**Guía2. Desarrollo Proyecto APT**

**Asignatura Capstone**

|  |
| --- |
| **1. Resumen avance Proyecto APT** |
| A continuación, encontrarás distintos campos que deberás completar con la información solicitada. |

|  |  |
| --- | --- |
| Resumen de avance proyecto APT | El proyecto **TechApps - SCAV** (Sistema de Control de Acceso Vehicular) ha avanzado considerablemente, alcanzando el **Sprint** de la **semana 10**. El objetivo principal es optimizar la gestión del acceso vehicular en el condominio Vista Parque, agilizando el flujo vehicular y mejorando la seguridad.  Hasta la fecha, se han completado los siguientes hitos:   * Implementación del lector de patentes utilizando Python, lo cual ya se presentó como un prototipo funcional. * Desarrollo de la API REST en Spring Boot, que permite la integración con la base de datos y la validación vehicular mediante una API externa. * Creación del MER para la base de datos en SQL Server. * Desarrollo parcial de la aplicación móvil en Flutter con enfoque en el módulo de administración. |
| Objetivos | Objetivos del Proyecto  Objetivo General: Desarrollar un sistema integral de control de acceso vehicular que permita mejorar la seguridad y agilizar el ingreso y salida de vehículos en el condominio Vista Parque, asegurando el control exclusivo de residentes y personal autorizado.  Objetivos Específicos:   1. Implementar un lector automático de patentes vehiculares utilizando Python. 2. Desarrollar una aplicación móvil en Flutter para que los residentes puedan registrar visitas anticipadamente. 3. Crear una API REST en Spring Boot para la gestión de permisos de acceso vehicular e integración con la API externa. 4. Diseñar una base de datos en SQL Server que soporte la gestión de residentes, visitas y vehículos comerciales. 5. Generar reportes de acceso vehicular y monitoreo en tiempo real. 6. Asegurar que el sistema cumpla con estándares de seguridad informática. |
| Metodología | La metodología seguida es **Scrum**, con 9 sprints de dos semanas. Las actividades incluyen la planificación de requisitos, desarrollo por módulos (lector de patentes, API, aplicación móvil, base de datos), pruebas de integración, y la futura puesta en marcha del sistema. |
| Evidencias de avance | Prototipo del Lector de Patentes: Se presentó un prototipo funcional implementado en Python, demostrando la capacidad del sistema para detectar patentes y asociarlas a vehículos registrados.  Diseño de la Base de Datos: MER diseñado en SQL Server.  Desarrollo de la API REST: La API está parcialmente implementada y conectada a la base de datos, permitiendo la validación vehicular y acceso a información de residentes, visitas y bitácoras. |

|  |
| --- |
| **2. Monitoreo del Plan de Trabajo** |
| Examina cuidadosamente tu plan de trabajo, enfocándote especialmente en la columna de estado de avance y ajustes. |

Competencia o unidades de competencias

* Administrar la configuración de ambientes, servicios de aplicaciones y bases de datos (Competencia 1):

Se ha configurado el entorno de desarrollo para la base de datos en SQL Server y la API REST en Spring Boot. La base de datos está conectada con los diferentes módulos del sistema para asegurar la continuidad de las operaciones.

Esto se ha trabajado desde el Sprint de la semana 5 hasta el Sprint de la semana 8, cuando se terminó el diseño de la base de datos y la integración de la API.

* Ofrecer propuestas de solución informática analizando los procesos (Competencia 2):

Desde las primeras fases, se realizó el levantamiento de requerimientos para identificar las problemáticas del condominio Vista Parque y proponer un sistema que gestione de manera eficiente el acceso vehicular.

Esto se implementó principalmente en la planificación inicial del proyecto y durante el Sprint de la semana 1 y Sprint de la semana 3 con la definición de la arquitectura del sistema.

* Desarrollar una solución de software utilizando técnicas sistematizadas (Competencia 3):

Hasta la semana 10, se han desarrollado múltiples módulos del sistema, incluyendo el lector de patentes en Python, la API REST en Spring Boot, y el inicio de la aplicación móvil en Flutter. Todo siguiendo una metodología ágil (Scrum) que sistematiza el proceso de desarrollo.

Esta competencia ha estado activa durante los Sprints de las semanas 5-10, con avances claros en la implementación técnica del software.

* Construir modelos de datos (Competencia 4):

Se trabajó en el MER de la base de datos en SQL Server, que gestiona los registros de residentes, visitas y vehículos, asegurando que sea escalable para futuras expansiones.

