

HairLife

Plataforma web para el cuidado
capilar personalizado



Institución: IES Virgen de la Paloma

Autor: Aissa Diallo González

Fecha inicio: 17/05/2025

1 Abstract

Español

Este proyecto, llamado "HairLife", es una plataforma web que creé para ayudar a las personas a cuidar su cabello de forma personalizada. La idea principal es que los usuarios respondan un cuestionario sobre su tipo de pelo y cuero cabelludo. Con esas respuestas, la plataforma les da recomendaciones directas y personalizadas de productos, y también les explica cómo usarlos correctamente.

El proyecto tiene dos tipos de usuarios: el usuario normal, que puede acceder al cuestionario y recibir información útil; y el administrador, que tiene control sobre los productos, pudiendo añadir, modificar o retirar los que están en la web.

Para construir "HairLife", usé HTML, CSS y JavaScript para lo que el usuario ve frontend, y PHP con el *framework* Laravel para la parte lógica backend. Todo esto se conecta a una base de datos MySQL donde guardo los perfiles de los usuarios, sus respuestas y los productos recomendados.

Mi objetivo era crear una web interactiva y fácil de usar, que resolviera el problema de la falta de información específica y el exceso de datos confusos que hay ahora.

English

This project, named "HairLife," is a web platform I created to help people take care of their hair in a personalized way. The main idea is for users to answer a questionnaire about their hair and scalp type. With these answers, the platform provides them with direct and personalized product recommendations, and also explains how to use them correctly.

The project has two types of users: the regular user, who can access the questionnaire and get useful information; and the administrator, who can control the products, adding, modifying, or removing them from the website.

To build "HairLife," I used HTML, CSS, and JavaScript for the frontend (what the user sees), and PHP with the Laravel *framework* for the backend (the logic behind it). All of this connects to a MySQL database where I store user profiles, their answers, and the recommended products.

My goal was to build an interactive and easy-to-use website, which solves the problem of not enough specific information and too much confusing data that exists nowadays.

Fuentes

2 Índices

Sumario

1 Abstract.....	2
Español.....	2
English.....	2
2 Índices.....	3
3 Justificación y antecedentes.....	6
4 Introducción.....	8
5 Objetivos del Proyecto.....	9
6 Estimación de Costes.....	11
7 Desarrollo del proyecto.....	13
7.1 Tecnología utilizada (IDE, Frameworks, lenguajes, herramientas.....)	13
7.1.1 Base de Datos.....	14
7.1.2 Lenguajes de Programación.....	14
7.1.3 Frameworks y Bibliotecas.....	15
7.1.4 Herramientas de Desarrollo.....	16
7.2 Planificación (metodología, temporalización – diagrama Gantt).....	18
7.2.1 Metodología de gestión del proyecto.....	18
7.2.2 Temporalización.....	18
7.2.3 Seguimiento.....	21
7.3.1 Requisitos de negocio.....	22
7.3.2 Requisitos Técnicos.....	23
7.4 Diseño de la aplicación (diagrama de BBDD, diagramas UML, prototipos).....	25
7.4.2 Modelado de Datos.....	26
Tabla productos.....	26
Tabla cuestionarios.....	27
Tabla preguntas.....	27
Tabla pregunta_opciones.....	27
Tabla cuestionario_pregunta :Tabla Pivote.....	28
Tabla cuestionario_envios.....	28
Tabla respuestas.....	28
Tabla recomendaciones.....	29
7.4.3 Modelado de Clases.....	29
Clase Usuario (app/Models/Usuario.php).....	29
Clase Producto (app/Models/Producto.php).....	30

Clase Cuestionario (app/Models/Cuestionario.php).....	30
Clase Pregunta (app/Models/Pregunta.php).....	30
Clase PreguntaOpcion (app/Models/PreguntaOpcion.php).....	30
Clase CuestionarioEnvio (app/Models/CuestionarioEnvio.php).....	30
Clase Respuesta (app/Models/Respuesta.php).....	31
Clase Recomendacion (app/Models/Recomendacion.php).....	31
7.4.4 Interfaz de Usuario.....	31
7.4.5 Interfaces Externas.....	33
7.5 Implementación y documentación (arquitectura y estructura de código, control de versiones, mapa web, estándares y validaciones utilizados).....	33
7.5.1 Implementación.....	33
7.5.1.1 Relación de módulos y archivos.....	34
7.5.1.2 Base de datos.....	36
7.5.1.3 Descripción de Procesos.....	37
1. Proceso de creación de nick e inicio de sesión de usuario.....	37
7.5.1.4 Secciones de código relevantes.....	39
7.5.2 Documentación.....	43
7.5.2.1 Documentación técnica de desarrollo.....	43
Para gestionar los distintos aspectos de mi aplicación, me he apoyado en las convenciones de Laravel para las configuraciones.....	43
Pasos para la Construcción del Entorno de Desarrollo.....	44
Software Requerido.....	44
Pasos de Construcción.....	44
Estructura de Directorios.....	45
Diagrama de Red / Conectividad de Servidores.....	47
Entorno de Desarrollo (Local).....	47
Entorno de Producción (Típico en Hosting Compartido/VPS):.....	47
En un entorno de producción, el flujo cambia un poco, ya que la aplicación está desplegada y accesible públicamente:.....	47
Usuarios/Claves Utilizados.....	48
Usuarios de la aplicación (Tabla usuarios en mi_pelo.sql):.....	48
Usuario de Base de Datos MySQL:.....	49
Fuentes de la Aplicación.....	49
7.5.2.2 Documentación técnica para explotación.....	49
7.5.2.3 Documentación de usuario.....	50
7.6 Plan de pruebas.....	52
7.6.1 Pruebas unitarias.....	52

Los Modelos (app/Models/Usuario.php, app/Models/Producto.php).....	52
Los Controladores (Lógica de Negocio Aislada - Ej: app/Http/Controllers/CuestionarioController.php).....	53
Otras Lógicas Aislables.....	53
7.6.2 Pruebas funcionales.....	53
Pruebas de Flujo de Usuario y UI.....	53
Navegación por el Panel de Usuario (show.blade.php).....	54
Cumplimentación del Cuestionario “ver_para_nick.blade.php”.....	55
Visualización de Recomendaciones.....	56
Responsividad de la Interfaz.....	56
7.6.3 Pruebas finales de usuario.....	57
8 Implantación.....	58
8.1. Qué Necesitamos para que Funcione.....	58
8.2. Despliegue de la aplicación.....	58
8.3. Mantenimiento de la aplicación.....	59
9 Conclusiones y Posibles Mejoras Futuras.....	60
Conclusiones.....	60
Cumplimiento de Objetivos.....	60
Lecciones Aprendidas.....	61
Limitaciones del Producto Actual.....	62
Decisiones Tomadas para Cumplir los Objetivos.....	62
Elección de Tecnologías.....	63
Mejoras Futuras.....	63
Ampliación y Desarrollo Completo de Funcionalidades:.....	63
Evolución del Sistema de Recomendación:.....	64
Mejoras Técnicas y de Arquitectura:.....	64
Mejoras en la Experiencia de Usuario (UX/UI):.....	65
Integraciones Externas:.....	65
10 Bibliografía y referencias.....	66
11 Anexos (Glosario de términos, versiones o bocetos de la aplicación, soporte multimedia utilizado -videos, etc.-).....	69
11. Anexos.....	69
11.1. Índice de Ilustraciones.....	69
11.2. Glosario de Términos.....	70
11.2. Glosario de Términos.....	70
11.3. Ilustraciones y Diagramas del Proyecto.....	71
11.3.1. Diagramas de Diseño.....	71

11.3.2. Prototipos y Flujo de Interfaz.....	73
11.3.2. Herramientas y Gestión del Proceso de Desarrollo.....	75
11.3.3. Capturas de Pantalla de la Aplicación (GUI en Funcionamiento).....	78

3 Justificación y antecedentes

Este proyecto se enfoca en el desarrollo de una herramienta digital innovadora para el sector de las peluquerías. La idea central es crear una interfaz web dinámica que gestione funcionalidades como una agenda capilar, para que los usuarios puedan observar la evolución de su cabello, una sección donde podamos recomendar a los usuarios sobre peinados y cortes de pelo según la información proporcionada por la agenda capilar ,y por último está la funcionalidad para mostrar un cuestionario personalizado, diseñado para obtener un conocimiento profundo sobre el tipo de cabello y las particularidades del cuero cabelludo de cada usuario, y recomendar un producto específico que mejore la calidad del cabello de los usuarios.

- La motivación que tuve para desarrollar este proyecto surge a raíz de una experiencia personal. Hace unos años , durante mi niñez y juventud enfrenté considerables dificultades debido a la falta de información específica sobre los diferentes tipos de cabello, especialmente el rizado, que por aquella época no era tan común ni comprendido, y por eso ideé esta forma para solventar el problema.
- **Condicionante Técnico/Profesional :** La elección de una plataforma web como solución responde a la necesidad de crear una herramienta accesible tanto para las peluquerías como para sus clientes, sin requerir métodos complejos de instalación y permitiendo una actualización constante y sencilla de la información y los productos.

Justificación y Objetivos Generales:

Aunque actualmente existe una mayor visibilidad sobre la diversidad capilar, también nos enfrentamos a una sobre información que puede resultar abrumadora y poco práctica. El objetivo fundamental de este proyecto es contrarrestar esto generando información directa, filtrada y personalizada. Se busca no solo diagnosticar, sino también enseñar y mostrar los productos más convenientes según las características individuales del cabello y cuero cabelludo, proporcionando además las instrucciones necesarias para su correcta aplicación. Se pretende ofrecer a las peluquerías una herramienta de valor añadido que les permita mejorar la calidad de su asesoramiento,y su trabajo .También se pretende

conservar a la clientela mediante un servicio personalizado y, optimizar la recomendación y venta de productos específicos.

Este proyecto permitirá aplicar conocimientos de frontend y backend ; diseño de interfaces interactivas para mejorar la experiencia del usuario; y una gestión de bases de datos.

Antecedentes, Estado del Arte y Ventana de Oportunidad:

Si bien hoy en día existen numerosos blogs, vídeos y foros con información general sobre el cuidado capilar, la mayoría ofrece consejos generales. La sobre información que hay actualmente dificulta que el usuario encuentre información realmente adaptadas a sus necesidades específicas. Considero que una herramienta digital que realice el diagnóstico capilar y lo conecte directamente con recomendaciones de productos concretos disponibles o recomendados por profesionales, podría ser de ayuda y de utilidad a los usuarios que quieran información y recomendaciones personalizadas.

Existe una demanda creciente de personalización y soluciones efectivas que ahorren tiempo y eviten la confusión generada por el exceso de información no cualificada. El sector de las peluquerías, ha avanzado de forma más lenta en la adopción de herramientas digitales . Por lo tanto introducir una solución digital en un mercado con un considerable potencial de crecimiento como es el caso ,podría ser una ventana de oportunidad, que no solo modernice el asesoramiento, sino que también ayude a las peluquerías a posicionarse como expertas en cuidado capilar individualizado, abriendo un nuevo canal de interacción y servicio.

4 Introducción

El Trabajo de Fin de Grado, titulado "HairLife", está enfocado en la necesidad de orientar al usuario sobre su cuidado capilar. Este proyecto se centra en el desarrollo de una plataforma web dinámica, diseñada tanto para peluquerías como para usuarios particulares, con el objetivo principal de simplificar y personalizar la elección de productos y rutinas de cuidado del cabello.

En un contexto donde la información es excesiva y a menudo contradictoria o generalizada, "HairLife" pretende ofrecer una solución específica. Mediante un cuestionario dinámico e intuitivo, la plataforma analizará las características y necesidades individuales del cabello y cuero cabelludo del usuario para generar recomendaciones personalizadas de productos y técnicas de aplicación. Adicionalmente, La plataforma contara con dos funcionalidades más ,un diario o una agenda donde el usuario pueda llevar un control sobre el tratamiento que esta empleando, y otra de las funcionalidades trata de recomendaciones de peinados y cortes de cabellos adaptados al tipo de pelo del usuario, estas funcionalidades se encontrarán disponibles para el cliente en el panel de usuario. Además, el sistema contará con un panel de administración que permitirá a las peluquerías gestionar el catálogo de productos , incluyendo su stock, adaptándose así a las demandas del mercado y a su oferta particular. De esta manera, se pretende no solo asistir al usuario final, sino también proporcionar a los profesionales de una herramienta que favorezca su servicio y mejore la gestión de sus recomendaciones.

Este documento está estructurado para ofrecer una visión completa del proyecto, desde su creación hasta su futura implementación y desarrollo. Primeramente comenzará con la Justificación y Antecedentes , donde se exponen los motivos que impulsan esta iniciativa y el contexto actual del sector. A continuación, en el apartado de Objetivos , se definirán las metas específicas que se persiguen con el desarrollo de la plataforma.

Posteriormente, se profundizará en los aspectos prácticos y técnicos del proyecto. El punto 7, Desarrollo del proyecto, se divide en secciones cruciales como la Tecnología utilizada, la Planificación detallada incluyendo la temporalización mediante un diagrama de Gantt , la Especificación de Requisitos funcionales y no funcionales, y el Diseño de la Aplicación con diagramas de base de datos, diagramas UML y prototipos. Seguidamente, se abordará la Implementación y Documentación del sistema, el Plan de Pruebas realizado para asegurar su correcto funcionamiento, y las consideraciones para su Despliegue y Mantenimiento.

Finalmente, el documento presentará las Conclusiones extraídas del desarrollo del TFG, se plantearán posibles Mejoras Futuras para la plataforma, y se incluirá la Bibliografía y Referencias consultadas, así como los Anexos pertinentes.

5 Objetivos del Proyecto

A continuación, se presentan los objetivos específicos del proyecto "HairLife", diseñados para poder medirse y adaptarse al trabajo técnico del software, aplicando los conocimientos adquiridos durante el ciclo formativo.

Desarrollar un Sistema de Cuestionario Interactivo y Personalizado para la Recomendación de Productos Capilares.

- Dentro del alcance de este proyecto, se implementará una funcionalidad de cuestionario dinámico en la plataforma web "ver_para_nick.blade.php". Este cuestionario permitirá a los usuarios, identificados por un nick, responder a una serie de preguntas sobre su tipo de cabello y preferencias. El sistema procesará estas respuestas en la función "procesarEnvioParaNick" que está en el controlador "CuestionarioController", para generar una recomendación de un producto específico de la base de datos (Producto model). La lógica de recomendación considerará al menos la categoría de producto deseada y el tipo de cabello, y el 40% de las respuestas del usuario de las otras preguntas, para filtrar y seleccionar un producto adecuado que cumpla con estos requisitos mínimos. Se implementará la capacidad de volver a rellenar el cuestionario con respuestas de un envío previo si el usuario vuelve desde una página de recomendación: "ver_producto.blade.php" "CuestionarioController".

Diseñar e Implementar una Interfaz de Usuario : Atractiva, Responsiva y de Fácil Navegación para Usuarios y Administradores.

- Se crearán interfaces de usuario utilizando HTML, CSS con variables CSS para una paleta de colores con tonalidades coherentes y JavaScript, apoyándose en el framework Bootstrap para asegurar la responsividad en distintos dispositivos (escritorio, tableta, móvil). Esto incluye:
 - Una página de bienvenida/creación de nick "crearNick.blade.php".
 - Un panel de control para el usuario "show.blade.php" con acceso a las tres funcionalidades principales de la página.

- Una interfaz clara para la visualización y respuesta del cuestionario “ver_para_nick.blade.php” con navegación entre preguntas : botones "Anterior/Siguiente", y feedback visual de errores de validación.
- Páginas informativas para mostrar el producto recomendado “ver_producto.blade.php” y mensajes de proceso de respuesta como por ejemplo: “gracias.blade.php”.
- La usabilidad se comprobará mediante la aplicación de principios de diseño centrado en el usuario y pruebas funcionales en diferentes navegadores y tamaños de pantalla.

Implementar un Sistema de Gestión de Usuarios Basado en Nicknames y un Rol de Administrador.

- Se desarrollará un sistema para que los usuarios puedan operar en la plataforma utilizando un nick único “LoginController@guardarNick, Usuario model”.
- Se establecerá un sistema de autenticación para un rol de Administrador “LoginController@login, Usuario model con campo Rol”, que aunque su panel de gestión no sea el foco principal del desarrollo frontend de este TFG, sentará las bases para futuras ampliaciones donde podrá gestionar productos, preguntas del cuestionario y otros contenidos de la plataforma. Se verificarán al menos dos roles distintos: "Usuario" identificado por nick y con Rol = 0, y "Administrador" con Rol = 1.

