**UNIVERSIDAD TECNOLÓGICA DEL PERÚ**  
**Programa Expedition Trainee – 2024**

**Reto Técnico - Expedition Trainee UTP - Desafío Frontend**

**Proyecto: Desarrollo de Ventanas para Matrícula en Línea**

**Postulante:**

* **Nombre Completo:** Victor Raul Herrera Castillo
* **Correo Electrónico:** vr.herrera.c@gmail.com
* **Celular:** 944073494
* **Fecha de Presentación:** 23/09/2024
* **Perfil:** Desarrollador Frontend con experiencia en React JS, diseño de interfaces UX/UI y capacidad de adaptación a las necesidades de plataformas con metodologías agiles.

**Objetivo del Proyecto:**  
Diseñar una plataforma responsiva para la matrícula en línea que facilite el proceso de selección de cursos y secciones, mejorando la experiencia de los estudiantes a través de una interfaz intuitiva y funcional.

**Descripción del Proyecto:**

Este proyecto consiste en la implementación de ventanas que permitan a los estudiantes visualizar sus cursos matriculados, seleccionar secciones, gestionar cruces de horarios, y confirmar su matrícula de manera eficiente, utilizando tecnologías como React JS y siguiendo las mejores prácticas de UX/UI.

Contenido de la documentación

[Capítulo I: Introducción 4](#_Toc177977587)

[1.1. Descripción del Proyecto 4](#_Toc177977588)

[1.2. Descripción del Postulante 4](#_Toc177977589)

[1.3. Perfil de la Solución 4](#_Toc177977590)

[1.4. Segmento Objetivo 4](#_Toc177977591)

[Capítulo II: Requirements Elicitation & Analysis 5](#_Toc177977592)

[2.1. Introducción 5](#_Toc177977593)

[2.2. Needfinding (Identificación de Necesidades) 5](#_Toc177977594)

[2.3. Ubiquitous Language (Lenguaje Ubicuo) 6](#_Toc177977595)

[2.4. Análisis de Requisitos 6](#_Toc177977596)

[2.4.1. Requisitos Funcionales 6](#_Toc177977597)

[2.4.2. Requisitos No Funcionales 7](#_Toc177977598)

[Capítulo III: Requirements Specification 7](#_Toc177977599)

[3.1. Introducción 7](#_Toc177977600)

[3.2. User Stories (Historias de Usuario) 7](#_Toc177977601)

[3.2.1. Historia de Usuario 1: Selección de Cursos 7](#_Toc177977602)

[3.2.2. Historia de Usuario 2: Selección de Secciones 8](#_Toc177977603)

[3.2.3. Historia de Usuario 3: Confirmación de Matrícula 8](#_Toc177977604)

[3.2.4. Historia de Usuario 4: Visualización de Horario 8](#_Toc177977605)

[3.3. Product Backlog 9](#_Toc177977606)

[3.4. Prioridades y Gestión de Requisitos 10](#_Toc177977607)

[Capítulo IV: Product Design 10](#_Toc177977608)

[4.1. Introducción 10](#_Toc177977609)

[4.2. General Style Guidelines (Pautas Generales de Estilo) 10](#_Toc177977610)

[4.3. Information Architecture (Arquitectura de la Información) 11](#_Toc177977611)

[4.4. Web Applications UX/UI Design 12](#_Toc177977612)

[4.5. Web Applications Wireframes (Esquemáticos de Aplicaciones Web) 12](#_Toc177977613)

[4.6. Web Applications Wireflow Diagrams (Diagrama de Flujo de Interacción de las Aplicaciones Web) 13](#_Toc177977614)

[4.7. Web Applications Mock-ups (Maquetas de Aplicaciones Web) 13](#_Toc177977615)

[4.8. Web Applications User Flow Diagrams (Diagramas de Flujo del Usuario) 13](#_Toc177977616)

[4.9. Domain-Driven Software Architecture (Arquitectura de Software Basada en el Dominio) 13](#_Toc177977617)

[4.10. Software Architecture Context Diagram (Diagrama de Contexto de Arquitectura de Software) 13](#_Toc177977618)

[4.11. Software Architecture Container Diagrams (Diagrama de Contenedores de Arquitectura de Software) 14](#_Toc177977619)

[4.12. Software Architecture Components Diagrams (Diagrama de Componentes de Arquitectura de Software) 14](#_Toc177977620)

[Capítulo V: Product Implementation, Validation & Deployment 14](#_Toc177977621)

[5.1. Software Configuration Management (Gestión de la Configuración de Software) 14](#_Toc177977622)

