**Guide2. APT Project Development**

**Capstone Subject**

|  |
| --- |
| **1. Summary of APT Project Progress** |
| Below, you will find different fields that you must fill in with the requested information. |

|  |  |
| --- | --- |
| APT Project Progress Summary | The **TechApps - SCAV** (Vehicle Access Control System) project has advanced considerably, reaching the **Sprint** of **week 10**. The main objective is to optimize the management of vehicular access in the Vista Parque condominium, streamlining the vehicular flow and improving safety.  To date, the following milestones have been completed:   * Implementation of the patent reader using Python, which has already been presented as a working prototype. * Development of the REST API in Spring Boot, which allows integration with the database and vehicle validation through an external API. * Creating the MER for the database in SQL Server. * Development of the mobile application in Flutter with a focus on the administration and resident module. |
| Objectives | Project Objectives  General Objective: To develop a comprehensive vehicular access control system that improves security and speeds up the entry and exit of vehicles in the Vista Parque condominium, ensuring the exclusive control of residents and authorized personnel.  Specific Objectives:   1. Implement an automatic vehicle license plate reader using Python. 2. Develop a mobile app in Flutter so residents can register visits in advance. 3. Create a REST API in Spring Boot for vehicular access permission management and integration with the external API. 4. Design a SQL Server database that supports resident, visitation, and commercial vehicle management. 5. Generate reports of vehicular access and monitoring in real time. 6. Ensure that the system complies with computer security standards. |
| Methodology | The methodology followed is **Scrum**, with 9 two-week sprints. Activities include requirements planning, modular development (patent reader, API, mobile app, database), integration testing, and future system commissioning. |
| Evidence of progress | Patent Reader Prototype: A functional prototype implemented in Python was presented, demonstrating the system's ability to detect license plates and associate them with registered vehicles.  Database Design: MER designed in SQL Server.  Development of the REST API: The API is partially implemented and connected to the database, allowing vehicle validation and access to information on residents, visits and logs. |

|  |
| --- |
| **2. Monitoring of the Work Plan** |
| Carefully examine your work plan, focusing especially on the progress status and adjustments column. |

Competence or units of competences

* Manage the configuration of environments, application services, and databases (Competency 1):

The development environment for the database has been configured in PostgreSQL and the REST API in Spring Boot. The database is connected to the different modules of the system to ensure continuity of operations.

This has been worked on from the Week 5 Sprint to the Week 8 Sprint, when the database design and API integration was finished.

* Offer proposals for an IT solution by analysing the processes (Competence 2):

From the first phases, the requirements were raised to identify the problems of the Vista Parque condominium and propose a system that efficiently manages vehicular access.

This was mainly implemented in the initial project planning and during the Week 1 Sprint and Week 3 Sprint with the definition of the system architecture.

* Develop a software solution using systematized techniques (Competence 3):

As of week 10, multiple system modules have been developed, including the patent reader in Python, the REST API in Spring Boot, and the launch of the mobile app in Flutter. All following an agile methodology (Scrum) that systematizes the development process.

This competition has been active during the Sprints of weeks 5-10, with clear advances in the technical implementation of the software.

* Building data models (Competence 4):

Work was done on the MER database in PostgreSQL, which manages resident, visitation, and vehicle records, ensuring that it is scalable for future expansions.

This was completed during the Week 7 Sprint.

* Schedule queries or routines to manipulate information (Competence 5):

The REST API has been implemented that allows queries to be made to the database on residents and visitors, manipulating the information in real time.

This has been worked on during the week 7-10 Sprints, when part of the development of the API and its connection to the database was completed.

* Implement comprehensive systemic solutions (Competence 6):

The license plate reader, database, and REST API are integrated to automate the vehicular access process, streamlining condominium operations.

This has been part of the development from the Week 5 Sprint onwards, with substantial progress in the Week 8-10 Sprints.

Work Plan and Monitoring

Competencies or Units:

* Project Management: Planning of sprints, taking requirements and creating the backlog.
* Software Development: Implementation of the patent reader, development of the REST API, database and mobile application.
* Software Quality: Security, functionality, and integration testing.