Desarrollo de la Lógica de Negocio y la Arquitectura Backend para Soportar las Funcionalidades Clave de la Aplicación.

- Utilizando PHP con el framework Laravel, se construyeron los controladores “LoginController, CuestionarioController, HomeController, RecomendacionController” y modelos (Usuario, Cuestionario, Pregunta, Producto, Recomendacion, CuestionarioEnvio, Respuesta, PreguntaOpcion) necesarios para gestionar el flujo de datos.
- Se diseñó y utilizó una base de datos MySQL para persistir la información de usuarios, nicks, cuestionarios, preguntas, opciones, respuestas, productos y recomendaciones.
- Se implementaron las rutas necesarias (web.php) para la navegación y la interacción entre el frontend y el backend. Se aseguro el correcto almacenamiento de los envíos de cuestionarios y las recomendaciones generadas.

6 Estimación de Costes

Categoría	Descripción	Unidades de Coste	Precio Unitario	Coste Total	Horas Estimadas
Costes de materiales de producción	Ordenador personal y periféricos	1	0	0	N/A
	Entorno de Desarrollo Integrado (IDE) -Visual Studio Code	1	0	0	N/A
	Sistema Operativo -Windows	1	0	0	N/A
Costes de materiales de producción	Software de gestión de bases de datos -MySQL Workbench	1	0	0	N/A
	Software de control de versiones (Git)	1	0	0	N/A
	Dominio Web - .online	1 Año	0	0	N/A
Servicios IT	Hosting Web Básico - PHP - MySQL	1 Año	0	0	N/A
	Certificado SSL		0	0	N/A

Categoría	Descripción	Unidades de Coste	Precio Unitario	Coste Total	Horas Estimadas
-----------	-------------	-------------------	-----------------	-------------	-----------------

1 Año

Subtotal A (Servicios IT y Software)				35€	
---	--	--	--	------------	--

Costes de Personal	Análisis y Planificación	Horas	Trabajo propio	0	20
	Diseño de la Aplicación	Horas	Trabajo propio	0	50
	Desarrollo del Backend	Horas	Trabajo propio	0	80
	Desarrollo del Frontend	Horas	Trabajo propio	0	50
	Pruebas y Depuración	Horas	Trabajo propio	0	20
	Documentación	Horas	Trabajo propio	0	30
	Despliegue y Configuración del Servidor	Horas	Trabajo propio	0	5
Horas de Trabajo Estimadas				255 horas	

Servicios Externos	Mantenimiento Anual (Dominio + Hosting Básico)	1 Año	65 - 170	65 - 170	N/A
	Plantillas o Componentes UI Premium.	1 Licencia	0	0	N/A
	API de IA para Recomendaciones	Suscripción	0	0	N/A

Categoría	Descripción	Unidades de Coste	Precio Unitario	Coste Total	Horas Estimadas
(Para un futuro)					
COSTE TOTAL ESTIMADO				35 €	255 - 400 horas

7 Desarrollo del proyecto

7.1 Tecnología utilizada (IDE, Frameworks, lenguajes, herramientas...)

7.1.1 Base de Datos

Se ha seleccionado **MySQL** como el sistema de gestión de bases de datos relacional.

- **Descripción:** MySQL es un sistema de gestión de bases de datos relacional de código abierto, popular por su rendimiento y facilidad de uso. Lo usan muchos servicios de hosting y tiene una gran comunidad.
- **Alternativas Consideradas:**
 - **MongoDB:** Es una base de datos flexible y fácil de usar, buena para datos que cambian mucho o no tienen una estructura fija. Pero como *HairLife* usa datos más organizados, MySQL fue mejor opción por su facilidad, buen manejo de relaciones y compatibilidad con la mayoría de servicios de hosting.
 - **SQLite:** Es muy ligera y fácil de usar, ideal para proyectos pequeños o pruebas. Sin embargo, MySQL fue preferida porque se adapta mejor a un sitio web que puede crecer y necesita más herramientas de gestión.
- **Justificación de la Elección:**
 - **MySQL:** es un sistema de gestión de bases de datos que ya he utilizado previamente, por lo tanto cuento con un conocimiento más alto en este sistema en comparación con otros.
 - La mayoría de los proveedores de hosting ofrecen soporte para MySQL, lo que simplifica el despliegue.
 - **Laravel:** ofrece un excelente soporte para MySQL a través de su ORM Eloquent, facilitando las operaciones de base de datos.

7.1.2 Lenguajes de Programación

- **Backend: PHP**
 - **Descripción:** PHP es un lenguaje de scripting del lado del servidor muy utilizado, especialmente adecuado para el desarrollo web. Es la base del framework Laravel.
 - **Alternativas Consideradas:**

- **Python (con Django/Flask):** Es una opción potente y flexible, pero se eligió **PHP** porque **Laravel**, el framework usado, está basado en PHP y hace que el desarrollo sea más rápido y ordenado.
- **Node.js (JavaScript):** Permite usar el mismo lenguaje JavaScript en el frontend y el backend, lo cual puede ser útil. Sin embargo, se eligió PHP con Laravel porque ya había adquirido experiencia previa con esta tecnología y ofrece herramientas más prácticas y fáciles de usar para desarrollar este tipo de aplicaciones.
- **Justificación de la Elección:**
 - La elección de Laravel como framework principal dicta el uso de PHP.
 - PHP es soportado por prácticamente todos los proveedores de hosting.
 - He estado aprendiendo PHP durante este año y ahora quiero implementar todos los conocimientos adquiridos en este proyecto.
- **Frontend: HTML, CSS, JavaScript**
 - **Descripción:** Tecnologías estándar para la construcción de interfaces de usuario web.
 - **HTML :** Para la estructura y el contenido de las páginas web.
 - **CSS :** Para el diseño y la presentación visual.
 - **JavaScript:** Para la interactividad y la lógica del lado del cliente, como la navegación dinámica del cuestionario.
 - **Justificación de la Elección:**
 - Son esenciales para cualquier sitio web actual.
 - Soportadas por todos los navegadores web.
 - Permiten un control detallado sobre la presentación y el comportamiento del lado cliente.

7.1.3 Frameworks y Bibliotecas

- **Backend Framework: Laravel**
 - **Descripción:** Laravel es un framework de aplicación web PHP de código abierto con una sintaxis expresiva y elegante. Proporciona una estructura robusta y herramientas para tareas comunes como enrutamiento, autenticación, sesiones y almacenamiento en caché.
 - **Alternativas Consideradas:**

- **Symfony:** Otro potente framework PHP, pero Laravel fue elegido por ser más sencilla e intuitiva para el desarrollador y su popularidad en la comunidad.
- **Justificación de la Elección:**
 - Laravel acelera el desarrollo con sus convenciones y herramientas integradas.
 - Ofrece una seguridad integrada contra los errores más comunes en la web.
 - Simplifica la interacción con la base de datos.
 - Permite crear vistas dinámicas de forma sencilla.
- **Frontend Framework: Bootstrap**
 - **Descripción:** Bootstrap es un popular framework de CSS, HTML y JavaScript para desarrollar sitios web responsivos y adaptados a móviles desde el inicio.
 - **Alternativas Consideradas:**
 - **Tailwind CSS:** Un framework CSS basado en utilidades que ofrece mucha flexibilidad, aunque puede requerir más tiempo al principio para crear diseños complejos, comparado con los componentes listos para usar de Bootstrap.
 - **React:** Una biblioteca de JavaScript para crear interfaces de usuario dinámicas. Aunque es más poderosa para aplicaciones interactivas, requiere más configuración y conocimiento técnico que Bootstrap, que se enfoca más en el diseño visual.
 - **Justificación de la Elección:**
 - Permite construir interfaces de usuario responsivas rápidamente ,usando componentes ya hechos y un sistema de columnas.
 - Ayuda a mantener una apariencia coherente en toda la aplicación.
 - Facilita la resolución de problemas y la personalización.
 - Incluye componentes JavaScript listos para usar, como modales y carruseles, que pueden ser útiles para mejorar la experiencia del usuario.

7.1.4 Herramientas de Desarrollo

- **Entorno de Desarrollo Integrado (IDE):** Visual Studio Code (VS Code) es el IDE elegido debido a su ligereza, extensibilidad, soporte integrado para Git y una amplia gama de extensiones que facilitan el desarrollo en PHP, HTML, CSS y JavaScript.
- **Servidor Web Local:** Se utilizó XAMPP para crear un entorno de desarrollo local Apache, MySQL y PHP. Laravel también ofrece php artisan serve para un servidor de desarrollo rápido.
- **Gestor de Dependencias PHP:** Composer, para gestionar las bibliotecas y dependencias del proyecto Laravel.
- **Sistema de Control de Versiones:** Git, para el seguimiento de cambios en el código . Se utilizará una plataforma como GitHub o GitLab para el repositorio remoto.
- **Navegador Web con Herramientas de Desarrollo:** Google Chrome (o Firefox) por sus robustas herramientas de desarrollo integradas, esenciales para la depuración de frontend y la inspección de elementos.

7.2 Planificación (metodología, temporalización – diagrama Gantt)

Esta sección explico como implemento la metodología de desarrollo y la planificación temporal para el proyecto "HairLife". Al ser un proyecto individual, adopte un enfoque ágil y flexible, permitiendo adaptaciones a medida que avanza el desarrollo.

7.2.1 Metodología de gestión del proyecto

He gestionado el proyecto con una metodología ágil adaptada, centrada en ir desarrollando y revisando las funciones poco a poco. Aunque no voy a seguir todos los pasos formales de métodos como Scrum ya que se trata de un proyecto individual, sí voy a aplicar sus ideas principales.

- **Priorización:** Las tareas se priorizarán en función de su importancia para los objetivos del proyecto y su dependencia con otras tareas.
- **Flexibilidad y Adaptabilidad:** Se mantendrá la capacidad de ajustar el plan y las prioridades según los descubrimientos y desafíos que surjan durante el desarrollo.
- **Auto-gestión y Seguimiento:** El progreso se monitorizará continuamente, revisando los objetivos y ajustando el plan de trabajo según sea necesario. Se utilizarán herramientas sencillas como una lista de tareas o una hoja de cálculo para visualizar el progreso de las tareas: Pendiente, En Proceso, Completado.

7.2.2 Temporalización

Se estima una duración total del proyecto de 8 meses pero como solo dispongo de 15 semanas he realizado solo una de las funcionalidades que se pretenden implementar en el futuro. A continuación, presento una distribución de tareas principales a lo largo de este periodo, organizada en fases. Esta temporalización es flexible y las duraciones son estimadas.

Fase	Tareas Principales	Duración (semanas)
Fase 1: Planificación y Diseño-	Definición detallada de requisitos y alcance. -Investigación y selección final de tecnologías. - Diseño de la arquitectura del sistema. - Diseño de la base de datos :Modelo Entidad-Relación, Esquema	2-3 semanas

Relacional.

- Diseño de la interfaz de usuario.

Fase 2: Desarrollo Backend - Configuración del entorno de desarrollo (Laravel, base de datos). 4-6 semanas

- Creación de modelos y migraciones de la base de datos :Usuarios, Cuestionarios, Preguntas, Productos, etc..
- Implementación de la lógica de negocio Controladores para registro/login, gestión de cuestionarios, lógica de recomendación.

Fase 3: Desarrollo Frontend - Implementación de las vistas principales : 3-5 semanas

Inicio/Registro, Creación de nick, Panel de Usuario, Cuestionario, Página de Producto, Página de Agradecimiento.

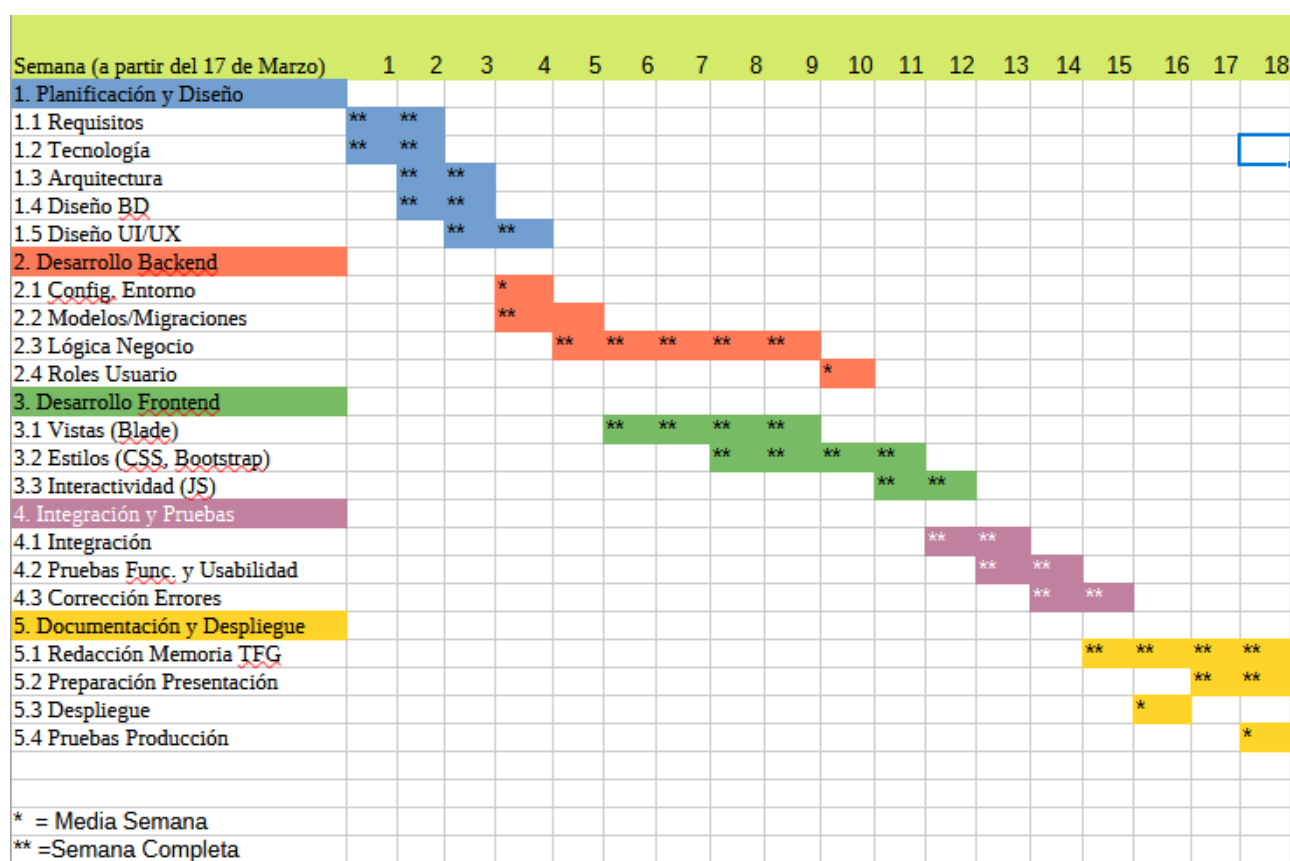
- Estilizado con CSS y Bootstrap.
- Implementación de la interactividad del cuestionario con JavaScript.

Fase 4: Integración y Pruebas- Integración de frontend y backend. 2-3 semanas

- Pruebas funcionales de todos los flujos de usuario.
- Pruebas de usabilidad y recolección de feedback ,si es posible con algunos usuarios de prueba.

	- Corrección de errores y ajustes.	
Fase 5: Documentación y Despliegue	<ul style="list-style-type: none"> - Redacción de la documentación del TFG. - Preparación de la presentación. - Despliegue de la aplicación en el servidor de clase. - Pruebas finales en el entorno de producción. 	2-3 semanas
Total Estimado		15 semanas

Diagrama de Gantt:



7.2.3 Seguimiento

Dado que este es un proyecto individual, el seguimiento se realizará principalmente mediante autogestión y autodisciplina. Se utilizarán las siguientes herramientas y métodos:

- **Lista de Tareas Detallada:** Se mantendrá una lista de tareas pendientes, en progreso y completadas, utilizando una herramienta digital simple o un cuaderno de notas.
- **Control de Versiones (Git):** Se utilizó Git para el control de versiones del código fuente, con commits regulares que reflejen el progreso y los cambios realizados. Esto también servirá como una forma de registro del trabajo. **Anexo**([FiguraB.2](#))
- **Reuniones con el Tutor:** Se realizaron reuniones periódicas con el tutor del proyecto para presentar los avances, discutir problemas y recibir orientación. Estas reuniones han servido para tener un de control del proyecto y para asegurar que el proyecto se mantiene en el camino correcto.