Esto se completó durante el Sprint de la semana 7.

* Programar consultas o rutinas para manipular información (Competencia 5):

Se ha implementado la API REST que permite realizar consultas a la base de datos sobre residentes y visitas, manipulando la información en tiempo real.

Esto ha sido trabajado durante los Sprints de las semana 7-10, cuando se completó parte del desarrollo de la API y su conexión con la base de datos.

* Implementar soluciones sistémicas integrales (Competencia 6):

El lector de patentes, la base de datos y la API REST están integrados para automatizar el proceso de acceso vehicular, lo que optimiza las operaciones del condominio.

Esto ha sido parte del desarrollo desde el Sprint de la semana 5 en adelante, con avances sustanciales en los Sprints de las semana 8-10.

Plan de Trabajo y Monitoreo

Competencias o Unidades:

* Gestión de Proyecto: Planificación de los sprints, toma de requisitos y creación del backlog.
* Desarrollo de Software: Implementación del lector de patentes, desarrollo de la API REST, base de datos y aplicación móvil.
* Calidad de Software: Pruebas de seguridad, funcionalidad e integración.

Actividades y Estado de Avance Materiales:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Actividad | Estado | Dificultades | Materiales/Recursos Físicos | Lenguajes/Entorno |
| Lector de Patentes | Completado | Dificultades para hacer pruebas en vivo con la cámara en la calle debido a condiciones climáticas y tráfico | Cámara de video, PC | Python, OpenCV |
| Diseño de Base de Datos | Completado | N/A | PC, Google Cloud SQL | Kubernetes |
| API REST | Completado | N/A | PC, GCP (Google Cloud Platform) API server | Java, Spring Boot |
| Aplicación Móvil en Flutter | Completado | N/A | PC, smartphone para pruebas | Flutter, Dart |
| Pruebas de Integración | No iniciado | Coordinación para pruebas con cámara en tiempo real en un entorno controlado | Cámara de video, PC, red local para la integración | Python, Java, SQL, Flutter |

Actividades Futuras

* Realizar pruebas de integración entre los módulos.
* Preparar el sistema para la puesta en marcha y capacitación a usuarios finales.

### 

### Ajuste:

### **Metodología: Corrección hacia Scrum**

Desde el inicio del proyecto TechApps - SCAV, la metodología Scrum fue la propuesta y establecida como la guía para gestionar el desarrollo. Sin embargo, durante las primeras fases de implementación, por un error del equipo, las actividades comenzaron a gestionarse bajo un enfoque más tradicional. Este desvío se debió a la organización de tareas como bloques independientes, con fechas de entrega fijas y poca interacción entre los módulos en desarrollo (lector de patentes, API REST, base de datos y aplicación móvil).

Al identificar este error, se realizó un ajuste inmediato para volver al enfoque ágil Scrum, lo cual permitió:

1. Recuperar la planificación iterativa e incremental: Se organizaron los trabajos en los sprints, lo que garantiza mayor flexibilidad para incorporar cambios según las necesidades.
2. Mejor coordinación y colaboración: Las reuniones diarias (daily meetings) se reintrodujeron para alinear al equipo y resolver bloqueos rápidamente.
3. Entregas parciales frecuentes: Se retomaron las entregas incrementales en cada sprint, lo que permitió presentar avances funcionales, como el lector de patentes y la integración de la API con la base de datos.
4. Retroalimentación continua: Se fomenta la interacción constante con los involucrados para garantizar que cada módulo se desarrollará conforme a las expectativas y requerimientos del proyecto.

Este ajuste ha sido crucial para recuperar la dinámica ágil que se había perdido. El equipo se alineó nuevamente con la metodología Scrum, permitiendo una mejor sincronización entre los módulos, optimizando los tiempos de desarrollo y asegurando que el proyecto avance según lo planeado.

### **cronograma del Proyecto: Corrección de Sprints**

### Inicialmente, el proyecto TechApps - SCAV se organizó en un número diferente de sprints, lo cual no permitía una planificación adecuada para la implementación de todos los módulos. Tras una revisión del cronograma, se realizó una corrección en la cantidad y duración de los sprints, con el fin de optimizar el desarrollo y cumplir con los objetivos planteados.