[5.1.1. Software Development Environment Configuration (Configuración del Entorno de Desarrollo de Software) 14](#_Toc177977623)

[5.1.2. Source Code Management (Gestión del Código Fuente) 15](#_Toc177977624)

[5.1.3. Source Code Style Guide & Conventions (Guía de Estilo del Código y Convenciones) 15](#_Toc177977625)

[5.1.4. Software Deployment Configuration (Configuración del Despliegue de Software) 16](#_Toc177977626)

[5.2. Project Management (Gestión del Proyecto) 16](#_Toc177977627)

[5.2.1. Requirements Management (Gestión de Requisitos) 16](#_Toc177977628)

[5.2.2. Product UX/UI Design (Diseño UX/UI del Producto) 16](#_Toc177977629)

[5.2.3. Software Development (Desarrollo de Software) 16](#_Toc177977630)

[5.3. Software Testing (Pruebas de Software) 17](#_Toc177977631)

[5.3.1. Unit Testing (Pruebas Unitarias) 17](#_Toc177977632)

[5.3.2. Integration Testing (Pruebas de Integración) 17](#_Toc177977633)

[5.3.3. End-to-End Testing (Pruebas de Extremo a Extremo) 17](#_Toc177977634)

[5.4. Software Documentation (Documentación de Software) 17](#_Toc177977635)

[5.5. Software Deployment (Despliegue de Software) 17](#_Toc177977636)

[5.5.1. Deployment Pipeline (Canal de Despliegue) 17](#_Toc177977637)

[5.5.2. Validación Post-Despliegue 18](#_Toc177977638)

[Conclusiones 18](#_Toc177977639)

[Recomendaciones 18](#_Toc177977640)

[Lecciones Aprendidas 19](#_Toc177977641)

# Capítulo I: Introducción

## 1.1. Descripción del Proyecto

La Universidad Tecnológica del Perú (UTP) ha decidido implementar una plataforma moderna y responsiva para la matrícula en línea de sus estudiantes. Este proyecto tiene como objetivo desarrollar un sistema que permita a los usuarios (estudiantes) seleccionar sus cursos, visualizar horarios y secciones, así como confirmar su matrícula de manera eficiente y sencilla. Se busca mejorar la experiencia del usuario a través de un diseño optimizado y una interfaz intuitiva que siga principios de UX/UI.

El desarrollo de esta solución permitirá reducir los tiempos de matrícula, optimizar el proceso y evitar errores comunes como la selección de cursos en conflicto de horario.

## 1.2. Descripción del Postulante

Mi nombre es Victor Raul Herrera Castillo, postulante en el Programa **Expedition Trainee UTP**. Cuento con experiencia en el desarrollo de aplicaciones web utilizando tecnologías modernas como React JS, jQuery, y siguiendo las mejores prácticas de diseño centrado en el usuario. Mi enfoque está en crear soluciones robustas y escalables, cumpliendo con los estándares de desarrollo y accesibilidad web.

## 1.3. Perfil de la Solución

La solución propuesta es una aplicación web con diseño responsivo que se adapta a dispositivos móviles, tabletas y computadoras de escritorio. Esta aplicación permitirá a los estudiantes de la UTP realizar su matrícula en línea de manera rápida y segura, garantizando la mejor experiencia de usuario mediante validaciones automáticas y una interfaz gráfica clara y funcional.

Los elementos clave del sistema son:

* **Plataforma Web Responsiva:** Asegura que la plataforma funcione de manera fluida en diferentes tipos de dispositivos.
* **Selección de Cursos y Secciones:** Incluye validaciones para evitar la selección incorrecta de cursos con cruces de horario.
* **UX/UI Optimizado:** Basado en principios de diseño de interfaces de usuario amigables e intuitivas.

## 1.4. Segmento Objetivo

El segmento objetivo de esta plataforma son los estudiantes de pregrado de la Universidad Tecnológica del Perú que realizan el proceso de matrícula en línea cada semestre. Los estudiantes suelen tener diferentes niveles de habilidad técnica, por lo que la interfaz debe ser simple y fácil de usar, sin sacrificar la funcionalidad.

# Capítulo II: Requirements Elicitation & Analysis

## 2.1. Introducción

En esta etapa del proyecto, se identifican y analizan los requisitos que permitirán desarrollar la plataforma de matrícula en línea para la Universidad Tecnológica del Perú (UTP). El objetivo es comprender las necesidades de los usuarios y las partes interesadas, garantizando que la solución propuesta cumpla con las expectativas y resuelva los problemas actuales del proceso de matrícula.