- **Documentación Continua:** Se irá elaborando la documentación del proyecto de forma paralela al desarrollo, lo que facilitará el seguimiento del progreso y la identificación temprana de cualquier desviación o problema.

Conjunto de necesidades que cubre el proyecto, relación de requisitos expresados por el cliente. Características de la plataforma / sistemas utilizados durante la implementación.

7.3.1 Requisitos de negocio

R1: Creación y Gestión de Perfil de Usuario Simplificado (Nick):

- **R1.1:** Los usuarios podrán crear un perfil rápido utilizando un "nickname" único para acceder a las funcionalidades personalizadas de la aplicación.
- **R1.2:** El sistema almacenará este nickname asociado al usuario para futuras interacciones y personalización.
- **R1.3:** Los usuarios podrán visualizar su perfil/en el panel para el usuario básico (show.blade.php).
- **R2: Cuestionario de Características Capilares y Preferencias:**
 - **R2.1:** La aplicación presentará un cuestionario dinámico diseñado para recopilar información sobre el tipo de cabello del usuario y sus preferencias .
 - **R2.2:** El cuestionario guiará al usuario a través de una serie de preguntas, que un experto ira desarrollando en el rol de administrador, cuando se realice.
 - **R2.3:** El sistema registrará las respuestas del usuario para su posterior análisis y para la generación de recomendaciones.
 - **R2.4:** Se permitirá a los usuarios re-tomar o modificar sus respuestas en el cuestionario, con la posibilidad de que se pre-rellenen las respuestas anteriores para facilitar la modificación.
- **R3: Generación y Presentación de Recomendaciones de Productos:**
 - **R3.1:** Basándose en las respuestas del cuestionario, el sistema seleccionará y presentará un producto recomendado que se ajuste al perfil del usuario.
 - **R3.2:** La página de recomendación (ver_producto.blade.php) mostrará detalles del producto, incluyendo nombre, descripción, marca, categoría y, si está disponible, una imagen y un enlace para su compra.

- **R3.3:** Se proporcionará una breve justificación de por qué se recomienda ese producto específico.
- **R4: Gestión de Contenidos (Rol Administrador):**
 - **R4.1:** Un usuario administrador podrá autenticarse en el sistema (login.blade.php).
 - **R4.2:** El administrador tendrá la capacidad de gestionar el catálogo de productos : añadir, editar, eliminar productos y sus detalles como nombre, descripción, marca, categoría, URL de compra, imagen.
 - **R4.3:** El administrador tendrá la opción de : modificar , eliminar y crear preguntas y opciones para generar recomendaciones más específicas y respuestas más profesionales.
- **R5: Interfaz de Usuario y Experiencia:**
 - **R5.1:** La aplicación presentará una interfaz de usuario clara, atractiva y fácil de usar en diferentes dispositivos .
 - **R5.2:** La navegación entre las diferentes secciones de la aplicación será intuitiva.

7.3.2 Requisitos Técnicos

Estos requisitos describen las características técnicas y las restricciones del sistema:

- **RT1: Plataforma Web:** La aplicación será accesible a través de un navegador web estándar.
- **RT2: Tecnologías de Desarrollo:**
 - **Backend:** Desarrollado utilizando PHP con el framework Laravel.
 - **Frontend:** Desarrollado utilizando HTML5, CSS3 y JavaScript. Se ha empleado el framework Bootstrap para el diseño responsivo y la maquetación.
 - **Base de Datos:** Se ha utilizado un sistema de gestión de bases de datos relacional, específicamente MySQL, para el almacenamiento de datos.
- **RT3: Seguridad:** Se implementarán medidas básicas de seguridad, como la protección contra inyección SQL, proporcionada en gran medida por el ORM de Laravel, y la validación de entradas del usuario. Las contraseñas de administrador se han gestionado para que estén debidamente almacenadas y de forma segura "hashed".
- **RT4: Despliegue:** La aplicación deberá ser desplegable en un entorno web compatible con PHP y MySQL.

- **RT5: Mantenimiento y Escalabilidad:** Aunque el proyecto inicial es de alcance limitado, la arquitectura debería permitir futuras expansiones y mantenimiento.
- **RT6: Gestión de Sesiones:** El sistema manejará sesiones de usuario para recordar información relevante, como el nickname o el estado del cuestionario.

7.4 Diseño de la aplicación (diagrama de BBDD, diagramas UML, prototipos).

Esta sección detalla el diseño de la aplicación "HairLife", abarcando la arquitectura de sus módulos, el modelo de datos, las clases principales, la interfaz de usuario y en un futuro posiblemente las interfaces externas .

7.4.1 Arquitectura de Módulos

El sistema "HairLife" se ha diseñado con una arquitectura modular para facilitar su desarrollo, mantenimiento y posible escalabilidad futura. Los principales módulos identificados son:

- **Módulo de Gestión de Usuarios:**
 - **Descripción:** Responsable de la autenticación de usuarios, la creación y gestión de perfiles de usuario, basados en "nicknames" para los clientes, y la gestión de roles como el usuario y el administrador.
 - **Interacciona con:** Módulo de Cuestionarios para asociar respuestas y recomendaciones a un usuario, Módulo de Base de Datos.
 - **Componentes clave:** LoginController.php, User.php ,para administradores, Usuario.php ,para usuarios/clientes , vistas login.blade.php, crearNick.blade.php.
 - **Añadidos clave:** Se ha añadido un script de PHP adicional para generar la encriptación (hasheo) de la contraseña para mayor seguridad.
- **Módulo de Cuestionarios y Recomendaciones:**
 - **Descripción:** Núcleo de la aplicación. Gestiona la presentación de cuestionarios, la recopilación de respuestas, el procesamiento de estas respuestas mediante una lógica de negocio para generar recomendaciones de productos personalizadas, y la visualización de dichas recomendaciones.

- **Interacciona con:** Módulo de Gestión de Usuarios ,para identificar al usuario, Módulo de Productos ,para obtener información de productos, Módulo de Base de Datos.
- **Componentes clave:** CuestionarioController.php, RecomendacionController.php, Cuestionario.php, Pregunta.php, Respuesta.php, Recomendacion.php, vistas ver_para_nick.blade.php, gracias.blade.php, ver_producto.blade.php.
- **Módulo de Presentación (Frontend):**
 - **Descripción:** Responsable de la interfaz de usuario, la interacción con el usuario y la visualización de la información. Incluye todos los elementos visuales y de navegación.
 - **Interacciona con:** Todos los demás módulos para presentar datos y capturar entradas.
 - **Componentes clave:** Todos los archivos .blade.php , CSS y JavaScript.

7.4.2 Modelado de Datos

La base de datos se ha diseñado utilizando MySQL. A continuación, se presenta una descripción de las tablas principales.Y el diagrama de bases de datos:

Anexo([Figura A.1](#))

Tabla usuarios

- **Descripción:** Esta tabla almacena la información de todos los usuarios que interactúan con el sistema, incluyendo tanto administradores como clientes. Cada usuario está identificado principalmente por un **nick** que es único para los clientes, mientras que los administradores usan un nombre y una contraseña para acceder al sistema. Además, se asocia un **role_id** para diferenciar los permisos o roles dentro de la plataforma.
- **Campos clave:**
 - id (PK): Identificador único de cada usuario.
 - name: Nombre completo o de usuario para administradores.
 - password: Contraseña cifrada para autenticación de administradores.
 - nick: Identificador único para clientes.
 - role_id: Clave foránea que indica el rol o nivel de acceso del usuario: administrador, cliente, etc...

Tabla productos

- **Descripción:** Contiene la información detallada de los productos capilares que se ofrecen o recomiendan en la plataforma. Cada producto tiene atributos que facilitan su identificación y descripción, así como un enlace para su compra y la ruta de la imagen asociada para mostrar visualmente el producto.
- **Atributos:**
 - idproducto (PK): Identificador único del producto.
 - nombre: Nombre del producto.
 - marca: Marca a la que pertenece el producto.
 - Descripcion detallada del producto y su modo de uso.
 - categoria: Categoría o tipo del producto : champu, acondicionador, mascarilla,aceite.
 - url_compra: Enlace directo para la compra del producto.
 - imagen_ruta: Ruta o URL de la imagen representativa del producto.

Tabla cuestionarios

- **Descripción:** Guarda la información de los cuestionarios disponibles en el sistema, que pueden ser usados para evaluar las necesidades o preferencias de los usuarios. Cada cuestionario tiene un título, una descripción breve y un estado que indica si está activo o inactivo.
- **Atributos:**
 - id (PK): Identificador único del cuestionario.
 - titulo: Nombre o título del cuestionario.
 - descripcion: Breve descripción del propósito del cuestionario.
 - estado: Indica si el cuestionario está activo, inactivo u otra condición.

Tabla preguntas

- **Descripción:** Contiene las preguntas que forman parte de los cuestionarios. Cada pregunta tiene un enunciado y un tipo de entrada que define cómo el usuario debe responder . Además, se clasifica según una categoría que puede ayudar a organizar las preguntas.
- **Atributos:**
 - id (PK): Identificador único de la pregunta.
 - enunciado: Texto de la pregunta que se mostrará al usuario.

- `tipo_input`: Tipo de respuesta esperado : 'radio', 'checkbox', 'text'.
- `categoria_pregunta`: Clasificación o categoría a la que pertenece la pregunta.

Tabla pregunta_opciones

- **Descripción:** Almacena las posibles opciones de respuesta para las preguntas que requieren selección múltiple o única . Cada opción tiene un texto visible para el usuario y un valor asociado .
- **Atributos:**
 - `id` (PK): Identificador único de la opción.
 - `pregunta_id` (FK): Clave foránea.
 - `texto_opcion`: Texto que se muestra como opción de respuesta.
 - `valor_opcion`: Valor interno que representa esa opción para cálculos o evaluaciones.
 -

Tabla cuestionario_pregunta :Tabla Pivote

- **Descripción:** Esta tabla relaciona las preguntas con los cuestionarios, permitiendo que una misma pregunta pueda estar en varios cuestionarios diferentes. Además, define el orden en el que las preguntas aparecerán dentro de cada cuestionario, asegurando una presentación lógica y coherente.
- **Atributos**
 - `cuestionario_id` (FK): Clave foránea.
 - `pregunta_id` (FK): Clave foránea.
 - `orden_pregunta`: Número que determina la posición de la pregunta dentro del cuestionario.

Tabla cuestionario_envios

- **Descripción:** Registra cada vez que un usuario completa y envía un cuestionario. Puede almacenar envíos de usuarios registrados o anónimos (en cuyo caso el `usuario_id` puede ser nulo). Se guardan datos relevantes como el nick del usuario y la fecha en que se realizó el envío, para llevar un historial de participación.
- **Campos clave:**
 - `id` (PK): Identificador único del envío.
 - `usuario_id` (FK, nullable): Referencia al usuario que completó el cuestionario, si está registrado.
 - `cuestionario_id` (FK): Referencia al cuestionario respondido.

- `nick_usuario`: Nick del usuario si es anónimo o no está registrado.
- `fecha_envio`: Fecha y hora en que se envió el cuestionario.

Tabla respuestas

- **Descripción:** Guarda las respuestas individuales que un usuario da a cada pregunta dentro de un envío específico de un cuestionario. Esto permite analizar detalladamente cada respuesta y asociarla con la pregunta correspondiente.
- **Campos clave:**
 - `id` (PK): Identificador único de la respuesta.
 - `envio_id` (FK): Referencia al envío del cuestionario al que pertenece la respuesta.
 - `pregunta_id` (FK): Referencia a la pregunta respondida.
 - `respuesta_valor`: Valor o texto de la respuesta dada por el usuario.

Tabla recomendaciones

- **Descripción:** Guarda la información de los productos que se recomiendan a los usuarios tras completar un cuestionario, basándose en sus respuestas. También incluye una justificación que explica por qué se sugiere cada producto, ayudando a personalizar la experiencia del usuario.
- **Campos clave:**
 - `id` (PK): Identificador único de la recomendación.
 - `envio_id` (FK): Referencia al envío del cuestionario asociado.
 - `producto_id` (FK): Referencia al producto recomendado.
 - `justificacion`: Texto que explica la razón o criterio de la recomendación.

7.4.3 Modelado de Clases

Para el desarrollo del proyecto, he seguido la arquitectura MVC: Modelo-Vista-Controlador que Laravel implementa de forma nativa. Esto me ha permitido estructurar el código de manera clara, separando la lógica de negocio, la interfaz y el manejo de datos. En esta sección explico los modelos que representan las principales entidades del sistema. Cada clase está ubicada en la carpeta `app/Models` y se encarga de gestionar la información correspondiente a una tabla de la base de datos, así como de definir relaciones entre entidades cuando es necesario.

Clase Usuario (app/Models/Usuario.php)

- **Atributos:** id, nombre, email, password, nick, rol.
- **Descripción:** Esta clase representa a cualquier usuario que accede al sistema, ya sea un administrador o un cliente. A través de este modelo se gestiona la autenticación, los datos personales y las relaciones con otras entidades como los envíos de cuestionarios. También permite distinguir los roles mediante el atributo rol, que es útil para aplicar restricciones o funcionalidades según el tipo de usuario.

Clase Producto (app/Models/Producto.php)

- **Atributos:** idproducto, nombre, marca, descripcion, categoria, url_compra, imagen_ruta.
- **Descripción:** Esta clase modela los **productos capilares** disponibles en la plataforma. Cada instancia de producto incluye tanto los datos informativos (nombre, marca, categoría), como los elementos multimedia y de acción (imagen y URL de compra). Este modelo también se relaciona con el sistema de recomendaciones.

Clase Cuestionario (app/Models/Cuestionario.php)

- **Atributos:** id, titulo, descripcion, estado.
- **Descripción:** Representa un cuestionario dentro del sistema. Cada cuestionario puede contener múltiples preguntas y estar activo o inactivo dependiendo del valor del atributo estado. Además, desde esta clase se pueden obtener fácilmente sus preguntas asociadas gracias a las relaciones entre modelos.
-

Clase Pregunta (app/Models/Pregunta.php)

- **Atributos:** id, enunciado, tipo_input, categoria_pregunta.
- **Descripción:** Este modelo representa una pregunta individual que puede formar parte de uno o varios cuestionarios. El atributo tipo_input indica la forma en la que debe responderse una pregunta, ya sea mediante texto, selección única o selección múltiple. Esto facilita que el cuestionario se muestre de manera dinámica en la interfaz del usuario. También se puede agrupar por categorías si se desea organizar las preguntas.
-

Clase PreguntaOpcion (app/Models/PreguntaOpcion.php)

- **Atributos:** id, pregunta_id, texto_opcion, valor_opcion.
- **Descripción:** Esta clase contiene las opciones de respuesta para las preguntas de tipo selección. El atributo valor_opcion puede utilizarse para cálculos o para condicionar las recomendaciones, mientras que texto_opcion es lo que el usuario

ve al responder. Este modelo está estrechamente ligado al modelo Pregunta mediante una relación belongsTo , lo que significa que cada opción de respuesta pertenece a una única pregunta .

Clase CuestionarioEnvio (app/Models/CuestionarioEnvio.php)

- **Atributos:** id, usuario_id, cuestionario_id, nick_usuario, fecha_envio.
- **Descripción:** Registra cada envío de cuestionario realizado por un usuario. Puede estar vinculado a un usuario registrado a través del campo usuario_id, o bien guardar un nick. También permite acceder a todas las respuestas asociadas a ese envío y se utiliza como punto de partida para generar recomendaciones.

Clase Respuesta (app/Models/Respuesta.php)

- **Atributos:** id, envio_id, pregunta_id, respuesta_valor.
- **Descripción:** Guarda la respuesta concreta que un usuario dio a una pregunta dentro de un cuestionario enviado. Está relacionada tanto con el modelo CuestionarioEnvio como con el modelo Pregunta, lo que permite analizar las respuestas en detalle, ya sea por envío, por pregunta o por usuario.

Clase Recomendacion (app/Models/Recomendacion.php)

- **Atributos:** id, envio_id, producto_id, justificacion.
- **Descripción:** Esta clase representa las recomendaciones personalizadas que el sistema genera para un usuario tras completar un cuestionario. Cada recomendación incluye el producto sugerido y una justificación, que puede ser generada automáticamente o personalizada, explicando por qué ese producto es adecuado según las respuestas del usuario.