### Nuevo Cronograma:

### Total, de Sprints: 9 Sprints.

### Duración de cada Sprint: 2 semanas cada uno.

### Distribución por Módulos:

### Gestión de Proyecto y Backlog: Sprint 1 (S1) y (S2), Sprint 2 (S3) y (S4).

### Desarrollo del Sistema de Reconocimiento de Patentes: Sprint 3 (S5) y (S6).

### Desarrollo del Modelo de Base de Datos: Sprint 4 (S7) y (S7).

### Implementación de API REST: Sprint 5 (S9) y (S10).

### Aplicación Móvil en Flutter: Se ajustó a dos sprints completos: Sprint 6 (S11) y (S11), Sprint 7 (S13) y (S14).

### Pruebas de Calidad e Integración: Sprint 8 (S15) y (S16).

### Marcha Blanca y Capacitación: Sprint 9 (S17) y (S18).

### Impacto del Ajuste:

### Redistribución del Tiempo: Las fechas se ajustaron para que los módulos más críticos tengan más tiempo asignado, como el desarrollo de la aplicación móvil en Flutter, que se desarrollará a lo largo de 2 sprints completos.

### Optimización del Flujo: El resto de los módulos se desarrollarán en sprints únicos para evitar la fragmentación del trabajo.

1. **Desarrollo de Aplicaciones:**

Unificación en una Única App con Roles Administrativos y Residentes

Durante la fase inicial de planificación del proyecto TechApps - SCAV, se propuso desarrollar dos aplicaciones independientes: una para administradores y otra para residentes. Sin embargo, a medida que avanzamos en el desarrollo y tras una revisión del flujo y la usabilidad, identificamos que la división generaba redundancia y aumentaba la carga de mantenimiento del sistema.

Decisión de Cambio: Unificación en una Única Aplicación con Roles Dinámicos

En lugar de gestionar dos aplicaciones separadas, decidimos integrar todas las funcionalidades en una única aplicación móvil. La nueva estrategia simplifica el desarrollo, facilita el mantenimiento y mejora la experiencia del usuario al unificar las operaciones en una sola plataforma.

Cómo funciona la nueva estructura:

* Al iniciar sesión, la aplicación detecta el tipo de cuenta (Administrador o Residente) basado en las credenciales del usuario.
* Dependiendo del rol del usuario, se derivan a diferentes pantallas y funcionalidades:
  + Administradores:
    - Gestión de residentes, visitas y vehículos comerciales.
    - Acceso a reportes y monitoreo en tiempo real de accesos vehiculares.
    - Administración de permisos de entrada y salida.
    - Automatización y gestión de multas.
  + Residentes:
    - Registro anticipado de visitas.
    - Notificaciones en tiempo real sobre la llegada de visitas y entregas.
    - Visualización del historial de accesos personales.
    - Solicitud y autorización de acceso para servicios comerciales.

Ventajas del Enfoque Unificado

1. Reducción de Complejidad: Al unificar ambas aplicaciones en una sola, se eliminan redundancias y se simplifica la administración del sistema.
2. Flexibilidad en el Uso de Roles: La aplicación ahora se adapta dinámicamente al rol del usuario, proporcionando un flujo intuitivo y centralizado.
3. Mantenimiento Simplificado: Facilita la gestión de actualizaciones, ya que cualquier mejora se refleja en una sola app.
4. Experiencia del Usuario Mejorada: Los usuarios (administradores y residentes) pueden acceder a todas sus funcionalidades desde una única plataforma sin tener que cambiar de aplicación.
5. **Implementación de Servicios en la Nube y Migración de la Base de Datos:**

Contexto del Cambio: Durante las fases iniciales del proyecto TechApps - SCAV, el sistema estaba configurado para operar en un entorno local, lo que generaba ciertas limitaciones en cuanto al acceso y escalabilidad. Para mejorar el rendimiento y garantizar un acceso más robusto y seguro, se decidió migrar la aplicación y la base de datos a un entorno en la nube utilizando Google Cloud Platform (GCP).

Detalles de la Implementación:

1. Migración de la Base de Datos:
   * La base de datos originalmente desarrollada en SQL Server fue migrada a un servicio en la nube utilizando Google Cloud SQL.
   * Esta migración permite un acceso más rápido y confiable a la información de residentes, visitas y vehículos comerciales desde cualquier lugar.
   * Ventajas:
     + Elimina la dependencia de un entorno localhost, mejorando la disponibilidad y accesibilidad.
     + Aumenta la seguridad al aprovechar las políticas de seguridad y encriptación de GCP.
     + Facilita el escalado de la base de datos según la demanda del sistema.
2. Despliegue del Servicio de Autenticación (Microservicio msautenticar):
   * El primer microservicio (msautenticar) ha sido desplegado en la nube usando Google Kubernetes Engine (GKE).
   * Esto permite un despliegue automatizado y escalable del sistema de autenticación, que es crucial para gestionar los accesos tanto de administradores como de residentes.
   * Ventajas:
     + Mejora la escalabilidad y disponibilidad del servicio de autenticación.
     + Facilita la gestión de cargas en función del número de usuarios concurrentes.
     + Reduce el tiempo de inactividad al permitir actualizaciones en caliente sin afectar a los usuarios.