## 2.2. Needfinding (Identificación de Necesidades)

La identificación de necesidades es una etapa crítica del proyecto, ya que asegura que el producto final satisfaga los requisitos reales de los usuarios. Para esta plataforma de matrícula en línea, las principales fuentes de necesidades incluyen:

* **Estudiantes:** Los usuarios principales del sistema, quienes deben poder seleccionar cursos, ver horarios y secciones, y confirmar su matrícula de manera eficiente.
* **Administración Académica:** Los administradores que gestionan el catálogo de cursos y secciones, asegurándose de que el sistema ofrezca toda la información necesaria y valide correctamente los cruces de horarios.
* **Tutores o Asesores Académicos:** Proveen orientación en la selección de cursos, asegurándose de que los estudiantes cumplan con los requisitos académicos.

**Técnicas de Needfinding:**

* **Entrevistas:** Realizar entrevistas con estudiantes y administradores para comprender sus necesidades específicas, problemas actuales y expectativas.
* **Cuestionarios:** Distribuir encuestas entre los estudiantes para obtener una muestra más amplia de necesidades y problemas con el sistema de matrícula actual.
* **Análisis de Sistemas Existentes:** Revisar el sistema de matrícula actual para identificar sus limitaciones y áreas de mejora, como la gestión de cruces de horarios y la navegación en dispositivos móviles.

## 2.3. Ubiquitous Language (Lenguaje Ubicuo)

El lenguaje ubicuo es fundamental para que todos los miembros del equipo de desarrollo, así como las partes interesadas, tengan una comprensión común del proyecto. A continuación se presentan los términos clave que se utilizarán a lo largo del desarrollo de la plataforma:

* **Estudiante:** El usuario principal del sistema, que debe seleccionar cursos y secciones.
* **Matrícula:** El proceso completo que involucra la selección, validación y confirmación de los cursos por parte del estudiante.
* **Curso Obligatorio:** Un curso que el estudiante debe tomar como parte de su plan académico.
* **Curso Opcional:** Un curso que el estudiante puede elegir tomar o no, dependiendo de su plan académico y disponibilidad.
* **Sección:** Un grupo específico de un curso, con horarios y profesores asignados.
* **Cruce de Horario:** Una situación en la que dos cursos seleccionados tienen horarios que se superponen.
* **Confirmación de Matrícula:** El paso final donde el estudiante confirma los cursos y secciones seleccionados para inscribirse oficialmente en ellos.
* **Ventanas de Matrícula:** Interfaces dentro de la plataforma que permiten la selección, visualización y confirmación de cursos.

Este vocabulario facilitará la comunicación clara entre los desarrolladores, diseñadores, administradores y usuarios durante todo el ciclo de desarrollo del proyecto.

## 2.4. Análisis de Requisitos

Los requisitos funcionales y no funcionales se recopilan a través de la interacción con los usuarios y las partes interesadas, y se analizan para asegurarse de que sean claros, viables y medibles. Los principales requisitos identificados son:

## 2.4.1. Requisitos Funcionales

* **Selección de Cursos:** Los estudiantes deben poder ver y seleccionar los cursos disponibles, tanto obligatorios como opcionales.
* **Selección de Secciones:** Los estudiantes pueden elegir secciones para los cursos, con validaciones que aseguren que no haya cruces de horario.
* **Confirmación de Matrícula:** El sistema debe permitir a los estudiantes revisar y confirmar los cursos y secciones seleccionados.
* **Validaciones:** Se deben implementar reglas de validación para evitar la selección de cursos que se crucen o que no sean aplicables según el plan académico del estudiante.

## 2.4.2. Requisitos No Funcionales

* **Responsividad:** La plataforma debe ser completamente funcional en dispositivos móviles, tabletas y computadoras de escritorio.
* **Usabilidad:** La interfaz debe ser intuitiva y fácil de navegar, reduciendo al mínimo el tiempo necesario para completar el proceso de matrícula.
* **Rendimiento:** La plataforma debe manejar múltiples usuarios simultáneamente sin que esto afecte el rendimiento o cause tiempos de espera largos.
* **Seguridad:** Debe garantizarse la protección de los datos personales y académicos de los estudiantes, incluyendo la prevención de inyecciones de código malicioso.

# Capítulo III: Requirements Specification

## 3.1. Introducción

La especificación de requisitos es un paso clave para asegurar que todas las necesidades del proyecto se traduzcan en funcionalidades concretas y claras que el equipo de desarrollo debe implementar. En este capítulo, se presentan las **User Stories** que describen las necesidades desde el punto de vista del usuario y el **Product Backlog** que define las tareas prioritarias para el desarrollo del sistema de matrícula en línea.