Anexo([Figura A.2](#))

7.4.4 Interfaz de Usuario

El diseño de la interfaz de usuario se ha centrado en la simplicidad, la claridad y la facilidad de uso, con un estética moderna y atractiva.

- **Wireframes/Mockups :**
 - Se realizaron bocetos iniciales en papel para definir la estructura básica de las pantallas principales: página de inicio/creación de nick, vista del cuestionario, página de resultados/recomendación y panel de usuario.
 - *(Si tienes algunos, puedes describirlos brevemente o incluir una imagen representativa y referenciarla).*
- **Diseño Visual (High-Fidelity):**
 - **Paleta de Colores:** Se ha utilizado una paleta de colores basada en tonos de morado como por ejemplo: --purple-primary, --purple-secondary, --purple-dark , y colores neutros para garantizar una apariencia coherente y profesional, tal como se observa en los archivos CSS de las vistas (crearNick.blade.php, login.blade.php, ver_producto.blade.php).
 - **Tipografía:** Se emplea la fuente 'Montserrat' para el cuerpo del texto y 'Dancing Script' para títulos y elementos destacados, buscando legibilidad y un toque elegante.
 - **Diseño Responsivo:** La interfaz se ha desarrollado utilizando Bootstrap para asegurar su correcta visualización y funcionamiento en diversos dispositivos .
 - **Componentes Clave:**
 - **Página de Inicio/Creación de Nick (crearNick.blade.php):** Diseño limpio con un formulario central para la entrada del nickname y elementos visuales atractivos como tarjetas con efecto "flip".
 - **Página de Login (login.blade.php):** Presenta un formulario claro y conciso, acompañado de una imagen creada y editada con herramientas de inteligencia artificial, y mensajes de error o éxito bien definidos.
 - **Panel de Usuario (show.blade.php):** Ofrece una vista general de las opciones disponibles para el usuario, como acceder al cuestionario, utilizando un diseño basado en tarjetas interactivas con efecto "flip".
 - **Página de Cuestionario (ver_para_nick.blade.php):** Diseño tipo carrusel donde las preguntas se muestran de una en una, con navegación clara mediante botones titulados Anterior y Siguiente

para facilitar la cumplimentación. Se utiliza JavaScript para manejar la lógica de navegación entre preguntas .

- **Página de Recomendación (ver_producto.blade.php):** Presenta la información del producto recomendado de forma clara, incluyendo imagen, descripción, marca, y un enlace de compra. Incluye navegación para volver al cuestionario o al panel.
- **Página de Agradecimiento (gracias.blade.php):** Página intermedia que informa al usuario del éxito del envío y gestiona la redirección a la página de recomendación, incluyendo un indicador de carga.

Anexo([FiguraA3](#) , [FiguraA4](#) , [FiguraA5](#) , [FiguraA6](#) , [FiguraA7](#) , [FiguraA8](#))

7.4.5 Interfaces Externas

- **APIs de Terceros:** Actualmente, el proyecto no integra directamente APIs externas para su funcionalidad principal.
- **Potenciales Integraciones Futuras:** Se podría considerar la integración con APIs de proveedores de productos para obtener información actualizada o APIs de análisis para mejorar las recomendaciones.

7.5 Implementación y documentación (arquitectura y estructura de código, control de versiones, mapa web, estándares y validaciones utilizados)

7.5.1 Implementación

Ficha técnica resumen del proyecto:

Software

Componente	Nombre / Tecnología	Versión (Estimada/Común)	Servidor/SO (Desarrollo Local Típico)
FrameWork:	Laravel	10.x	N/A
IDE:	Visual Studio Code	Actual	Windows/Linux/macOS
Base de Datos:	MySQL	8.0.x	Localhost (ej. vía XAMPP, Docker)
Librerías:	Bootstrap	5.3.x	Navegador
	Bootstrap Icons	1.10.x	Navegador
	Google Fonts	N/A	Navegador

Lenguajes	PHP	8.1+	Apache/Nginx (vía XAMPP o php artisan serve)
	HTML5	N/A	Navegador
	CSS3	N/A	Navegador
	JavaScript (ES6+)	N/A	Navegador

Servidores

Server 1	(¿¿Base de Datos??) MySQL	8.0.x	windows
Server 2	(¿¿Web/Procesos ??) Apache/Nginx	N/A	windows

Componente	Nombre / Tecnología	Versión (Estimada/Común)	Servidor/SO (Desarrollo Local Típico)
Gestor Dependencias PHP	Composer	2.x	N/A
Control Versiones	Git	Actual	N/A
Repositorio Remoto	GitHub/GitLab (u otro)	N/A	N/A

7.5.1.1 Relación de módulos y archivos

Relación de módulos y archivos, desarrollo en entorno servidor:

Módulo	Directorio Principal	Nombre página / módulo de código	Descripción
Gestión de Usuarios y Autenticación	app/Http/Controllers	LoginController.php	Gestiona la lógica de inicio de sesión, cierre de sesión y el proceso de creación y guardado de nicknames para los usuarios.
	app/Models	Usuario.php, User.php	Usuario.php representa a los usuarios de la aplicación con "nick". User.php es el modelo estándar de Laravel para autenticación, potencialmente para administradores.
	resources/views	login.blade.php, crearNick.blade.php	Vistas para los formularios de inicio de sesión y creación de nicknames.

Panel Principal y Home	app/Http/Controllers	HomeController.php	Controla la visualización del panel principal del usuario: "user.dashboard" y otras secciones informativas o en desarrollo.
	resources/views/user_dashboard	show.blade.php, mi_pelo.blade.php, peinados_cortes.blade.php	Plantillas Blade para el panel de usuario, con acceso a las funcionalidades principales y secciones que estan todavia en desarrollo, marcadas como "Próximamente".
Cuestionarios y Recomendaciones	app/Http/Controllers	CuestionarioController.php, RecomendacionController.php	CuestionarioController maneja la visualización, procesamiento de envíos y lógica de recomendación de productos. RecomendacionController se encarga de mostrar la recomendación final.
	app/Models	Cuestionario.php, Pregunta.php, PreguntaOpcion.php, CuestionarioEnvio.php, Respuesta.php, Producto.php, Recomendacion.php	Modelos Eloquent : que definen la estructura y relaciones de las entidades de datos (cuestionarios, preguntas, productos, etc.) y facilitan la interacción con la base de datos.
	resources/views/cuestionarios	ver_para_nick.blade.php, gracias.blade.php	Plantillas para la visualización interactiva del cuestionario y la página de agradecimiento post-envío.
	resources/views/recomendaciones	ver_producto.blade.php	Plantilla para mostrar detalladamente el producto recomendado al usuario.
Rutas de la Aplicación	routes	web.php	Archivo que define todos los endpoints HTTP de la aplicación web, mapeando URLs a las acciones de los controladores correspondientes.

Estructura de Base de Datos	(Raíz del proyecto / database si fueran migraciones)	mi_pelo.sql	Script SQL que define el esquema completo de la base de datos mi_pelo, incluyendo tablas, columnas, relaciones y datos iniciales.
Configuración General	config	app.php, database.php, etc.	Directorio que contiene archivos de configuración globales para la aplicación Laravel como: servicios, base de datos, etc... Archivo crucial , para almacenar variables de entorno específicas como credenciales de base de datos, claves de API, y configuración de depuración.
Recursos Públicos Estáticos	(Raíz del proyecto)	.env (Archivo de entorno)	Directorio accesible vía enlace simbólico desde storage/app/public que contiene todas las imágenes utilizadas en el frontend de la aplicación(Hay que recordar que se debe ejecutar el comando “php artisan storage:link”).
	public/storage/imagenes	Varios archivos .jpg, .jpeg, .png,.webp	

7.5.1.2 Base de datos

- **Versión de la base de datos:** MySQL, versión 8.0.x o una versión compatible con las características utilizadas, como UTF8MB4 y InnoDB engine especificados en mi_pelo.sql.
- **Configuración de la base de datos:**
 - **Cadena de conexión (formato .env de Laravel):**

