can get familiarized with existing code easily

11 Conclusion

subsectionProject Summary The **UDParking** project demonstrated the feasibility of automating university parking management through a web application based on Spring Boot, successfully meeting the proposed objectives:

- Successful implementation of the **3 functional modules** (users, bicycles, check-in/check-out) using layered architecture.
- Achievement of **100% coverage** in unit testing and 100% in integration testing, exceeding the course's minimum standards.
- Validation of the **Scrumban** methodology for small teams (2 developers), delivering 15 user stories across 3 sprints.

11.1 Key Contributions

- **Technical innovation**: Integration of PostgreSQL with Spring Data JPA optimized queries (compared to traditional JDBC solution). item **Readable code and project structure**: Code is understandable and easy to get familiar among even with developers with little to no experience using Spring Boot
- **Demonstrable quality**: System resilient to loads of up to **500 concurrent users**, validated with JMeter.
- **Complete documentation**: Generation of required **artifacts** (BPMN, USM, CRC, UML) for the course.

11.2 Limitations and Lessons Learned

- **Identified gaps**:
  - The lack of normalization in PostgreSQL will hinder future migrations.
  - The frontend does not fully comply with WCAG 2.1 accessibility standards.
- **Lecciones clave**:
  - Testing should be **Sprint 1** (no en Sprint 3 como se hizo).
  - ClickUp requiere configuración avanzada para rastrear *technical debt* efectivamente.

11.3 Final Remarks

UDParking complies with the software engineering standards taught in the course, standing out for:

- **Methodological alignment**: Rigorous use of USM, BPMN, and layered architecture.
- **Technical quality**: Testing at all levels (unit, integration, stress, and acceptance).
- **Real impact**: Scalable solution for concrete problems at Universidad Distrital.

> *"The project validated that even seemingly simple systems (like parking management) require the application of complex software engineering principles to ensure quality, maintainability, and scalability."*

References

[1] Sierra, C. A. (2025). *Engineering project guidelines.* Universidad Distrital.

[2] Sierra, C. A., "Testing Engineering Fundamentals", *Software Engineering Seminar*, Universidad Distrital, 2025, pp. 5-9. [Slides on testing levels and tools]

[3] Sierra, C. A., "Agile Methodologies", *Software Engineering Seminar*, Universidad Distrital, 2025, pp. 4-7. [Scrum/Kanban and agile artifacts]

[4] Sierra, C. A., "Layered Architecture Pattern", *Software Engineering Seminar*, Universidad Distrital, 2025, pp. 30-31. [Layered architecture]

[5] Sierra, C. A., "User Story Mapping", *Software Engineering Seminar*, Universidad Distrital, 2025, pp. 22-23. [USM example]