## 3.2. User Stories (Historias de Usuario)

Las historias de usuario son una herramienta ágil que permite describir las funcionalidades del sistema desde la perspectiva del usuario final. A continuación se presentan las historias de usuario relevantes para el sistema de matrícula en línea de la UTP:

### 3.2.1. Historia de Usuario 1: Selección de Cursos

* **Como** estudiante de la UTP,
* **quiero** poder ver y seleccionar los cursos disponibles para el próximo semestre,
* **para** asegurarme de que cumplo con los requisitos de mi plan académico.

**Criterios de Aceptación:**

* + Debo poder ver una lista de todos los cursos obligatorios y opcionales.
  + Debo poder seleccionar cursos opcionales según mis preferencias.

### 3.2.2. Historia de Usuario 2: Selección de Secciones

* **Como** estudiante que ha seleccionado un curso,
* **quiero** poder elegir una sección para cada curso de acuerdo a mis horarios,
* **para** evitar conflictos de tiempo entre los cursos seleccionados.

**Criterios de Aceptación:**

* + Debo poder ver una lista de secciones disponibles para cada curso.
  + Si no hay secciones disponibles para un curso, debe aparecer un mensaje de alerta.
  + El sistema debe validar los cruces de horario, pero no restringir la matrícula.

### 3.2.3. Historia de Usuario 3: Confirmación de Matrícula

* **Como** estudiante que ha seleccionado cursos y secciones,
* **quiero** confirmar mi matrícula,
* **para** completar el proceso de inscripción.

**Criterios de Aceptación:**

* + Debo poder revisar los cursos y secciones seleccionados antes de confirmar.
  + Debo recibir una confirmación final de que mi matrícula ha sido procesada.

### 3.2.4. Historia de Usuario 4: Visualización de Horario

* **Como** estudiante que ha completado la selección de cursos,
* **quiero** ver mi horario de manera gráfica,
* **para** tener una visión clara de cómo se distribuyen mis clases a lo largo de la semana.

**Criterios de Aceptación:**

* + Debo poder ver un horario semanal (de lunes a sábado) con los cursos seleccionados.

## 3.3. Product Backlog

El **Product Backlog** es una lista priorizada de todas las tareas y funcionalidades que deben implementarse en el sistema de matrícula en línea. Cada ítem en el backlog se relaciona con una historia de usuario y sigue un orden de prioridad que facilita el desarrollo iterativo.

**Lista de Product Backlog:**

1. **Desarrollar ventana de selección de cursos** (Historia de Usuario 1)
   * Prioridad: Alta
   * Descripción: Implementar la interfaz que permite al estudiante ver y seleccionar cursos obligatorios y opcionales.
2. **Desarrollar ventana de selección de secciones** (Historia de Usuario 2)
   * Prioridad: Alta
   * Descripción: Crear una interfaz que muestre las secciones disponibles y valide cruces de horario.
3. **Implementar validaciones de cruce de horarios** (Historia de Usuario 2)
   * Prioridad: Media
   * Descripción: Asegurar que el sistema identifique los cursos con conflictos de horarios, sin restringir la matrícula.
4. **Desarrollar ventana de confirmación de matrícula** (Historia de Usuario 3)
   * Prioridad: Alta
   * Descripción: Permitir que el estudiante revise y confirme su matrícula.
5. **Implementar visualización gráfica del horario** (Historia de Usuario 4)
   * Prioridad: Media
   * Descripción: Mostrar un calendario semanal que refleje los cursos y secciones seleccionados por el estudiante.
6. **Desarrollar sistema de alertas para cursos sin secciones** (Historia de Usuario 2)
   * Prioridad: Baja
   * Descripción: Implementar alertas para cursos sin secciones disponibles.
7. **Optimización para dispositivos móviles y tabletas**
   * Prioridad: Alta
   * Descripción: Asegurar que todas las ventanas sean completamente responsivas y funcionen bien en dispositivos móviles.

## 3.4. Prioridades y Gestión de Requisitos

El backlog está priorizado en función de la importancia para el usuario final y el impacto en la usabilidad del sistema. Las tareas con prioridad alta son críticas para el correcto funcionamiento del sistema de matrícula, mientras que las tareas de prioridad media y baja se pueden abordar en iteraciones posteriores.