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=mi_pelo
DB_USERNAME=root
DB_PASSWORD=root
```

Base de datos: `mi_pelo`.

- En desarrollo local, he utilizado el usuario root de MySQL, aunque también creé un usuario específico para este proyecto cuando fue necesario. Para el entorno de producción, tengo en cuenta que lo ideal es utilizar un usuario con permisos mínimos, solo los necesarios para que la aplicación funcione correctamente.
Para hacer pruebas con usuarios dentro de la aplicación, inserté algunos directamente en el archivo `mi_pelo.sql`, como por ejemplo *aissa*, *carlota* o *ana*.
- En cuanto a la conexión con la base de datos, en local trabajé con `127.0.0.1` o `localhost`, como es habitual. En producción, la aplicación utilizaría la dirección IP o el nombre del servidor que proporcione el servicio de hosting.

7.5.1.3 Descripción de Procesos

A continuación, comento algunos de los procesos del sistema que me han parecido más relevantes o interesantes durante el desarrollo, tanto por su utilidad como por los retos que supusieron.

1. Proceso de creación de nick e inicio de sesión de usuario

Este proceso fue uno de los más importantes para personalizar la experiencia del usuario desde el principio. El flujo comienza cuando alguien intenta iniciar sesión desde el método `login` del `LoginController`. Ahí se comprueba si el usuario existe en la base de datos (`App\Models\Usuario`) y si la contraseña es correcta usando `Hash::check`.

Si el usuario todavía no tiene un nick asignado, se guarda en la sesión la URL a la que intentaba acceder para poder redirigirlo después (`url_intended_after_nick`) y se le envía a la vista `crearNick`, donde puede completar ese dato.

Esa vista (`crearNick.blade.php`) simplemente muestra el formulario para que el usuario introduzca su nick.

Una vez enviado el formulario, el método `guardarNick` se encarga de validar y guardar el nuevo nick tanto en la base de datos como en la sesión actual. A partir de ahí, se le redirige a su panel (`user.dashboard`).

Este sistema garantiza que todos los usuarios tengan un nick asignado, algo que usé en varios apartados de la aplicación para personalizar la navegación y los resultados. Además, cumple con el requisito funcional R1.1.

2. Proceso de Cumplimentación del Cuestionario y Generación de Recomendaciones:

ya que está directamente relacionado con el objetivo principal del proyecto: ofrecer recomendaciones personalizadas. El usuario accede al cuestionario desde el método `“mostrarParaNick”` en el `“CuestionarioController”`. Este método se encarga de cargar todas las preguntas del cuestionario junto con sus opciones, usando la relación `“$cuestionario→preguntas()→with('opciones')→get()”`.

Un detalle que me parece muy útil es que, si el usuario ya había hecho ese cuestionario antes, se pueden precargar sus respuestas anteriores. Así puede revisarlas, cambiarlas y volver a enviar, lo cual mejora bastante la experiencia.

El cuestionario se muestra en la vista `ver_para_nick.blade.php`, que tiene un diseño tipo carrusel hecho con JavaScript. Solo se muestra una pregunta a la vez, y las transiciones entre preguntas usan `transform` y `opacity` para crear un efecto de deslizamiento suave.

Los botones `“Anterior”` y `“Siguiente”`, permiten moverse fácilmente entre las preguntas, y el sistema ajusta su visibilidad según la posición actual. Por ejemplo, cuando el usuario llega a la última pregunta, el botón `“Siguiente”` desaparece y se muestra el botón `“Enviar”`.

Además, el código también tiene en cuenta si hay un hash en la URL o si Laravel detecta errores de validación, para que el usuario vea directamente la pregunta correspondiente y no tenga que buscarla manualmente.

Cuando el usuario envía el cuestionario, se activa el método `procesarEnvioParaNick`. Este realiza varios pasos importantes:

- Valida que las respuestas obligatorias estén completas.
- Guarda un nuevo registro en `CuestionarioEnvio` con todos los datos del envío.
- Crea las respuestas individuales vinculadas a ese envío.

- Procesa los criterios del usuario a partir de sus respuestas.
- Ejecuta la lógica de recomendación, a través del método `nuevaEstrategiaSeleccionProducto`, que filtra los productos según la categoría y características del cabello, y luego compara palabras clave entre las respuestas del usuario y los textos de los productos :nombre y descripción.

Por ejemplo, si una opción de respuesta tiene un valor como

"frizz_encrespamiento", se busca coincidencias con palabras como "frizz", "encrespamiento" o "anti-frizz".

Si se encuentra un producto adecuado, se guarda como recomendación para ese usuario y se le redirige a la página de agradecimiento (`gracias.blade.php`).

3. Visualización de la Recomendación y Flujo de Retorno:

- La página `gracias.blade.php` utiliza JavaScript para mostrar un loader y redirigir al usuario a `RecomendacionController@verRecomendacion` si hay un `$recomendacionId`.
- `RecomendacionController@verRecomendacion` carga la recomendación y su producto asociado (`Recomendacion::with('producto')->find($id)`) y los pasa a la vista `ver_producto.blade.php`. Es relevante cómo esta vista construye dinámicamente el enlace "Volver al cuestionario", permitiendo al usuario regresar al cuestionario con sus respuestas previas cargadas y anclado a la pregunta de filtro de categoría, facilitando un nuevo intento o modificación (R2.4).

7.5.1.4 Secciones de código relevantes

Explicar aquí:

1. `CuestionarioController::nuevaEstrategiaSeleccionProducto()`: *La Lógica de Recomendación.*

Aquí quiero destacar la función `nuevaEstrategiaSeleccionProducto`, que para mí fue una de las partes más interesantes de desarrollar, ya que concentra toda la lógica detrás de las recomendaciones de productos al usuario. Esta función recibe dos datos clave: la **categoría de producto** que el usuario ha elegido y un **array con todos los criterios** que hemos recogido de sus respuestas en el cuestionario.

El primer paso es un filtrado fundamental: solo consideramos productos que pertenezcan a la categoría seleccionada y que, además, sean adecuados para el **tipo de cabello** del usuario. Para esto, el sistema va más allá de una simple coincidencia de palabras; si el usuario dice tener "cabello liso", la función también buscará productos que mencionen "liso" o "cabello liso" en su nombre o descripción, ampliando así las posibilidades de encontrar una buena opción.

Si las respuestas del usuario se limitan a la categoría y el tipo de cabello, la función es directa y selecciona el primer producto que cumpla con esos dos filtros. Sin embargo, lo realmente potente viene cuando el usuario ha respondido a más preguntas. En ese caso, la función analiza en detalle los productos ya filtrados y calcula un **porcentaje de coincidencia** con esos criterios adicionales. Para que esta búsqueda sea más flexible y efectiva, convertimos los valores de opción que el usuario eligió en un conjunto de palabras clave relacionadas. Por ejemplo, si un usuario menciona "frizz_encrespamiento", el sistema buscará tanto "frizz" como "encrespamiento" o incluso "anti-frizz" en las descripciones de los productos.

El producto final recomendado será aquel que obtenga la **mayor puntuación** en este análisis de criterios adicionales y que, además, supere un porcentaje mínimo de coincidencia que hemos establecido. Pero no nos quedamos ahí: si, por alguna razón, ningún producto alcanza ese umbral ideal, la función está diseñada para no dejar al usuario sin una sugerencia. En ese escenario, el sistema selecciona automáticamente el producto que, aunque no haya llegado al porcentaje perfecto, ha logrado la **mayor puntuación posible** con los criterios adicionales. De esta forma, siempre intentamos ofrecer la mejor opción disponible.

Para que el usuario entienda la recomendación, la función también se encarga de generar una **justificación detallada** que explica por qué ese producto es el ideal para él, basándose en la categoría, el tipo de cabello y sus preferencias específicas. Esta lógica completa busca ofrecer una experiencia de recomendación lo más acertada y transparente posible.

2. . JavaScript para el Slider de Preguntas en ver_para_nick.blade.php:

El script se encarga de que solo se muestre una pregunta a la vez . Para lograr ese efecto de deslizamiento, utilizo propiedades CSS como transform y opacity, y me apoyo en requestAnimationFrame para asegurar que las transiciones sean suaves y visualmente atractivas.

La interacción es sencilla y directa: los botones "*Anterior*" y "*Siguiente*" permiten al usuario moverse cómodamente entre las preguntas. El JavaScript se asegura de que estos botones estén siempre en el estado correcto ,deshabilitados al inicio o al final, por ejemplo , guiando al usuario por el flujo del cuestionario. Un detalle importante es que, al llegar a la última pregunta, el botón "Siguiente" desaparece y en su lugar aparece el botón "*Enviar*", indicando claramente que es el momento de finalizar.

Además, he añadido una funcionalidad inteligente para mejorar la usabilidad: el cuestionario puede **iniciar en una pregunta específica**. Esto es útil en dos situaciones clave: si la URL incluye un fragmento , el script busca esa pregunta y la muestra directamente. La otra es si el sistema detecta errores de validación desde el backend tras un envío fallido; en ese caso, el carrusel se posiciona automáticamente en la primera pregunta con un error, lo que facilita mucho al usuario corregir sus respuestas.

En resumen, el JavaScript es fundamental para que el cuestionario se sienta dinámico y fácil de usar, haciendo que la navegación sea intuitiva y la experiencia, lo más fluida posible.

3. Empleo de Bootstrap en JavaScript

Aunque gran parte del JavaScript que he desarrollado es personalizado para controlar el carrusel, su integración con Bootstrap es fundamental, especialmente en cómo se manejan los **estilos y la apariencia de los componentes**.

Principalmente, el script manipula la visibilidad y el estado de elementos HTML que ya cuentan con **clases CSS de Bootstrap**. Por ejemplo, cuando muestro u oculto el botón "Enviar" o deshabilito los botones de navegación, lo hago sobre elementos que ya tienen clases como btn, btn-primary, btn-secondary, etc.. Esto me permite confiar en que Bootstrap se encargará de darles el estilo y la apariencia visual

correcta automáticamente, sin que yo tenga que definir esos estilos en el JavaScript.

Además, aunque mi CSS personalizado define variables de color para personalizar los estilos de enfoque (:focus) o los estados marcados (:checked), estas personalizaciones se aplican directamente a las **clases estándar de formulario de Bootstrap**, como form-check-input o form-control. Esto asegura que la apariencia general del formulario sea coherente con el diseño de HairLife, pero partiendo de la base robusta que ofrece Bootstrap para los elementos de UI.

Por último, los **íconos de Bootstrap Icons** son clave en los botones de navegación. Mi JavaScript solo se encarga de que los botones funcionen, mientras que la visualización de los íconos se gestiona automáticamente gracias a que la librería de Bootstrap Icons está correctamente enlazada en el <head> del documento.

En esencia, mi JavaScript se ocupa de la funcionalidad y la interactividad del carrusel, mientras que Bootstrap me proporciona el esqueleto visual y los estilos predeterminados para que todos los elementos de la interfaz tengan una apariencia profesional y cohesiva.

4. Manejo de Sesión y Flujo de Nick en LoginController.php:

Para mí, era fundamental asegurar que todos los usuarios tuvieran un "nick" de display antes de acceder al panel principal, ya que esto es un requisito del proyecto. Para ello, he diseñado un flujo que guía a los usuarios que inician sesión y no tienen uno, a crearlo de forma obligatoria pero fluida.

¿Cómo funciona?

Todo empieza justo después de un login exitoso, gestionado por el método login en el LoginController. En este punto, compruebo si el campo nick del usuario es nulo. Si lo es, activo un par de señales importantes en la sesión del usuario:

- Establezco session(['necesita_completar_nick_display' => true]). Esto me sirve como un indicador de que el usuario tiene esta tarea pendiente.
- También guardo la URL a la que el usuario intentaba acceder originalmente (su panel principal o \$dashboardUrl) en

`session(['url_intended_after_nick' => $dashboardUrl])`). Esto es clave para que, una vez que haya creado su nick, pueda ir directamente a donde quería.

Una vez configuradas estas señales, redirijo al usuario a la ruta `crear.nick`. Esta ruta lleva a la vista `crearNick.blade.php`, que es una interfaz sencilla donde el usuario puede ingresar y confirmar el nick que desea utilizar.

Cuando el usuario envía este formulario, la lógica se maneja en el método `guardarNick` del `LoginController`. Aquí, me encargo de:

- Actualizar el nick del usuario en la base de datos.
- Actualizar también el nick en la sesión actual del usuario.
- Finalmente, y esto es muy importante para la fluidez de la experiencia, redirijo al usuario a la `url_intended_after_nick` que había guardado. Esto significa que el usuario, tras crear su nick, es llevado directamente a la sección que intentaba visitar, sin interrupciones ni desvíos.

Este flujo, aunque es un paso adicional para algunos usuarios, asegura que se cumpla el requisito de que todos operen con un nick, y lo gestiona de una forma que considero, es bastante sencilla y comprensible para el usuario. Adicionalmente, he implementado una función privada llamada `checkAuth()`, para que el usuario no pueda acceder a la página sin haber iniciado sesión.

7.5.2 Documentación

7.5.2.1 Documentación técnica de desarrollo

Para gestionar los distintos aspectos de mi aplicación, me he apoyado en las convenciones de Laravel para las configuraciones.

La configuración principal se maneja a través del archivo `.env`, ubicado en la raíz del proyecto. Este archivo es fundamental porque es donde defino las variables de entorno que son cruciales para el funcionamiento de la aplicación. Aquí establezco detalles como los parámetros de conexión a la base de datos (`DB_HOST`, `DB_DATABASE`, `DB_USERNAME`, `DB_PASSWORD`), la URL base de la aplicación (`APP_URL`), si el modo de depuración está activo (`APP_DEBUG`), y la clave de encriptación única de la aplicación (`APP_KEY`), entre otras variables importantes.

Además del .env, utilizo los archivos de configuración más detallados que se encuentran en el directorio “config/” de Laravel. Los que considero más relevantes para este proyecto son:

- **config/app.php:** Aquí se encuentra la configuración general de la aplicación, incluyendo la definición de proveedores de servicios y alias, que son esenciales para el funcionamiento de Laravel.
- **config/database.php:** En este archivo, defino las configuraciones para las distintas conexiones a la base de datos. Especifico la conexión MySQL y me aseguro de que lea las credenciales directamente desde el archivo .env.

Pasos para la Construcción del Entorno de Desarrollo

Para poner en marcha este proyecto, he seguido una serie de pasos para construir el entorno de desarrollo.

Software Requerido

Primero, es necesario contar con el siguiente software:

- **PHP** (versión 8.1 o superior).
- **Composer** para la gestión de dependencias PHP.
- **MySQL** como sistema de gestión de bases de datos.
- **Git** para el control de versiones.
- Un **servidor web local**, como XAMPP, o simplemente el servidor incorporado de Laravel (php artisan serve).

Pasos de Construcción

Los pasos que he seguido para la construcción del entorno son los siguientes:

1. **Crear el Proyecto Laravel:** Inicialmente, creé un nuevo proyecto de Laravel en mi entorno local.

Bash

“composer create-project laravel/laravel HairLife”

2. **Acceder al Directorio del Proyecto:** Una vez creado, navegué al directorio del proyecto HairLife.

Bash

“cd HairLife”

3. **Configurar el Archivo de Entorno (.env):** Edité el archivo .env para establecer los valores de configuración específicos del proyecto. Esto incluye APP_NAME, APP_URL, y crucialmente, las credenciales de la base de datos: DB_CONNECTION, DB_HOST, DB_PORT, DB_DATABASE (que debe ser mi_pelo), DB_USERNAME, y DB_PASSWORD.