Activities and Material Progress Status:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Activity | State | Difficulties | Physical Materials/Resources | Languages/Environment |
| Patent Reader | Completed | Difficulties in doing live tests with the camera on the street due to weather conditions and traffic | Video Camera, PC | Python, OpenCV |
| Database Design | Completed | N/A | PC, Google Cloud SQL | PostgreSQL |
| API REST | Completed | N/A | PC, GCP (Google Cloud Platform) API server | Java, Spring Boot |
| Mobile App in Flutter | Completed | N/A | PC, Smartphone for Testing | Flutter, Dart |
| Integration Testing | Not started | Coordination for real-time camera testing in a controlled environment | Video camera, PC, local network for integration | Python, Java, SQL, Flutter |

Future Activities

* Perform integration tests between modules.
* Prepare the system for commissioning and training end users.

### 

### Adjustment:

### **Methodology: Correcting Scrum**

From the beginning of the TechApps - SCAV project, the Scrum methodology was proposed and established as the guide to manage the development. However, during the first phases of implementation, due to a team error, activities began to be managed under a more traditional approach. This deviation was due to the organization of tasks as independent blocks, with fixed deadlines and little interaction between the modules under development (patent reader, REST API, database and mobile application).

Upon identifying this error, an immediate adjustment was made to return to the agile Scrum approach, which made it possible to:

1. Recover iterative and incremental planning: Work has been organized into sprints, ensuring greater flexibility to incorporate changes as needed.
2. Better coordination and collaboration: Daily meetings were reintroduced to align the team and resolve blockages quickly.
3. Frequent partial deliveries: Incremental deliveries were resumed in each sprint, which allowed functional advances, such as the patent reader and the integration of the API with the database, to be presented.
4. Continuous feedback: Constant interaction with those involved is encouraged to ensure that each module will be developed according to the expectations and requirements of the project.

This adjustment has been crucial to recover the agile dynamic that had been lost. The team aligned again with the Scrum methodology, allowing for better synchronization between modules, optimizing development times, and ensuring that the project progresses as planned.

### **Project Schedule: Sprint Correction**

### Initially, the TechApps - SCAV project was organized into a different number of sprints, which did not allow adequate planning for the implementation of all modules. After a review of the schedule, a correction was made in the number and duration of the sprints, in order to optimize the development and meet the objectives set.

### New Schedule:

### Total, of Sprints: 9 Sprints.

### Duration of each Sprint: 2 weeks each.

### Distribution by Modules:

### Project Management and Backlog: Sprint 1 (S1) and (S2), Sprint 2 (S3) and (S4).

### Development of the Patent Recognition System: Sprint 3 (S5) and (S6).

### Development of the Database Model: Sprint 4 (S7) and (S7).

### REST API implementation: Sprint 5 (S9) and (S10).

### Mobile App on Flutter: Adjusted for two full sprints: Sprint 6 (S11) and (S11), Sprint 7 (S13) and (S14).

### Quality and Integration Testing: Sprint 8 (S15) and (S16).

### White March and Training: Sprint 9 (S17) and (S18).

### Impact of the Adjustment:

### Time Redistribution: Dates have been adjusted so that the most critical modules have more time allocated, such as the development of the mobile application in Flutter, which will be developed over 2 full sprints.

### Flow Optimization: The rest of the modules will be developed in single sprints to avoid work fragmentation.

1. **Application Development:**

Unification in a Single App with Administrative and Resident Roles

During the initial planning phase of the TechApps - SCAV project, it was proposed to develop two independent applications: one for administrators and one for residents. However, as we progressed through development and after a flow and usability review, we identified that the split was creating redundancy and increasing the maintenance burden on the system.

Change Decision: Unification in a Single Application with Dynamic Roles

Instead of managing two separate apps, we decided to integrate all the functionalities into a single mobile app. The new strategy simplifies development, eases maintenance, and improves the user experience by unifying operations on a single platform.