**Criterios de Prioridad:**

* **Alta:** Funcionalidades esenciales para completar el proceso de matrícula (selección de cursos, secciones, y confirmación de matrícula).
* **Media:** Funcionalidades que mejoran la experiencia del usuario, como la visualización gráfica de horarios.
* **Baja:** Mejoras adicionales o funciones que no impiden la operatividad del sistema si no están presentes en la primera versión.

# Capítulo IV: Product Design

## 4.1. Introducción

El diseño del producto es una fase crucial en la que se define cómo interactuará el usuario con el sistema de matrícula en línea, y cómo se organizarán los elementos visuales y funcionales para proporcionar una experiencia de usuario óptima. Este capítulo cubre desde las pautas generales de estilo, pasando por la arquitectura de la información, hasta el diseño de la interfaz de usuario (UX/UI) y la arquitectura de software.

## 4.2. General Style Guidelines (Pautas Generales de Estilo)

Las pautas de estilo aseguran consistencia visual y funcional a lo largo de la aplicación. Para la plataforma de matrícula en línea de la UTP, se definen los siguientes aspectos:

* **Paleta de Colores:**  
  Utiliza colores que reflejen la identidad institucional de la UTP. Por ejemplo:
  + **Color Primario:** Negro (#000000) y Rojo (#D01A49) - color institucional de la UTP.
  + **Color Secundario:** Blanco (#FFFFFF) - para el fondo y elementos neutros.
  + **Color de Énfasis:** Verde (#007523) - para botones importantes como "Confirmar Matrícula" o alertas.
* **Tipografía:**  
  Se sugiere usar una fuente legible y moderna como **Roboto** o **Open Sans**, con los siguientes tamaños:
  + Título: 24px
  + Subtítulos: 18px
  + Texto principal: 14px
  + Etiquetas de botones: 16px
* **Estilo de Botones:**  
  Los botones principales (acciones críticas) deben ser de un color de alto contraste como el verde (#007523) con texto blanco. Los botones secundarios deben tener un contorno y colores menos resaltantes (gris oscuro).
* **Espaciado y Márgenes:**  
  El diseño debe ser limpio y minimalista, con un espaciado generoso entre secciones para evitar sobrecargar la interfaz.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

## 4.3. Information Architecture (Arquitectura de la Información)

La arquitectura de la información se refiere a cómo está organizada y estructurada la plataforma. En este sistema de matrícula en línea, se utiliza un modelo jerárquico simple, donde los usuarios navegan a través de diferentes ventanas de interacción:

1. **Pantalla de Inicio:**
   * Bienvenida y enlace a las ventanas de matrícula.
2. **Ventana de Selección de Cursos:**
   * Listado de cursos disponibles.
   * Clasificación en obligatorios y opcionales.
3. **Ventana de Selección de Secciones:**
   * Listado de secciones disponibles para cada curso seleccionado.
   * Validación de cruces de horarios.
4. **Ventana de Confirmación de Matrícula:**
   * Revisión de cursos y secciones seleccionadas.
   * Botón para confirmar y finalizar matrícula.
5. **Pantalla de Horario:**
   * Visualización gráfica del horario con las clases seleccionadas.

## 4.4. Web Applications UX/UI Design

El diseño de la experiencia de usuario (UX) y la interfaz de usuario (UI) se centra en hacer el proceso de matrícula lo más intuitivo posible. Algunos principios clave son:

* **Navegación Clara:**  
  Cada paso del proceso de matrícula debe ser fácil de entender y navegar, con botones claramente etiquetados y visibles. La navegación debe ser lineal, permitiendo al usuario avanzar y retroceder entre ventanas con facilidad.
* **Validaciones Visuales:**  
  Los errores de usuario (como la selección de cursos que se superponen en horario) deben ser señalados claramente, con alertas visuales y mensajes descriptivos.
* **Diseño Adaptativo:**  
  El sistema debe ser responsivo, asegurando que los elementos cambien de tamaño y posición de manera adecuada en dispositivos móviles y tabletas.

## 4.5. Web Applications Wireframes (Esquemáticos de Aplicaciones Web)

Los wireframes son representaciones esquemáticas que muestran el diseño básico de cada pantalla. A continuación se describen los wireframes clave del sistema:

* **Wireframe 1: Selección de Cursos**
  + Un listado con casillas de verificación para los cursos.
  + Secciones de "Cursos Obligatorios" y "Cursos Opcionales".
  + Botón para continuar a la selección de secciones.
* **Wireframe 2: Selección de Secciones**
  + Una tabla con las secciones disponibles para el curso seleccionado.
  + Mensaje de advertencia si no hay secciones disponibles.
  + Botón de "Confirmar Selección".
* **Wireframe 3: Confirmación de Matrícula**
  + Lista de los cursos y secciones seleccionadas.
  + Botón grande y destacado para "Confirmar Matrícula".
* **Wireframe 4: Visualización de Horario**
  + Un calendario semanal (lunes a sábado).
  + Bloques de colores para cada clase seleccionada.