Impacto de estos Cambios en el Proyecto

1. Eliminación de Dependencias Locales:
   * Con la migración a la nube, se elimina la necesidad de ejecutar servicios en localhost, lo que simplifica la integración y despliegue de la aplicación en entornos de producción.

1. **Uso de Kubernetes para la Gestión de Contenedores:**

Contexto del Cambio: Para optimizar la orquestación y la escalabilidad del sistema, se tomó la decisión de implementar Kubernetes utilizando Google Kubernetes Engine (GKE). Esto ha permitido un enfoque más moderno y eficiente en el despliegue de los servicios del proyecto TechApps - SCAV.

Detalles de la Implementación:

1. Orquestación de Contenedores con Kubernetes:
   * Todos los microservicios del sistema, como el lector de patentes, la API REST, y el sistema de notificaciones están ahora contenedorizados utilizando Docker y gestionados por Kubernetes.
   * Kubernetes permite la automatización del despliegue, escalado y gestión de estos contenedores, garantizando que el sistema funcione de manera fluida bajo diferentes niveles de carga.
2. Ventajas del Uso de Kubernetes:
   * Escalabilidad Automática: Kubernetes ajusta automáticamente la cantidad de réplicas de cada microservicio en función de la demanda.
   * Despliegue Continuo: Facilita la implementación continua (CI/CD), permitiendo realizar actualizaciones sin interrupciones.
   * Alta Disponibilidad: Gracias a su capacidad para reubicar servicios fallidos y gestionar la disponibilidad de los recursos, el sistema se mantiene operativo incluso ante fallos en algunos nodos.

Impacto de estos Cambios en el Proyecto

1. Eliminación de Dependencias Locales:

* Con la migración a la nube y el uso de Kubernetes, se elimina la necesidad de ejecutar servicios en localhost, lo que simplifica la integración y despliegue de la aplicación en entornos de producción.

1. Mejora en la Escalabilidad y Disponibilidad:

* Ahora el sistema puede escalar dinámicamente en función de la demanda, lo que es esencial para un proyecto que busca optimizar el acceso vehicular en un condominio de gran tamaño como Vista Parque.

1. Facilita la Colaboración y el Desarrollo:

* Al centralizar los servicios en la nube, todos los miembros del equipo pueden acceder al entorno de desarrollo y producción de manera remota, agilizando las pruebas y el desarrollo colaborativo.

|  |
| --- |
| **3. Ajustes a partir del monitoreo** |
| Profundiza en las observaciones de tu plan de trabajo. Analiza las actividades planificadas y señala qué aspectos facilitaron u obstaculizaron la ejecución del plan. Plantea cómo abordaste y/o abordarás los obstáculos. Por último, señala los ajustes que realizaste al plan de trabajo a partir de este análisis. |

|  |
| --- |
| Factores que han facilitado el desarrollo del plan de trabajo:   * Coordinación del equipo: La implementación de reuniones diarias mediante la metodología Scrum ha permitido una mejor comunicación y resolución rápida de problemas. * Herramientas tecnológicas adecuadas: Contamos con un entorno de desarrollo bien configurado, incluyendo Python, SQL Server y Spring Boot, lo que ha facilitado el trabajo de los desarrolladores. * Prototipo funcional temprano: El lector de patentes fue completado en las primeras fases del proyecto, lo que permitió avanzar rápidamente en la integración con otros módulos.   Factores que han dificultado el desarrollo del plan de trabajo:   1. Cambio en la estructura de las aplicaciones:    * Dificultad: Inicialmente, se desarrolló una única aplicación para residentes y administración, lo que complicó la gestión y aumentó la carga de trabajo.    * Acción tomada: Se dividió el desarrollo en dos aplicaciones: una administrativa y otra para residentes, priorizando en este momento la administrativa. 2. Error en la aplicación de la metodología Scrum:    * Dificultad: Aunque se planificó trabajar con Scrum, en las primeras semanas se adoptó por error un enfoque tradicional, afectando la planificación y las entregas.    * Acción tomada: Se realizó una corrección inmediata para retomar la metodología Scrum, estableciendo sprints de 2 semanas con reuniones diarias y entregas incrementales. 3. Ajuste en el cronograma de sprints:    * Dificultad: La planificación inicial de los sprints no asignaba suficiente tiempo para algunos módulos clave, como la aplicación en Flutter.    * Acción tomada: Se redistribuyeron las tareas en 9 sprints, dando 2 sprints completos para el desarrollo en Flutter y ajustando las fechas del resto de los módulos. |