How the new structure works:

* Upon login, the app detects the account type (Administrator or Resident) based on the user's credentials.
* Depending on the user's role, they are referred to different screens and functionalities:
  + Administrators:
    - Management of residents, visitors and vehicles.
    - Access to reports and real-time monitoring of vehicular accesses.
    - Entry and exit permit management.
    - Automation and management of fines.
  + Residents:
    - Advance registration of visits.
    - Real-time notifications about the arrival of visitors.
    - Display of personal access history.

Advantages of the Unified Approach

1. Complexity Reduction: By unifying both applications into one, redundancies are eliminated and system administration is simplified.
2. Role Flexibility: The app now dynamically adapts to the user's role, providing an intuitive and centralized flow.
3. Simplified Maintenance: Makes it easier to manage updates, as any improvements are reflected in a single app.
4. Improved User Experience: Users (administrators and residents) can access all its functionalities from a single platform without having to switch applications.
5. **Cloud Services Implementation and Database Migration:**

Context of the Change: During the initial phases of the TechApps - SCAV project, the system was configured to operate in a local environment, which generated certain limitations in terms of access and scalability. To improve performance and ensure more robust and secure access, it was decided to migrate the application and database to a cloud environment using Google Cloud Platform (GCP).

Implementation Details:

1. Database Migration:
   * The database originally developed in SQL Server was migrated to the cloud using the NeonTech tool, which gives you a cloud instance on AWS to host databases of up to 512mb for free.
   * This migration allows for faster, more reliable, and centralized access to resident, visitation, and vehicle information from anywhere.
   * Advantages:
     + Eliminates dependency on a localhost environment, improving availability and accessibility.
     + Increase security by leveraging security and encryption policies.
     + It makes it easy to scale the database according to the demand of the system.
2. Deploying all microservices on GCP:
   * The four microservices that make up SCAV have so far been deployed in the cloud using Google Kubernetes Engine (GKE).
   * This allows for an automated and scalable deployment of the authentication system, which is crucial for managing access for both administrators and residents.
   * Advantages:
     + It improves the scalability and availability of the authentication and management service for vehicles, residents and visitors.
     + Facilitates load management based on the number of concurrent users.
     + Reduces downtime by enabling hot updates without impacting users.

Impact of these changes on the project

1. Elimination of Local Dependencies:
   * With cloud migration, the need to run services on localhost is eliminated, simplifying application integration and deployment in production environments.

1. **Using PostgreSQL for Database Management**

Context of the Change: To optimize the storage, querying, and scalability of system data, the decision was made to implement PostgreSQL as the primary database engine. This has allowed for a more modern and efficient approach to data management of the TechApps - SCAV project.

* Implementation Details:

1. Migration to PostgreSQL in Neon Tech: The database has been migrated to PostgreSQL, taking advantage of Neon Tech to ensure fast and secure access. PostgreSQL was selected for its ability to handle large volumes of data and its support for advanced queries and ACID transactions.

* High Availability: Cloud configuration ensures continuous service availability, even in the event of hardware failures.
* Continuous Deployment and Automatic Backup: Enables automated backups, facilitating disaster recovery and non-disruptive upgrades.
* Enhanced Security: Supports encryption of data in transit and at rest, ensuring the confidentiality of information.

Impact of these changes on the project

* Elimination of Local Dependencies:

Migrating to PostgreSQL in the Cloud eliminates the need to run databases on-premises, simplifying deployment and integration into production.

* Improved Scalability and Availability:

The system can now dynamically scale based on load, which is crucial for a project that optimizes vehicular access in a large condominium like Vista Parque.