Tabla

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente

Imagen que contiene Código QR

Descripción generada automáticamente

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

## 4.6. Web Applications Wireflow Diagrams (Diagrama de Flujo de Interacción de las Aplicaciones Web)

El **Wireflow Diagram** combina el esquema de la interfaz de usuario con el flujo de navegación. A continuación, se describe el flujo del usuario en la plataforma:

1. **Pantalla de Inicio** → **Selección de Cursos** → **Selección de Secciones** → **Confirmación de Matrícula** → **Visualización del Horario**

Diagrama

Descripción generada automáticamente

Cada paso está claramente conectado y permite al usuario navegar de manera lógica y directa a través del proceso de matrícula.

## 4.7. Web Applications Mock-ups (Maquetas de Aplicaciones Web)

Los **Mock-ups** son representaciones más detalladas y con estilo visual aplicado de los wireframes. En esta fase, se integran los colores, tipografías, y elementos de estilo definidos en las pautas generales. Cada maqueta debe reflejar el diseño final que verá el usuario, incluyendo botones, mensajes de validación, y otros elementos interactivos.

Interfaz de usuario gráfica

Descripción generada automáticamente

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

## 4.8. Web Applications User Flow Diagrams (Diagramas de Flujo del Usuario)

El **User Flow Diagram** detalla los posibles caminos que el usuario puede tomar en la aplicación. Los puntos clave son:

* **Flujo principal:** Desde la pantalla de selección de cursos hasta la confirmación de matrícula.
* **Ramas secundarias:** Alertas por cruces de horario o falta de secciones, con opciones para regresar a la selección de cursos.

Interfaz de usuario gráfica

Descripción generada automáticamente

## 4.9. Domain-Driven Software Architecture (Arquitectura de Software Basada en el Dominio)

El desarrollo de la plataforma sigue un enfoque **Domain-Driven Design (DDD)**, donde los conceptos clave del dominio académico (cursos, secciones, matrícula) están claramente definidos y organizados. Esto facilita la comprensión del sistema tanto para los desarrolladores como para las partes interesadas no técnicas.

## 4.10. Software Architecture Context Diagram (Diagrama de Contexto de Arquitectura de Software)

El **Context Diagram** ilustra cómo la plataforma de matrícula en línea interactúa con otros sistemas:

* **Usuarios:** Estudiantes acceden a la plataforma a través de dispositivos móviles y computadoras.
* **Base de Datos Académica:** Fuente de los cursos y secciones disponibles.
* **Servidor Web:** Procesa las solicitudes de los usuarios y gestiona la lógica de matrícula.

## 4.11. Software Architecture Container Diagrams (Diagrama de Contenedores de Arquitectura de Software)

El **Container Diagram** muestra los componentes principales del sistema y cómo interactúan entre sí:

* **Front-end:** Desarrollado con React JS, maneja la interfaz de usuario.
* **Back-end:** API que gestiona las solicitudes de los estudiantes y las validaciones.
* **Base de Datos:** Almacena la información de cursos, secciones y matrículas confirmadas.

## 4.12. Software Architecture Components Diagrams (Diagrama de Componentes de Arquitectura de Software)

El **Components Diagram** desglosa los componentes internos del sistema, como módulos de selección de cursos, validación de horarios y generación de horarios.

# Capítulo V: Product Implementation, Validation & Deployment

## 5.1. Software Configuration Management (Gestión de la Configuración de Software)

La gestión de la configuración es un aspecto crucial para garantizar que el desarrollo y despliegue del sistema de matrícula en línea sea ordenado y eficiente. En esta sección, se describen las configuraciones necesarias tanto para el entorno de desarrollo como para la gestión del código fuente y las convenciones de estilo.