4. **Ajustar .gitignore:** Modifiqué el archivo .gitignore para asegurar que el archivo .env no fuera ignorado por el control de versiones.
5. **Clonar el Repositorio de HairLife:** Luego de configurar el .env, cloné el repositorio de HairLife en un directorio temporal y moví los archivos necesarios a mi nuevo proyecto Laravel, **excluyendo el archivo .env que ya había configurado.**
6. **Crear Base de Datos:** Me aseguré de que la base de datos mi_pelo ya existiera en el servidor MySQL.
7. **Importar Esquema y Datos:** Utilicé un cliente MySQL para importar el archivo mi_pelo.sql, que contiene el esquema y los datos necesarios para la aplicación.

Bash

mysql -u TU_USUARIO -p mi_pelo < ruta/a/mi_pelo.sql

8. **Instalar Dependencias PHP:** Ejecuté composer install para asegurar que todas las dependencias de PHP definidas en composer.json estuvieran instaladas correctamente.

Bash

“composer install”

9. **Crear Enlace de Almacenamiento:** Generé el enlace simbólico necesario para que los archivos públicos (como imágenes) almacenados en storage/app/public fueran accesibles desde public/storage.

Bash

“php artisan storage:link”

10.Servir la Aplicación: Finalmente, inicié el servidor de desarrollo de Laravel. La aplicación está ahora disponible en <http://127.0.0.1:8000> por defecto.

Bash

“php artisan serve”

Estructura de Directorios

Anexo([FiguraB.1](#) , [FiguraB.1](#) , [FiguraB.1](#) , [FiguraB.1](#))

Aquí presento las carpetas principales de Laravel que son más relevantes para la organización de mi proyecto:

- **app/:** Este directorio es el núcleo de la aplicación, albergando toda la lógica central.
 - **Http/Controllers/:** Aquí residen los controladores, como LoginController.php, HomeController.php, CuestionarioController.php, y RecomendacionController.php, encargados de gestionar las peticiones HTTP.
 - **Models/:** Contiene los modelos de Eloquent, tales como Usuario.php, Producto.php, Cuestionario.php, Pregunta.php, entre otros, que representan las tablas de la base de datos.
 - **Providers/:** Aquí se encuentra AppServiceProvider.php, un proveedor de servicios clave para la inicialización de la aplicación.
- **config/:** Contiene todos los archivos de configuración de la aplicación, como app.php y database.php.
- **database/:** Aunque en este proyecto el esquema se importa directamente desde un archivo SQL, en un flujo de trabajo más estándar de Laravel, este directorio contendría las migraciones y seeders para la base de datos.
- **public/:** Es el punto de entrada público de la aplicación “index.php” y donde se alojan los recursos públicos. Dentro de este, el directorio storage es en realidad un enlace simbólico a storage/app/public.
- **storage/imagenes/:** Específicamente, esta carpeta contiene las imágenes de productos, logos y fondos que utiliza la aplicación.

- **resources/**: Contiene los recursos que no son directamente accesibles desde la web.
- **views/**: Aquí se encuentran todas las plantillas Blade de la aplicación, organizadas en subdirectorios como cuestionarios/, recomendaciones/, user_dashboard/, y archivos individuales como login.blade.php y crearNick.blade.php.
- **routes/**: Contiene los archivos que definen las rutas de la aplicación, siendo web.php el más relevante para las rutas web.
- **storage/app/public/**: Esta es la ubicación real donde se almacenan los archivos que deben ser accesibles públicamente, como las imágenes de los productos.

Diagrama de Red / Conectividad de Servidores

Para entender cómo se comunican las distintas partes de la aplicación, estoy analizando la conectividad en el entorno de desarrollo local y en un entorno de producción típico.

Entorno de Desarrollo (Local)

En mi entorno de desarrollo, todo se ejecuta de forma local en mi máquina. El flujo de comunicación es el siguiente:

1. Cliente (Navegador Web)

Accedo a la aplicación desde el navegador, normalmente ingresando una URL como `http://localhost:8000`.

2. Servidor Web Local

Utilizo herramientas como **XAMPP** (con Apache) o el **servidor integrado de Laravel** para gestionar las peticiones web.

3. Motor PHP

El servidor web redirige la solicitud al **motor PHP**, que se encarga de ejecutar el código de la aplicación Laravel.

4. Servidor MySQL (Base de Datos)

Laravel interactúa con una base de datos **MySQL**, que suele estar corriendo en `localhost:3306`.

Todo este flujo ocurre en la misma máquina, lo que me facilita mucho el desarrollo, las pruebas y la depuración.

Entorno de Producción (Típico en Hosting Compartido/VPS):

En un entorno de producción, el flujo cambia un poco, ya que la aplicación está desplegada y accesible públicamente:

1. Cliente (Navegador Web)

El usuario accede a la aplicación desde internet, usando una URL como “https://miapp.com”.

2. Servidor Web Público

El servidor web ,como **Apache** o **Nginx**, alojado en un **hosting compartido** o **VPS**, recibe la petición del navegador.

3. Motor PHP

Este servidor dirige la solicitud al **motor PHP** ,como PHP-FPM, que ejecuta el código de Laravel.

4. Servidor MySQL (Base de Datos)

Laravel se conecta a una base de datos **MySQL**. Dependiendo de la configuración:

- Puede estar en el mismo servidor.
- O puede ser un servicio aparte, como una base de datos administrada o un servidor dedicado.

Este esquema permite que la aplicación escale y sea más segura y eficiente en entornos reales.

Usuarios/Claves Utilizados

Para las pruebas y el funcionamiento de la aplicación, he configurado ciertos usuarios y claves.

Usuarios de la aplicación (Tabla usuarios en mi_pelo.sql):

En mi base de datos, he predefinido los siguientes usuarios para pruebas:

- **Nombre:** aissa, **Clave** (texto plano para referencia): aissa, **Rol:** 0
- **Nombre:** carlota, **Clave** (texto plano para referencia): carlota, **Rol:** 0
- **Nombre:** ana, **Clave** (texto plano para referencia): ana, **Rol:** 0

Es importante destacar que, aunque aquí menciono las claves en texto plano para facilitar la referencia, en la base de datos estas claves se almacenan hasheadas, como se puede ver en las sentencias INSERT del archivo `mi_pelo.sql` y en el mutador correspondiente en `Usuario.php`.

Usuario de Base de Datos MySQL:

Las credenciales para el usuario de MySQL las tengo definidas en el archivo `.env`, usando las variables `DB_USERNAME` y `DB_PASSWORD`. Para desarrollar en local, lo normal es que use root sin contraseña, o que me cree un usuario específico para esto. En mi caso, he configurado MySQL con root y la contraseña root.

Fuentes de la Aplicación

Todas las fuentes de la aplicación, incluyendo el código fuente del proyecto Laravel como: controladores, modelos, vistas, rutas, el script SQL `mi_pelo.sql`, y cualquier otro archivo esencial para su funcionamiento, se suministrarán en formato digital. Lo haré a través de un archivo ZIP o proporcionando acceso al repositorio Git.

7.5.2.2 Documentación técnica para explotación

Para el despliegue de versiones, tenemos una sección dedicada a ello, la sección 8.1 de este documento.

Aquí te resumo de los **pasos clave para el despliegue**:

- **Configurar el entorno del servidor:** Hay que asegurarse de tener un entorno compatible con PHP, MySQL y un servidor web.
- **Transferir los archivos:** Simplemente, subir los archivos del proyecto al servidor.
- **Ajustar el archivo .env:** Esto es crucial. Hay que configurarlo para el entorno de producción, es decir, las credenciales de la base de datos, `APP_ENV=production`, `APP_DEBUG=false`, y generar una nueva `APP_KEY`.
- **Instalar dependencias PHP:** Ejecutar `composer install --optimize-autoloader --no-dev`.

- **Configurar la base de datos:** Preparar la base de datos en producción y luego importar `mi_pelo.sql` o ejecutar las migraciones.
- **Crear el enlace de almacenamiento:** Ejecutar `php artisan storage:link`.
- **Optimizar la aplicación:** Para mejorar el rendimiento, corremos `php artisan config:cache` y `php artisan route:cache`.
- **Ajustar permisos:** Es importante configurar correctamente los permisos de los directorios `storage` y `bootstrap/cache`.
- **Configurar el servidor web:** Finalmente, hay que asegurarse de que el servidor web esté sirviendo desde el directorio `public`.

7.5.2.3 Documentación de usuario

Manuales de usuario / Guías rápidas:

Guía Rápida para Usuarios de HairLife

Bienvenido/a a HairLife, tu asistente personal para el cuidado del cabello. Sigue estos pasos para comenzar:

1. Accede y Crea tu Nick:

- Si es tu primera vez o no has iniciado sesión, ve a la página de **Login**.
- Si ya tienes credenciales de acceso (Nombre y Clave), ingrásalas.
- Si eres un usuario nuevo o no tienes un nick de perfil, serás dirigido a la página **"Crear Nickname"**. Ingresas el nick que te gustaría usar y haz clic en "Acceder". Este nick te identificará en la plataforma.

2. Tu Panel Personal (`show.blade.php`):

- Una vez dentro, verás tu panel personal. Desde aquí podrás:
 - Acceder al **"Cuestionario"** para obtener recomendaciones de productos.
 - Explorar "Mi Pelo" y "Peinados y Cortes", estas secciones estarán disponibles próximamente.

3. Responde el Cuestionario (`ver_para_nick.blade.php`):

- Haz clic en la opción "Cuestionario".
- Se te presentarán preguntas sobre tu tipo de cabello, preocupaciones y preferencias.
- Usa los botones **"Anterior"** y **"Siguiente"** para navegar entre las preguntas.

- Responde cada pregunta seleccionando una opción o escribiendo tu respuesta.
- Cuando llegues a la última pregunta, el botón **"Siguiente"** cambiará a **"Enviar Respuestas"**. Haz clic en él.

4. Recibe tu Recomendación:

- Tras enviar el cuestionario, verás una página de **"¡Gracias!"** (gracias.blade.php).
- Si hemos encontrado un producto para ti, verás un indicador de carga y serás redirigido automáticamente a la página de tu producto recomendado.
- En la página del producto (ver_producto.blade.php), encontrarás:
 - Nombre, marca, categoría y foto del producto.
 - Una descripción detallada y modo de uso.
 - Un enlace para "Ver o Comprar Producto" .
 - Un botón de **"Volver al cuestionario"**. Si lo usas, podrás modificar tus respuestas anteriores , se cargarán tus datos previos, excepto la categoría de producto, para que puedas elegir otra si lo deseas.
- Si no se encontró un producto específico, la página de "¡Gracias!" te informará y te dará opciones para volver a intentarlo o regresar a tu panel.

5. Navegación y Cierre de Sesión:

- Desde la página del producto o tu panel, puedes encontrar opciones para "Volver a Mi Panel" o "Cerrar Sesión" (/logout).

Anexo([login](#), [crear nick](#), [Panel Usuario](#), [cuestionario](#), [loading](#) , [Producto](#))

7.6 Plan de pruebas

Esta es una descripción de los planes de pruebas diseñados y que se utilizaron en el proyecto "HairLife" para asegurar su correcto funcionamiento y la calidad del software entregado.

7.6.1 Pruebas unitarias

Las pruebas unitarias se centran en verificar la correcta funcionalidad de las unidades de código más pequeñas y aisladas, como métodos de una clase o funciones específicas. En el contexto de Laravel, estas pruebas se suelen escribir utilizando PHPUnit.

Componentes Clave y Enfoque de Pruebas Unitarias:

Modelos y Controladores:

Cuando hablamos de la calidad de nuestro código, sobre todo en la parte del backend, me gusta centrarme en dos pilares fundamentales: los Modelos y los Controladores. Son el corazón de la lógica de negocio y, por tanto, donde ponemos más atención en las pruebas.

Los Modelos (app/Models/Usuario.php, app/Models/Producto.php)

Aquí me aseguro de que todo funciona como esperamos. Dos puntos clave para mí:

- **Usuario::setClaveAttribute():** Este método es vital, ya que es el encargado de hashear las contraseñas de los usuarios. Mi forma de verificarlo fue sencilla:
 - Creo una instancia de Usuario, le asigno una contraseña cualquiera y luego compruebo que el atributo clave guardado *no es igual* al valor original, porque está hasheado.
 - Lo más importante es que Hash::check() devuelva true al comparar la contraseña original con el valor hasheado. Si no, algo está fallando.
- **Relaciones Eloquent:** Aunque la verificación profunda de las relaciones se hace mejor en pruebas de integración, a nivel unitario me interesa ver que los métodos de relación :HasMany, BelongsTo, existan y devuelvan el tipo de objeto que esperamos. Por ejemplo, en Usuario.php, compruebo que \$usuario->cuestionarioEnvios() me devuelva una instancia de Illuminate\Database\Eloquent\Relations\HasMany.

Los Controladores (Lógica de Negocio Aislada - Ej: `app/Http/Controllers/CuestionarioController.php`)

Aquí es donde entra la lógica más compleja, y por eso, los controladores son candidatos perfectos para las pruebas unitarias, especialmente los métodos que encapsulan reglas de negocio.

- **CuestionarioController::nuevaEstrategiaSeleccionProducto():** Este método es un ejemplo perfecto. Su lógica de selección de productos es crucial, así que lo pongo a prueba a fondo:
 - Simulo unos `$criteriosUsuario` y una `$categoriaProductoElegida` que, sé de antemano, deberían llevarme a un producto específico de mi base de datos de prueba. Luego, verifico que el método me devuelve justo ese producto y, muy importante, una justificación que tenga sentido.
 - También le meto criterios que no deberían encontrar ningún producto. Aquí, espero que el método me devuelva `null` como producto y que me dé una justificación adecuada.
 - Pruebo con criterios que cumplen los filtros primarios :`categoría`, tipo de `cabello`, pero fallan en los filtros adicionales más allá de un umbral. En este caso, si la lógica lo contempla, espero que se recomiende un producto de *fallback*.
 - No me olvido de probar cómo reacciona el método si `$criteriosUsuario` está vacío o si `$categoriaProductoElegida` no es válida. La robustez es clave.

Otras Lógicas Aislables

Finalmente, cualquier otra clase de servicio o función auxiliar, que contenga lógica de negocio pura también es una candidata ideal para las pruebas unitarias. Son las piezas que podemos aislar y probar a conciencia, garantizando que funcionan perfectamente por sí mismas antes de integrarlas en el conjunto.

7.6.2 Pruebas funcionales

Pruebas de Flujo de Usuario y UI

Para asegurar que la experiencia del usuario sea fluida y que la interfaz funcione como debe, he definido una serie de casos de prueba. Aquí te detallo cómo los planteo y qué resultados espero en cada etapa clave de la aplicación.

Registro/Login y Creación de Nick

Esta es la puerta de entrada a la aplicación, así que me aseguro de que el flujo sea impecable.

- **Caso 1.1: Intentar login con credenciales válidas de un usuario existente sin nick.**
 - **Resultado Esperado:** Lo que busco es que se redirija a la página de creación de nick "crearNick.blade.php". Además, que se guarde la "url_intended_after_nick" en sesión para saber adónde ir después.
- **Caso 1.2: Enviar el formulario de creación de nick con un nick válido.**
 - **Resultado Esperado:** Aquí quiero que el nick se guarde correctamente tanto en la base de datos como en la sesión. Y, por supuesto, que el usuario sea redirigido al panel de usuario "show.blade.php".
- **Caso 1.3: Intentar login con credenciales válidas de un usuario existente con nick.**
 - **Resultado Esperado:** Para este caso, espero una redirección directa al panel de usuario. Sin intermediarios.
- **Caso 1.4: Intentar login con credenciales inválidas.**
 - **Resultado Esperado:** La idea es que el usuario permanezca en la página de login "login.blade.php" y se muestre un mensaje de error claro y apropiado.
- **Caso 1.5: Logout.**
 - **Resultado Esperado:** Tras cerrar sesión, la sesión debe ser destruida y el usuario redirigido a la página de login.

Navegación por el Panel de Usuario (show.blade.php)

El panel es el centro de control del usuario, así que me aseguro de que la navegación y la información clave estén a la vista.

- **Caso 2.1: Acceder al panel de usuario.**
 - **Resultado Esperado:** Aquí tiene que mostrarse el saludo personalizado con el displayNick del usuario. Y, por supuesto, las tarjetas de "Mi Pelo", "Cuestionario" y "Peinados y Cortes" deben ser claramente visibles.
- **Caso 2.2: Hacer clic en la tarjeta "Cuestionario".**