* Facilitates Collaboration and Development:

By centralizing the database in the cloud, all team members can access the development and production environment remotely, streamlining testing and collaborative development.

|  |
| --- |
| **3. Adjustments from monitoring** |
| Dig deeper into the observations in your work plan. It analyses the planned activities and points out which aspects facilitated or hindered the execution of the plan. Discuss how you addressed and/or will address the obstacles. Finally, point out the adjustments you made to the work plan based on this analysis. |

|  |
| --- |
| Factors that have facilitated the development of the work plan:   * Team coordination: The implementation of daily meetings using the Scrum methodology has allowed for better communication and quick problem solving. * Appropriate technological tools: We have a well-configured development environment, including Python, SQL Server and Spring Boot, which has made the developers' work easier. * Early functional prototype: The patent reader was completed in the early stages of the project, allowing rapid progress in integration with other modules.   Factors that have hindered the development of the work plan:   1. Change in the structure of the applications:    * Difficulty: Initially, a single application was developed for residents and administration, which complicated management and increased the workload.    * Action taken: The development was divided into two applications: one administrative and one for residents, prioritizing at this time the administrative one. 2. Error in the application of the Scrum methodology:    * Difficulty: Although it was planned to work with Scrum, in the first few weeks a traditional approach was mistakenly adopted, affecting planning and deliverables.    * Action taken: An immediate correction was made to resume the Scrum methodology, establishing 2-week sprints with daily meetings and incremental deliverables. 3. Adjustment in the sprint schedule:    * Difficulty: The initial planning of the sprints did not allocate enough time for some key modules, such as the application in Flutter.    * Action taken: Tasks were redistributed into 9 sprints, giving 2 full sprints for development in Flutter and adjusting the dates of the rest of the modules. |

|  |
| --- |
| Adjusted or deleted activities Adjustments made:   1. **Unification of Applications for Administration and Residents:**    * **Reason**: Originally, it was planned to develop two independent applications (one for **Administration** and one for **Residents**). However, as we progressed in development, we noticed that keeping two separate applications led to redundancy and complexity in management.    * **Adjustment**: The functionalities were unified in a **single mobile application**. Depending on the **user's role** (Administrator or Resident), when logging in, the app redirects to specific screens and functionalities based on the profile.    * **Impact**: Unification simplifies maintenance and upgrades, improves user experience, and reduces development burden In addition, the system can now scale more efficiently by centralizing all functionalities on a single platform.      1. **Backend Migration to the Cloud with PostgreSQL**   **Reason: Initially, the backend (REST API in Spring Boot) was configured to run on localhost, which limited system access and scalability.**  **Fit: The REST API developed in Spring Boot was migrated to Google Cloud Platform (GCP), using Neon Tech with PostgreSQL for database management, and orchestrating the backend with Google Kubernetes Engine (GKE).**  **Impact:**   * + **Elimination of localhost usage: The system now connects to the backend and database via cloud services, improving availability and reliability.**   + **Scalability and availability: Kubernetes allows you to automatically scale your API and database based on load, ensuring optimal performance.**   + **Simplified maintenance: The use of containers and Kubernetes, along with the database in PostgreSQL, allows continuous updates without interrupting the service, ensuring that the system is always available.**  1. Adjustment in the sprint schedule:    * Reason: The initial planning allocated one sprint per module, which did not provide adequate time for the development of some more complex components, such as the mobile app in Flutter.    * Adjustment: A redistribution of the schedule was made, increasing the time for the development of the application in Flutter to two sprints and maintaining one sprint for the simplest modules. This new structure ensures better planning and avoids delays.   Deleted activities: At the moment, no activities have been deleted, as all the initially defined phases are still relevant to achieve the project objectives.  Justification:  These adjustments were made to ensure the efficiency of the development and improve the quality of the system, allowing the established deadlines to be met without compromising key functionalities. Separating applications and reorganizing the schedule have been critical to keeping the project flowing according to expectations. |

|  |
| --- |
| Activities that you have not started or are delayed: In case you have not started activities or they are delayed according to plan, indicate the reasons why you have not been able to meet these deadlines and what strategies you would use to advance in these activities and not affect your APT project.  3. White March and Staff Training:   * Current status: Although significant progress has been made on the core modules, the white march and training have not yet begun because they are dependent on the completion of integration testing. * Strategy: Staff training is scheduled to take place in parallel with the latest tests of the system. This will allow personnel to be prepared when the system goes into operation. The white march will start once it is validated that the modules interact correctly. |