### 5.1.1. Software Development Environment Configuration (Configuración del Entorno de Desarrollo de Software)

El entorno de desarrollo debe configurarse adecuadamente para garantizar la correcta implementación y mantenimiento del sistema. Las herramientas y tecnologías utilizadas en este proyecto incluyen:

* **Framework de Desarrollo:** React JS para la construcción del front-end.
* **Versionamiento del Código:** Git para el control de versiones.
* **IDE:** Visual Studio Code o cualquier otro editor de código compatible.
* **Gestor de Paquetes:** npm o Yarn para la gestión de dependencias.
* **Servidor de Desarrollo Local:** Usar herramientas como Webpack Dev Server o el servidor incorporado de React para pruebas locales.
* **Herramientas de Validación de Código:** ESLint y Prettier para mantener un código limpio y consistente.

### 5.1.2. Source Code Management (Gestión del Código Fuente)

El código fuente se gestiona utilizando Git y se versiona en un repositorio de GitHub.

* **Repositorio GitHub:**  
  <https://github.com/christian10x/internship_evaluations/tree/victor_herrera_front_end>
* **Estrategia de Branching: Gitflow**  
  El proyecto sigue la metodología **Gitflow** para la gestión de ramas:
  + **Rama main:** Contiene la versión estable y lista para producción.
  + **Rama victor\_herrera\_front\_end:** Contiene el código en desarrollo con nuevas funcionalidades y documentacion.

### 5.1.3. Source Code Style Guide & Conventions (Guía de Estilo del Código y Convenciones)

Es importante seguir una serie de convenciones y estándares para asegurar la mantenibilidad del código:

* **Estructura de Carpetas:**
  + **/src:** Contiene el código fuente de la aplicación.
  + **/components:** Componentes React reutilizables.
  + **/services:** Lógica de negocio y llamadas a la API.
  + **/assets:** Imágenes, fuentes y otros recursos estáticos.
* **Guía de Estilo:**
  + Utilización de **ESLint** para asegurar la calidad del código.
  + Estandarización del formato de código con **Prettier** (uso de comillas simples, sangría de 2 espacios, etc.).
* **Convenciones de Nomenclatura:**
  + Archivos y carpetas con nombres en **camelCase**.
  + Componentes React deben nombrarse con **PascalCase**.

### 5.1.4. Software Deployment Configuration (Configuración del Despliegue de Software)

El despliegue del software debe realizarse de manera automática y eficiente, asegurando que los cambios en el código lleguen a producción sin errores.

* **Plataforma de Despliegue:**  
  Utilizar **Netlify** o **Vercel** para el despliegue del front-end, ya que ofrecen soporte para proyectos en React y permiten despliegues automáticos desde GitHub.
* **Pipeline de Despliegue Automático:**
  + Al realizar un **merge** en la rama main, el despliegue debe ocurrir de manera automática a través del pipeline configurado en Netlify/Vercel.
  + Utilizar integraciones de GitHub para desencadenar el despliegue en cada commit hacia main.

## 5.2. Project Management (Gestión del Proyecto)

### 5.2.1. Requirements Management (Gestión de Requisitos)

La gestión de requisitos se realiza en una herramienta como **Trello** o **Jira**, donde se puede gestionar el backlog del proyecto, asignar tareas, y seguir el progreso de cada historia de usuario. Cada funcionalidad corresponde a una **User Story** del capítulo III, lo que facilita la trazabilidad.

### 5.2.2. Product UX/UI Design (Diseño UX/UI del Producto)

El diseño UX/UI es iterativo, y las versiones de los prototipos (wireframes, mockups) se versionan junto con el código fuente. Las herramientas utilizadas para el diseño incluyen **Figma** y **Adobe XD**, que permiten una colaboración ágil entre diseñadores y desarrolladores.

### 5.2.3. Software Development (Desarrollo de Software)

El desarrollo sigue un enfoque ágil, con iteraciones cortas que permiten lanzar versiones incrementales del producto. Cada iteración se basa en los **sprints** planificados en la herramienta de gestión del proyecto.

## 5.3. Software Testing (Pruebas de Software)

Las pruebas son fundamentales para garantizar la calidad del producto. A continuación se describen los tipos de pruebas implementadas:

### 5.3.1. Unit Testing (Pruebas Unitarias)

* Utilización de **Jest** y **React Testing Library** para realizar pruebas unitarias de los componentes React.

### 5.3.2. Integration Testing (Pruebas de Integración)

* Las pruebas de integración se enfocan en validar la interacción entre los componentes principales de la aplicación, asegurando que las distintas partes del sistema funcionen correctamente juntas.

### 5.3.3. End-to-End Testing (Pruebas de Extremo a Extremo)

* Utilización de **Cypress** para pruebas E2E, simulando la interacción de un usuario real con la plataforma desde la selección de cursos hasta la confirmación de la matrícula.