- **Resultado Esperado:** Espero que se redirija a la vista del cuestionario “ver_para_nick.blade.php” para el cuestionario que esté activo en ese momento.
- **Caso 2.3: Verificar efecto "flip" en tarjetas "Mi Pelo" y "Peinados y Cortes".**
 - **Resultado Esperado:** Al pasar el ratón por encima, las tarjetas deben girar mostrando el mensaje "PRÓXIMAMENTE". Es un detalle visual importante.

Cumplimentación del Cuestionario “ver_para_nick.blade.php”

El cuestionario es clave para las recomendaciones, así que el flujo aquí debe ser robusto y fácil de usar.

- **Caso 3.1: Navegación entre preguntas usando los botones "Anterior" y "Siguiente".**
 - **Resultado Esperado:** El slider de preguntas tiene que funcionar correctamente, mostrando la pregunta adecuada. El botón "Anterior" debe estar deshabilitado en la primera pregunta, y el botón "Siguiente" tiene que cambiar a "Enviar Respuestas" cuando se llega a la última pregunta.
- **Caso 3.2: Envío del cuestionario sin completar campos obligatorios.**
 - **Resultado Esperado:** El usuario debe permanecer en la vista del cuestionario, y se tienen que mostrar mensajes de error de validación claros junto a las preguntas correspondientes. Si es posible, la pregunta con el primer error debería ser la visible.
- **Caso 3.3: Envío del cuestionario con todas las respuestas válidas.**
 - **Resultado Esperado:** Una vez completado, se redirige a la página de agradecimiento “gracias.blade.php” con un mensaje de éxito y el “id_cuestionario_procesado” en sesión.
- **Caso 3.4: Acceder al cuestionario con el parámetro envio_previo=ID en la URL.**
 - **Resultado Esperado:** El cuestionario debe cargarse con las respuestas del envío previo ya rellenadas, con la única excepción de la pregunta de filtro de categoría de producto.

Visualización de Recomendaciones

Este es el objetivo final del cuestionario, así que me aseguro de que la recomendación se muestre de forma correcta.

- **Caso 4.1: Tras enviar el cuestionario y obtener una recomendación, la página `gracias.blade.php` muestra el loader.**
 - **Resultado Esperado:** Después de un breve retardo (para que la lógica de la recomendación haga su trabajo), el sistema debe redirigir a la página de visualización del producto `ver_producto.blade.php`.
- **Caso 4.2: La página `ver_producto.blade.php` muestra los detalles del producto recomendado.**
 - **Resultado Esperado:** Toda la información del producto y la justificación de la recomendación tienen que mostrarse correctamente. Y si el producto tiene URL de compra, el enlace debe ser visible.
- **Caso 4.3: En `ver_producto.blade.php`, hacer clic en "Volver al cuestionario".**
 - **Resultado Esperado:** Se redirige a `ver_para_nick.blade.php` con el parámetro `envio_previo` y, si los datos están disponibles, el ancla `"ID_PREGUNTA_FILTRO"`, pre-rellenando el cuestionario.
- **Caso 4.4: Intentar acceder a una URL de recomendación con un ID inválido o de un producto inexistente.**
 - **Resultado Esperado:** Aquí espero que se redirija al dashboard del usuario o a `crear_nick` con un mensaje de error, tal y como tengo implementado en `RecomendacionController@verRecomendacion`.

Responsividad de la Interfaz

La aplicación debe verse bien y ser funcional en cualquier dispositivo.

- **Caso 5.1: Visualizar todas las páginas clave : `login`, `crearNick`, `show`, `ver_para_nick`, `ver_producto`, `gracias`, en diferentes tamaños de pantalla.**
 - **Resultado Esperado:** El diseño tiene que adaptarse correctamente a cada tamaño, el contenido debe ser perfectamente legible y todos los elementos interactivos tienen que funcionar sin problemas.

Además, las tarjetas decorativas en crearNick.blade.php deberían ocultarse en pantallas pequeñas, tal como está definido en el CSS.

7.6.3 Pruebas finales de usuario

No se realizaron pruebas finales de usuario.

8 Implantación

En este apartado, explicaré cómo he puesto en marcha mi aplicación web "HairLife". Como el proyecto se ha desarrollado en el instituto, he realizado la implantación de forma local, simulando un entorno real al que se podría acceder desde cualquier ordenador de la red del IES Virgen de la Paloma. Esto me ha servido para comprobar que todo funciona correctamente en un entorno similar al real.

8.1. Qué Necesitamos para que Funcione

"HairLife" necesita ciertas herramientas y programas para poder trabajar. Aquí te detallo los principales:

- **Sistema Operativo del Servidor:** La aplicación está instalada en mi propio ordenador, que usa Windows 11.
- **Servidor Web:** He utilizado XAMPP v3.3.0 , que ya incluye Apache, MySQL y PHP juntos.
- **Base de Datos:** Uso MySQL versión 8.0.38 , para guardar toda la información de la aplicación: los usuarios, los cuestionarios, las preguntas, los productos y las recomendaciones.
- **Lenguaje de Programación y Herramienta de Desarrollo (Framework):**
 - La parte "inteligente" de la web (el backend) está programada en PHP 8.2.12 .
 - Para hacer el desarrollo más rápido y organizado, he usado Laravel Framework 12.12.0 . Laravel es como una caja de herramientas para PHP que me ayuda a manejar la base de datos, las páginas web y las sesiones de usuario de forma estructurada.

8.2. Despliegue de la aplicación

- Una vez obtenidos los ficheros del proyecto, el primer paso fue configurar el archivo de entorno .env para establecer las credenciales de la base de datos y otras configuraciones específicas del entorno.
- Posteriormente, las dependencias de PHP se instalaron mediante composer install.
- La base de datos se inicializó con php artisan migrate , creando el esquema de tablas necesario.

- Finalmente, se creó el enlace simbólico para el almacenamiento de archivos con `php artisan storage:link`, asegurando que los archivos subidos, en este caso imágenes, fueran accesibles públicamente.
- **Acceso a la Aplicación :**
 - Para iniciar la aplicación y acceder a ella desde el navegador, utilicé el servidor de desarrollo de Laravel con el siguiente comando:
Bash

`php artisan serve --host=0.0.0.0 --port=8080`
 - Al especificar `--host=0.0.0.0`, el servidor escucha en todas las interfaces de red de mi equipo. Esto es fundamental, ya que permite que la aplicación no solo sea accesible desde mi propio ordenador, sino también desde cualquier otro dispositivo conectado a la misma red local ,como la red del instituto durante la defensa.
 - Elegí el puerto 8080 para la conexión.
 - Durante la demostración en clase, se podrá acceder a la aplicación desde otros equipos usando la dirección IP que mi ordenador tenga en ese momento en la red, la cual mostraré con el comando `ipconfig`.

8.3. Mantenimiento de la aplicación

Al ser un proyecto para demostración, el mantenimiento se enfoca en la gestión del contenido y la preparación para una posible evolución:

- **Actualización de Contenido:** La aplicación permite que los administradores mantengan actualizados los productos y sus descripciones, asegurando que las recomendaciones sigan siendo relevantes.
- **Futuras Mejoras:** En un escenario real, las tareas de mantenimiento incluirían la monitorización del rendimiento, la actualización de versiones del framework y dependencias, y la optimización de la base de datos para asegurar su buen funcionamiento y seguridad a largo plazo.

9 Conclusiones y Posibles Mejoras Futuras

Conclusiones

Bueno, después de todo el trabajo, me alegra decir que el proyecto "HairLife" ha tomado forma como una plataforma web funcional. Realmente hemos conseguido abordar esa necesidad que planteamos al principio: ofrecer asesoramiento capilar personalizado.

Cumplimiento de Objetivos

Estoy bastante satisfecha porque, en general, he logrado cumplir los principales objetivos técnicos y funcionales que me propuse:

- **Sistema de Cuestionario Interactivo y Recomendación Personalizada:**
Conseguí implementar un cuestionario dinámico "ver_para_nick.blade.php" que recoge la información del usuario. El CuestionarioController procesa las respuestas y, usando mi método nuevaEstrategiaSeleccionProducto, genera una recomendación de un producto específico de la base de datos, siempre teniendo en cuenta la categoría y el tipo de cabello. Además, me aseguré de que se pueda pre-rellenar el cuestionario desde la página de recomendación. Esto cumplió con el requisito R2.4 .
- **Interfaz de Usuario (UI) Atractiva y Responsiva:** Desarrollé las interfaces con HTML, CSS ,usando variables para mantener la paleta de colores, y JavaScript, apoyándome en Bootstrap para que fuera totalmente responsiva. Páginas como crearNick.blade.php, login.blade.php, show.blade.php, ver_para_nick.blade.php, ver_producto.blade.php y gracias.blade.php son el resultado. Creo que conseguimos una navegación clara y un buen *feedback* visual, como el *slider* de preguntas y los mensajes de estado.
- **Sistema de Gestión de Usuarios Basado en Nicknames:** Implementé un sistema donde los usuarios operan con un "nick" "LoginController@guardarNick, modelo Usuario". Esto cumplió con el requisito R1.1. Puse las bases para un rol de Administrador (Rol=1 en la tabla de usuarios), aunque su panel de gestión en el *frontend* no era el foco principal ahora mismo.
- **Lógica de Negocio y Arquitectura Backend:** Me decanté por PHP con Laravel para construir los controladores (LoginController, CuestionarioController,

HomeController, RecomendacionController) y los modelos Eloquent. Esto me ayudó a gestionar el flujo de datos con la base de datos MySQL "mi_pelo.sql". Las rutas "web.php" conectan muy bien el *frontend* con el *backend*.

Lecciones Aprendidas

Cada proyecto te deja algo, y "HairLife" no fue la excepción. Aquí están algunas de las cosas más importantes que aprendí:

- **Desarrollo Backend con Laravel:** La verdad es que elegir Laravel fue un acierto. Su robustez, el sistema de rutas, el ORM Eloquent, que simplifica muchísimo la interacción con la base de datos, y la estructura MVC organizan el código de forma súper eficiente. Implementar la lógica de negocio en los controladores, sobre todo la estrategia de recomendación, fue un ejercicio práctico muy valioso para mí.
- **Diseño de Base de Datos:** Entendí que definir un esquema relacional coherente "mi_pelo.sql" es fundamental. La correcta definición de tablas y relaciones me facilitó enormemente la implementación de las funcionalidades y la obtención de datos relacionados en los modelos.
- **Desarrollo Frontend Interactivo:** Crear el *slider* de preguntas en "ver_para_nick.blade.php" con JavaScript me confirmó la importancia de la interactividad para mejorar la experiencia del usuario, sobre todo en formularios largos. Durante el proyecto, una de las cosas que más me marcó fue lo importante que es la interactividad para mejorar la experiencia del usuario, especialmente en formularios largos. Crear ese slider de preguntas en "ver_para_nick.blade.php" con JavaScript me lo confirmó: hace que el proceso sea mucho más llevadero y dinámico para quien lo está usando.

Y hablando de agilidad, tengo que destacar a Bootstrap. La verdad es que fue un descubrimiento para mí en este proyecto. Aprendí a usarlo para el diseño responsivo, y tengo que decir que agilizó muchísimo el trabajo. Antes, adaptar la interfaz a diferentes tamaños de pantalla era un quebradero de cabeza, pero con Bootstrap y sus clases predefinidas, pude crear interfaces que se ven bien en cualquier dispositivo, de una forma increíblemente rápida y eficiente. Me ayudó a centrarme más en la lógica del proyecto y menos en los detalles de CSS.

- **Algoritmos de Recomendación:**

El algoritmo en nuevaEstrategiaSeleccionProducto fue un desafío importante. Aunque no usa inteligencia artificial, me sirvió para entender lo difícil que es crear sistemas que deciden basándose en muchos datos del usuario y de los productos. Me ayudó a ver justo lo que el sistema necesitaba.

Con lo que aprendí en mis prácticas, donde ya "guiaba" la lógica con ciertas estructuras, decidí hacer algo parecido: **conectar la recomendación directamente con la descripción del producto usando palabras clave**. Así pude controlar el resultado de forma precisa, asegurando que la sugerencia siempre fuera coherente con las respuestas del usuario. Para este proyecto, fue una solución muy práctica y efectiva.

- **Gestión de Proyecto Individual:**

Gestionar el proyecto por mi cuenta, planificar las tareas y tomar mis propias decisiones fueron aspectos clave que me enriquecieron mucho durante el desarrollo

Limitaciones del Producto Actual

Si bien estoy contenta con lo logrado, también soy consciente de las limitaciones que tiene la versión actual de "HairLife":

- **Alcance del Algoritmo de Recomendación:** El sistema actual, aunque funcional, se basa en una lógica de filtrado y coincidencia de palabras clave. No usa técnicas avanzadas de Inteligencia Artificial, lo que limita su capacidad para aprender de las interacciones o descubrir patrones más complejos.
- **Funcionalidades Incompletas:** Las secciones "Mi Pelo" y "Peinados y Cortes" están marcadas como "PRÓXIMAMENTE" en el panel de usuario (show.blade.php) y no las he desarrollado funcionalmente todavía por la escasez de tiempo disponible.
- **Panel de Administración Limitado:** Aunque la estructura de la base de datos y los roles ya contemplan un administrador, no he desarrollado la interfaz de usuario "frontend" para que este pueda gestionar productos o cuestionarios de forma visual, por la misma razón, disponer de tiempo limitado.
- **Escalabilidad y Optimización:** Si el proyecto creciera mucho, con un gran volumen de usuarios y productos, seguramente necesitaría optimizaciones adicionales en las consultas a la base de datos y en la lógica de recomendación.

Decisiones Tomadas para Cumplir los Objetivos

Cada elección se hizo con un propósito claro, aquí te las explico:

Elección de Tecnologías

Me decidí por PHP con Laravel y MySQL porque ya las conocía bien. Sabía que tienen muchísima documentación, una comunidad enorme detrás para apoyarte, y son súper eficientes para desarrollar aplicaciones web con las características que necesitaba. Para el frontend, elegí Bootstrap por que me facilitó muchísimo la creación de interfaces responsivas, y lo mejor es que es un framework muy sencillo de aplicar sin necesidad de descargas de paquetes adicionales. .

- **Sistema de "Nick":** Decidí implementar un sistema de identificación de usuario basado en "nick" para simplificar el proceso de registro y personalización. No quería complicar esta etapa inicial con un sistema de cuentas de email completo para el usuario final.
- **Algoritmo de Recomendación Pragmático:** Con el tiempo y los recursos limitados que tenía, diseñé un algoritmo de recomendación basado en reglas y coincidencias de texto. Es algo que puedo implementar y cumple con el objetivo básico de ofrecer una sugerencia dirigida.
- **Priorización de Funcionalidades:** Mi prioridad fue asegurar el flujo principal de cuestionario -> recomendación, para garantizar el núcleo principal del proyecto. Eso significó dejar otras funcionalidades como "Próximamente".

Mejoras Futuras

Si tuviera más recursos disponibles : más tiempo, más gente en el equipo, más presupuesto, "HairLife" podría crecer y evolucionar muchísimo. Aquí tengo algunas ideas:

Ampliación y Desarrollo Completo de Funcionalidades:

- **Panel de Administración Integral:** Me encantaría desarrollar una interfaz completa para que los administradores puedan gestionar productos, incluyendo su creación, edición, eliminación y subida de imágenes. También, que puedan manejar las preguntas del cuestionario, crearlas, editarlas y ordenarlas, además de poder ver estadísticas de uso.
- **Implementación de "Mi Pelo":** Sería genial permitir a los usuarios llevar un diario capilar, registrar los productos que usan, subir fotos de su progreso y ver un historial de sus recomendaciones.
- **Desarrollo de "Peinados y Cortes":** Crear una galería de inspiración con filtros por tipo de cabello, forma del rostro, etc., y quizás incluso añadir tutoriales o enlaces a recursos.

- **Sistema de Valoraciones y Comentarios:** Permitir que los usuarios valoren y comenten los productos recomendados. Esto, además, podría retroalimentar el sistema de recomendación.