|  |
| --- |
| Actividades ajustadas o eliminadas Ajustes realizados:   1. **Unificación de Aplicaciones para Administración y Residentes:**    * **Motivo**: Originalmente, se planificó desarrollar dos aplicaciones independientes (una para **Administración** y otra para **Residentes**). Sin embargo, al avanzar en el desarrollo, notamos que mantener dos aplicaciones separadas generaba redundancia y complejidad en la gestión.    * **Ajuste**: Se unificaron las funcionalidades en una **única aplicación móvil**. Dependiendo del **rol del usuario** (Administrador o Residente), al iniciar sesión, la aplicación redirige a las pantallas y funcionalidades específicas según el perfil.    * **Impacto**: La unificación simplifica el mantenimiento y actualizaciones, mejora la experiencia del usuario y reduce la carga de desarrollo. Además, ahora el sistema puede escalar de manera más eficiente al centralizar todas las funcionalidades en una sola plataforma.      1. **Migración del Backend a la Nube con Kubernetes:**     * **Motivo**: Inicialmente, el backend (API REST en Spring Boot) estaba configurado para ejecutarse en **localhost**, lo que limitaba el acceso y la escalabilidad del sistema.    * **Ajuste**: Se migró la **API REST desarrollada en Spring Boot** a **Google Cloud Platform (GCP)**, utilizando **Google Kubernetes Engine (GKE)** para su orquestación.    * **Impacto**:      + **Eliminación del uso de localhost**: Ahora, el sistema se conecta al backend a través de un **servicio en la nube**, lo que mejora la disponibilidad y confiabilidad.      + **Escalabilidad y disponibilidad**: Kubernetes permite **escalar automáticamente** la API en función de la carga de usuarios, asegurando un rendimiento óptimo en todo momento.      + **Mantenimiento simplificado**: El uso de contenedores y Kubernetes permite realizar **actualizaciones continuas** sin interrumpir el servicio, garantizando que el sistema esté siempre disponible para los usuarios. 2. Ajuste en el cronograma de sprints:    * Motivo: La planificación inicial asignaba un sprint por módulo, lo que no proporcionaba el tiempo adecuado para el desarrollo de algunos componentes más complejos, como la aplicación móvil en Flutter.    * Ajuste: Se realizó una redistribución del cronograma, incrementando a dos sprints el tiempo para el desarrollo de la aplicación en Flutter y manteniendo un sprint para los módulos más simples. Esta nueva estructura asegura una mejor planificación y evita retrasos.   Actividades eliminadas: Por el momento, no se han eliminado actividades, ya que todas las fases definidas inicialmente siguen siendo relevantes para alcanzar los objetivos del proyecto.  Justificación:  Estos ajustes se realizaron para asegurar la eficiencia del desarrollo y mejorar la calidad del sistema, permitiendo cumplir con los plazos establecidos sin comprometer las funcionalidades clave. La separación de las aplicaciones y la reorganización del cronograma han sido fundamentales para mantener el flujo del proyecto de acuerdo con las expectativas. |

|  |
| --- |
| Actividades que no has iniciado o están retrasadas: En caso de que no hayas iniciado actividades o estén retrasadas de acuerdo con la planificación, señala los motivos por los que no has podido cumplir dichos plazos y qué estrategias utilizarías para avanzar en dichas actividades y no afectar tu proyecto APT.  3. Marcha Blanca y Capacitación del Personal:   * Estado actual: Aunque se ha avanzado significativamente en los módulos principales, la marcha blanca y la capacitación aún no han comenzado porque dependen de la finalización de las pruebas de integración. * Estrategia: La capacitación del personal está programada para llevarse a cabo en paralelo con las últimas pruebas del sistema. Esto permitirá que el personal esté preparado cuando el sistema entre en operación. La marcha blanca se iniciará una vez que se valide que los módulos interactúan de forma correcta. |