## 5.4. Software Documentation (Documentación de Software)

Toda la documentación técnica, incluyendo la descripción de la arquitectura, guías de instalación y manuales de usuario, se almacenan en el repositorio y están disponibles en formatos accesibles.

* **Manual de Usuario:** Incluye instrucciones paso a paso para que los estudiantes utilicen el sistema de matrícula.
* **Documentación Técnica:** Explica las decisiones arquitectónicas, diseño de APIs y la estructura del código.

## 5.5. Software Deployment (Despliegue de Software)

El despliegue del sistema de matrícula en línea sigue los siguientes pasos:

### 5.5.1. Deployment Pipeline (Canal de Despliegue)

* **Integración Continua (CI):** Las pruebas unitarias y de integración se ejecutan automáticamente en cada push al repositorio.
* **Despliegue Continuo (CD):** El código que pasa las pruebas en la rama main se despliega automáticamente a producción.

### 5.5.2. Validación Post-Despliegue

Una vez desplegado el sistema, se realizan pruebas adicionales para asegurar que el entorno de producción funciona sin problemas. Esto incluye pruebas de carga para verificar el rendimiento bajo condiciones de uso real.

# Conclusiones

El desarrollo de la plataforma de matrícula en línea para la Universidad Tecnológica del Perú ha sido un proyecto enriquecedor que me ha permitido aplicar diversas tecnologías y metodologías, al mismo tiempo que he adquirido valiosas lecciones en la implementación de soluciones web orientadas al usuario. A lo largo de este proceso, he aprendido a:

1. **Aplicar Principios de UX/UI:** He reforzado la importancia de diseñar interfaces claras y funcionales, que simplifiquen la experiencia del usuario y le permitan realizar tareas complejas de manera intuitiva.
2. **Trabajar con Componentes Reutilizables:** El uso de **React JS** me permitió desarrollar componentes reutilizables, mejorando la eficiencia y escalabilidad del código, lo que facilita futuras modificaciones o expansiones de la plataforma.
3. **Gestionar un Proyecto Complejo bajo Metodologías Ágiles:** Implementar **Gitflow** y utilizar herramientas ágiles como **Jira** o **Trello** para gestionar las historias de usuario y el backlog me permitió organizar mejor las fases de desarrollo y priorizar las tareas críticas.

## Recomendaciones

El desarrollo de un proyecto en un corto período de tiempo presenta diversos retos, y a partir de esta experiencia, puedo ofrecer las siguientes recomendaciones:

1. **Priorizar Funcionalidades Críticas:** En proyectos con tiempos de desarrollo limitados, es fundamental centrarse primero en las funcionalidades esenciales (selección de cursos, validaciones básicas y confirmación de matrícula). La implementación de mejoras adicionales, como la visualización avanzada de horarios o alertas complejas, puede reservarse para iteraciones posteriores.
2. **División del Trabajo en Iteraciones Cortas:** Seguir una metodología ágil como **Scrum** permite organizar el trabajo en sprints cortos y manejables. Esto garantiza que se pueda entregar una versión funcional del producto al final de cada ciclo, y facilita la incorporación de feedback temprano por parte de los usuarios.
3. **Mantener la Documentación Actualizada:** En proyectos con plazos ajustados, la documentación puede quedar rezagada. Sin embargo, mantenerla actualizada es crucial para asegurar la mantenibilidad del proyecto a largo plazo. Esto incluye tanto la documentación técnica como los manuales de usuario.
4. **Optimización desde el Principio:** Desde las primeras fases del proyecto, es importante pensar en la **optimización de rendimiento**, tanto en dispositivos móviles como en computadoras. Minimizar los tiempos de carga y asegurar una buena experiencia de usuario en diferentes dispositivos debería ser una prioridad, especialmente en un sistema tan utilizado como el de matrícula.

## Lecciones Aprendidas

1. **Gestión del Tiempo:** Desarrollar una plataforma funcional en un corto periodo de tiempo requiere una excelente planificación y gestión del tiempo. Cada día debe tener objetivos claros y alcanzables, priorizando siempre lo más importante.
2. **Flexibilidad y Adaptación:** En un entorno de desarrollo ágil, es importante adaptarse rápidamente a los cambios y ajustar el enfoque según las prioridades del proyecto. Esto puede implicar reducir el alcance de ciertas funcionalidades o iterar rápidamente sobre prototipos iniciales.
3. **Comunicación Eficiente:** Mantener una comunicación clara y constante con el equipo y las partes interesadas es esencial para garantizar que todos estén alineados y que el producto final cumpla con los requisitos esperados.