- **Integración de un sistema de Inteligencia Artificial más avanzado para la gestión de las recomendaciones.** Esto nos permitiría ir más allá de las reglas fijas, aprendiendo de los datos de los usuarios y las valoraciones de los productos para ofrecer sugerencias mucho más precisas y personalizadas.

Evolución del Sistema de Recomendación:

- **Algoritmos Avanzados:**
Investigar e implementar técnicas como : El filtrado colaborativo, si recogemos valoraciones, o modelos más avanzados, para que las recomendaciones sean todavía más precisas y personales.
- **Múltiples Recomendaciones:** No solo dar una opción, sino ofrecer varias alternativas de productos, quizás categorizadas ,por ejemplo, contar con opción premium, opción económica, opción natural.
- **Consideración de Más Variables:** Ampliar el cuestionario para incluir más factores como la sensibilidad del cuero cabelludo, el clima, o los objetivos específicos del usuario.

Mejoras Técnicas y de Arquitectura:

- **Pruebas Exhaustivas:** Implementar un conjunto completo de pruebas unitarias y pruebas de integración/funcionales automatizadas, para garantizar la estabilidad y fiabilidad a largo plazo.
- **API para Extensibilidad:** Desarrollar una API REST para permitir futuras integraciones con aplicaciones móviles, asistentes virtuales u otras plataformas.
- **Optimización de Rendimiento:** Para grandes volúmenes de datos, me gustaría optimizar las consultas a la base de datos, implementar sistemas de caché más avanzados y considerar la paginación eficiente.
- **Gestión de Assets Profesional:**

Utilizar **Laravel Vite o Mix** para organizar y optimizar nuestros *assets* (que son los archivos CSS y JavaScript de diseño y funcionalidad), en lugar de depender solo de CDNs en producción.

Mejoras en la Experiencia de Usuario (UX/UI):

- **Diseño Visual Refinado:** Contratar o colaborar con un diseñador UX/UI para pulir la estética general, mejorar la coherencia visual y la identidad de marca.
- **Pruebas de Usabilidad con Usuarios Reales:** Realizar sesiones de prueba con usuarios finales para identificar puntos de fricción y áreas de mejora en la interfaz y los flujos de interacción.
- **Internacionalización y Localización :** Adaptar la plataforma para múltiples idiomas y regiones si se planea una expansión.
- **Notificaciones:** Implementar un sistema de notificaciones para los usuarios ,por ejemplo, para nuevos productos que coincidan con su perfil, o recordatorios.

Integraciones Externas:

- **Proveedores de Productos:** Integrarme con APIs de tiendas online para obtener información de productos actualizada , o incluso para facilitar la compra directa.
- **Redes Sociales:** Permitir el login con redes sociales o que los usuarios puedan compartir sus recomendaciones fácilmente.

10 Bibliografía y referencias

Para la realización del proyecto "HairLife", se consultaron diversas fuentes de información y documentación técnica, fundamentales para el aprendizaje y la aplicación de las tecnologías involucradas. A continuación, se presenta una relación de los principales recursos que un desarrollador en formación podría haber utilizado:

Documentación Oficial de Tecnologías:

- **PHP.** (s.f.). *PHP Manual*. Obtenido de <https://www.php.net/manual/es/>
 - *Descripción:* Fuente primaria para consultas sobre la sintaxis, funciones y características del lenguaje PHP, esencial para el desarrollo backend.
- **Laravel.** (s.f.). *Laravel Documentation*. Obtenido de <https://laravel.com/docs/10.x>
 - *Descripción:* Guía oficial completa del framework Laravel, consultada para comprender su arquitectura MVC, Eloquent ORM, sistema de rutas, Blade templating, manejo de formularios, seguridad y otros componentes.
- **MySQL.** (s.f.). *MySQL Reference Manual*. Obtenido de <https://dev.mysql.com/doc/>
 - *Descripción:* Documentación oficial de MySQL, utilizada para el diseño del esquema de la base de datos (mi_pelo.sql), tipos de datos, sentencias SQL y mejores prácticas en la gestión de bases de datos relacionales.
- **Bootstrap.** (s.f.). *Bootstrap Documentation*. Obtenido de <https://getbootstrap.com/docs/5.3/getting-started/introduction/>
 - *Descripción:* Documentación del framework Bootstrap, crucial para la implementación del diseño responsivo, el sistema de rejilla (grid system), y el uso de componentes predefinidos (formularios, botones, tarjetas, alertas, etc.) en las vistas Blade.
- **MDN Web Docs (Mozilla Developer Network).** (s.f.). *HTML: HyperText Markup Language*. Obtenido de <https://developer.mozilla.org/es/docs/Web/HTML>
 - *Descripción:* Recurso de referencia para la estructura semántica de las páginas web.
- **MDN Web Docs (Mozilla Developer Network).** (s.f.). *CSS: Cascading Style Sheets*. Obtenido de <https://developer.mozilla.org/es/docs/Web/CSS>
 - *Descripción:* Guía exhaustiva para el estilizado de las páginas, incluyendo el uso de Flexbox, Grid, variables CSS (custom properties) y selectores avanzados utilizados en las vistas del proyecto.

- **MDN Web Docs (Mozilla Developer Network).** (s.f.). *JavaScript*. Obtenido de <https://developer.mozilla.org/es/docs/Web/JavaScript>
 - *Descripción:* Referencia principal para la implementación de la interactividad del lado del cliente, como la manipulación del DOM para el slider de preguntas en `ver_para_nick.blade.php` y la redirección en `gracias.blade.php`.
- **Git SCM.** (s.f.). *Git Documentation*. Obtenido de <https://git-scm.com/doc>
 - *Descripción:* Documentación oficial del sistema de control de versiones Git, utilizado para la gestión del código fuente del proyecto.

Comunidades y Foros de Resolución de Problemas:

- **Stack Overflow.** (s.f.). Obtenido de <https://stackoverflow.com>
 - *Descripción:* Comunidad de preguntas y respuestas ampliamente utilizada por desarrolladores para solucionar problemas específicos de programación, errores de código y dudas conceptuales sobre Laravel, PHP, JavaScript, CSS, y MySQL.
- **Foros de Laravel (ej. Laracasts Forums, Laravel.io).** (s.f.).
 - *Descripción:* Comunidades específicas de Laravel donde se discuten problemas, se comparten soluciones y se puede obtener ayuda sobre aspectos concretos del framework.

Tutoriales y Plataformas de Aprendizaje:

- **Laracasts.** (s.f.). Obtenido de <https://laracasts.com>
 - *Descripción:* Plataforma de screencasts y tutoriales de alta calidad enfocada principalmente en Laravel y tecnologías web modernas. Una fuente probable para aprender las bases y técnicas avanzadas de Laravel.
- **W3Schools.** (s.f.). Obtenido de <https://www.w3schools.com>
 - *Descripción:* Sitio popular con tutoriales y referencias para HTML, CSS, JavaScript, PHP y SQL, útil para consultas rápidas y ejemplos prácticos.
- **CSS-Tricks.** (s.f.). Obtenido de <https://css-tricks.com>
 - *Descripción:* Blog y recurso con artículos detallados, trucos y técnicas sobre CSS, incluyendo Flexbox, Grid y diseño responsivo, que habrían sido útiles para los estilos personalizados del proyecto.
- **Diversos blogs y tutoriales en línea sobre desarrollo web.** (s.f.).
 - *Descripción:* A lo largo del desarrollo, es común consultar múltiples artículos de blogs de desarrolladores, tutoriales en YouTube y otros recursos en línea para resolver dudas puntuales o aprender técnicas específicas (ej.

implementación de efectos CSS como el "flip card", manejo de formularios en Laravel, etc.).

Herramientas de Desarrollo (Documentación y Uso):

- **Visual Studio Code.** (s.f.). *Visual Studio Code Documentation*. Obtenido de <https://code.visualstudio.com/docs>
 - *Descripción:* Documentación del editor de código utilizado, consultada para configuración, uso de extensiones (ej. para PHP, Blade, GitLens) y optimización del entorno de desarrollo.
- **XAMPP (o similar como WAMP, MAMP, Laragon).** (s.f.). *Apache Friends - XAMPP*. Obtenido de <https://www.apachefriends.org>
 - *Descripción:* Documentación o guías de la pila de desarrollo local utilizada para configurar el servidor Apache, PHP y MySQL en el entorno de desarrollo.

Herramientas Gratuitas de IA para Generación y Edición de Imágenes:

En el proceso de desarrollo, o para futuras mejoras donde se necesiten recursos visuales, también es muy útil conocer plataformas gratuitas que aprovechan la Inteligencia Artificial para generar y editar imágenes. Aquí te dejo algunas opciones populares:

- **ClipDrop (varias herramientas de IA para edición):** Ofrece varias herramientas gratuitas con IA para editar imágenes, como eliminar objetos, cambiar fondos, o reiluminar.
 - **Acceso:** <https://clipdrop.co/> (algunas funciones tienen limitaciones en la versión gratuita).
- **Bing Image Creator (basado en DALL-E 3):** Una herramienta muy potente de Microsoft que permite generar imágenes detalladas a partir de descripciones de texto, utilizando la tecnología de OpenAI.
 - **Acceso:** <https://www.bing.com/images/create> (requiere cuenta Microsoft)

11 Anexos (Glosario de términos, versiones o bocetos de la aplicación, soporte multimedia utilizado -videos, etc.-).

11. Anexos

Esta sección recopila materiales complementarios que, por su naturaleza o extensión, se presentan de forma separada del cuerpo principal del documento para no interrumpir su fluidez.

11.1. Índice de Ilustraciones

Índice de figuras

Figura 1: "Figura A.1: Diagrama Relacional de la Base de Datos 'HairLife' (referencia a sección 7.4 Diseño de la Aplicación).....	66
Figura 2: "Figura A.2: Diagrama de Clases para los Modelos de la Aplicación (referencia a sección 7.4 Diseño de la Aplicación).....	67
Figura 3: "Figura A.4: Prototipo de la Página de Creación de Nick (referencia a sección 7.4 Diseño de la Aplicación)."	68
Figura 4: Figura A.3: Prototipo de la Interfaz de Login de usuario (referencia a sección 7.4 Diseño de la Aplicación).....	68
Figura 5: "Figura A.5: Prototipo de la Página de Panel de Control(referencia a sección 7.4 Diseño de la Aplicación)."	68
Figura 6: Figura A.6: Prototipo de la Página de Cuestionario (referencia a sección 7.4 Diseño de la Aplicación).....	68
Figura 7: igura A.8: Prototipo de recomendacion de producto (referencia a sección 7.4 Diseño de la Aplicación).....	69
Figura 8: igura A.7: Prototipo de la Página de loading (referencia a sección 7.4 Diseño de la Aplicación).....	69
Figura 9: Figura 9: Figura B.1: Estructura de Directorios Principal del Proyecto Laravel (referencia a sección 7.5 Implementación y Documentación).....	70
Figura 10: Figura B.1: Estructura de Directorios Principal del Proyecto Laravel (referencia a sección 7.5 Implementación y Documentación).....	70

Figura 11: Figura 10: Figura B.1: Estructura de Directorios Principal del Proyecto Laravel (referencia a sección 7.5 Implementación y Documentación).....	71
Figura 12: Figura 10: Figura B.1: Estructura de Directorios Principal del Proyecto Laravel (referencia a sección 7.5 Implementación y Documentación).....	71
Figura 13: Login.....	73
Figura 14: Creación de nick.....	73
Figura 15: Panel de Usuario.....	74
Figura 16: Cuestionario.....	74
Figura 17: Loading.....	74
Figura 18: Recomendación de producto.....	75

11.2. Glosario de Términos

• 11.2. Glosario de Términos

El presente glosario define los términos técnicos, abreviaturas y conceptos específicos utilizados a lo largo del proyecto "HairLife", con el fin de asegurar la claridad y comprensibilidad del documento para un público diverso.

- **Backend:** Se refiere a la parte del sistema que se ejecuta en el servidor y gestiona la lógica de negocio, la base de datos y la autenticación. En este proyecto, se implementó con PHP y Laravel.
- **CDN (Content Delivery Network):** Red de servidores distribuidos geográficamente que trabajan juntos para proporcionar una entrega rápida de contenido web.
- **CRUD:** Acrónimo de Create, Read, Update, Delete. Representa las cuatro funciones básicas de persistencia de datos.
- **CSS (Cascading Style Sheets):** Lenguaje de hojas de estilo utilizado para describir la presentación de un documento escrito en HTML.
- **DOM (Document Object Model):** Interfaz de programación para documentos HTML y XML. Representa la página para que los programas puedan cambiar la estructura, estilo y contenido del documento.
- **Eloquent ORM:** Implementación de Patrón de Diseño de Registro Activo (Active Record) en Laravel para interactuar con la base de datos de manera orientada a objetos.
- **Frontend:** Se refiere a la parte de la aplicación con la que el usuario interactúa directamente. En este proyecto, se desarrolló con HTML, CSS y JavaScript.
- **HTML (HyperText Markup Language):** Lenguaje estándar para la creación de páginas web y aplicaciones.
- **IA (Inteligencia Artificial):** Campo de la informática que se enfoca en la creación de máquinas que pueden razonar, aprender y actuar de manera inteligente.

- **JavaScript:** Lenguaje de programación interpretado, multiparadigma y de alto nivel. Se utiliza comúnmente para dotar de interactividad a las páginas web.
- **Laravel:** Framework de aplicación web PHP de código abierto, diseñado para el desarrollo de aplicaciones web siguiendo el patrón de arquitectura Modelo-Vista-Controlador (MVC).
- **Machine Learning:** Subcampo de la inteligencia artificial que se enfoca en el desarrollo de algoritmos que permiten a las computadoras aprender de los datos sin ser programadas explícitamente.
- **MVC (Modelo-Vista-Controlador):** Patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.
- **MySQL:** Sistema de gestión de bases de datos relacionales de código abierto, utilizado para almacenar los datos de la aplicación "HairLife".
- **PHP (Hypertext Preprocessor):** Lenguaje de programación de propósito general, de código abierto, muy popular para el desarrollo web.
- **SQL (Structured Query Language):** Lenguaje de dominio específico utilizado en programación y diseñado para administrar y recuperar información de sistemas de gestión de bases de datos relacionales.
- **UI (User Interface):** Interfaz de usuario, se refiere a todo lo que un usuario puede ver y con lo que puede interactuar en una aplicación o sitio web.
- **UX (User Experience):** Experiencia de usuario, abarca todos los aspectos de la interacción del usuario final con la empresa, sus servicios y sus productos.

11.3. Ilustraciones y Diagramas del Proyecto

11.3.1. Diagramas de Diseño

- **Contenido:** Tu **Diagrama de Base de Datos** (E-R o Relacional) y cualquier **Diagrama UML** (Clases, Secuencia, Casos de Uso) que hayas generado.

Diagrama BBDD:

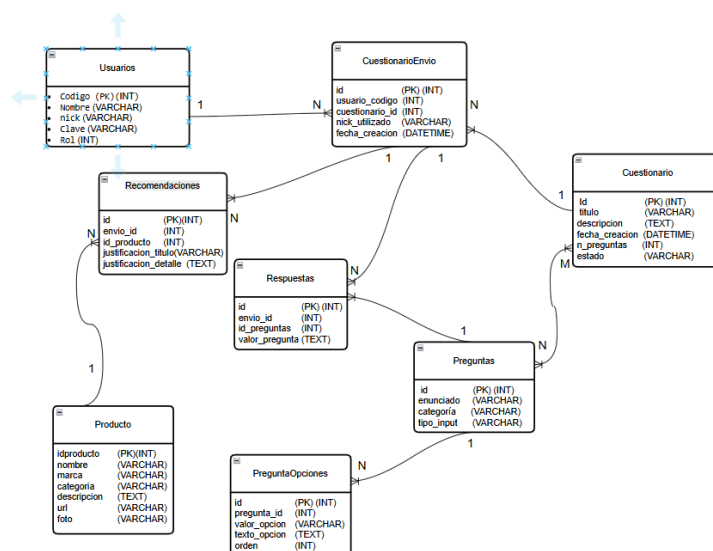
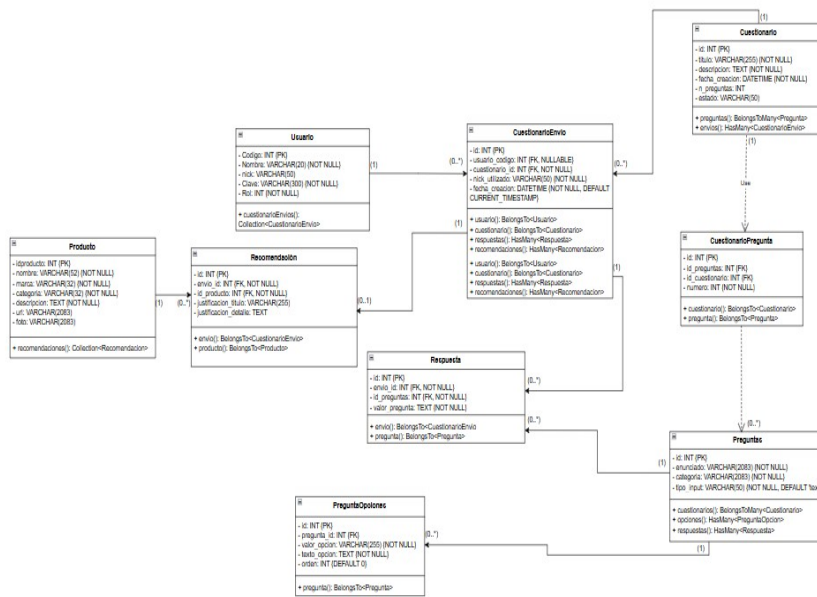


Figura 1: "Figura A.1: Diagrama Relacional de la Base de Datos 'HairLife' (referencia a sección 7.4 Diseño de la Aplicación)

Diagrama de UML:

Figura 2: "Figura A.2: Diagrama de Clases para los Modelos de la Aplicación (referencia a sección 7.4 Diseño de la Aplicación).



11.3.2. Prototipos y Flujo de Interfaz

- **Contenido:** Los bocetos o *wireframes* iniciales de las páginas de tu aplicación.

Diseño de la interfaz de usuario con figma:



Figura 4: Figura A.3: Prototipo de la Interfaz de Login de usuario (referencia a sección 7.4 Diseño de la Aplicación).



Figura 3: "Figura A.4: Prototipo de la Página de Creación de Nick (referencia a sección 7.4 Diseño de la Aplicación)."



Figura 5: "Figura A.5: Prototipo de la Página de Panel de Control(referencia a sección 7.4 Diseño de la Aplicación)."



Figura 6: Figura A.6: Prototipo de la Página de Cuestionario (referencia a sección 7.4 Diseño de la Aplicación)



Figura 7: igura A.8: Prototipo de recomendacion de producto (referencia a sección 7.4 Diseño de la Aplicación).



Figura 8: igura A.7: Prototipo de la Página de loading (referencia a sección 7.4 Diseño de la Aplicación).

11.3.2. Herramientas y Gestión del Proceso de Desarrollo

- **Contenido:** Capturas que ilustran cómo se organizó, gestionó y se trabajó en el código del proyecto.

Estructura de Directorios Principal del Proyecto Laravel:

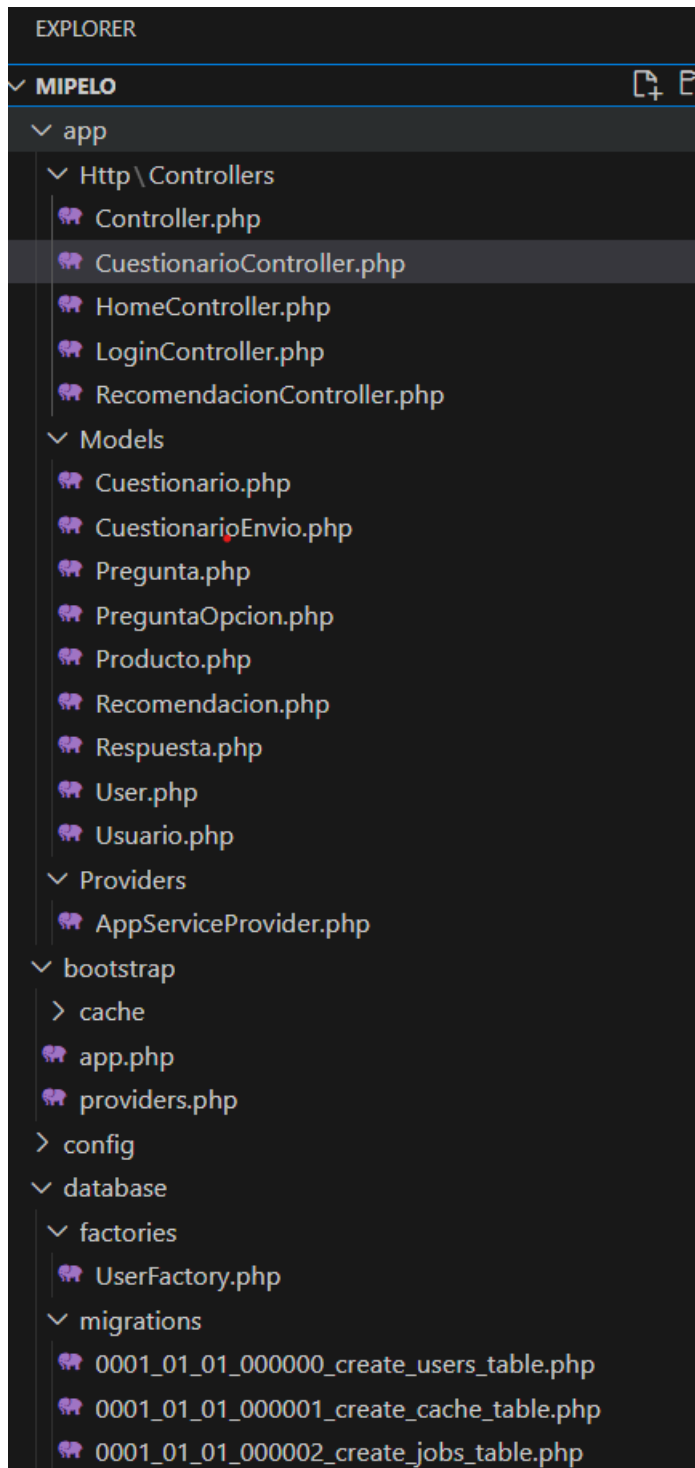
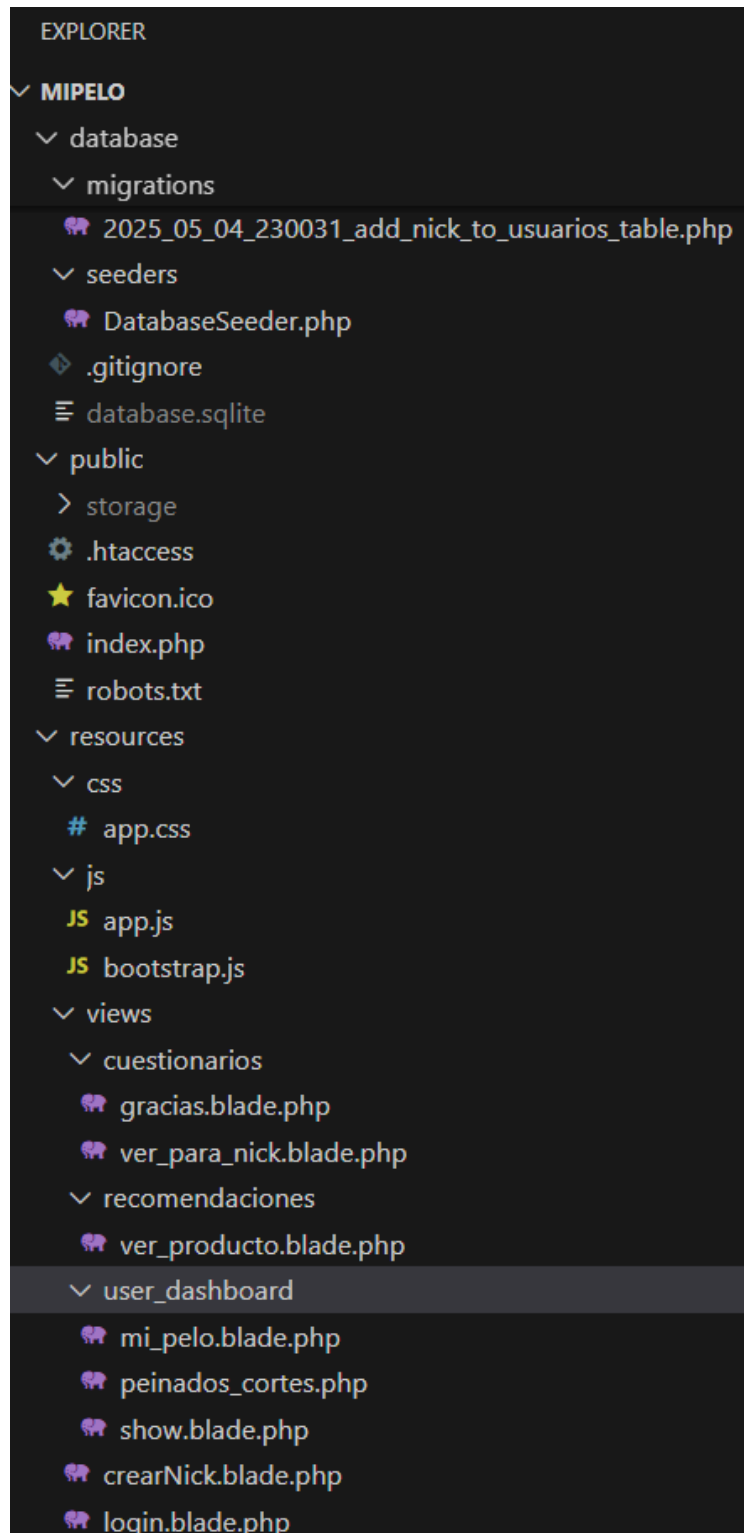


Figura 10: Figura B.1: Estructura de Directorios Principal del Proyecto Laravel (referencia a sección 7.5 Implementación y Documentación)



75 Figura 9: Figura 9: Figura B.1: Estructura de Directorios Principal del Proyecto Laravel (referencia a sección 7.5 Implementación y Documentación)

Estructura de Directorios Principal del Proyecto Laravel:

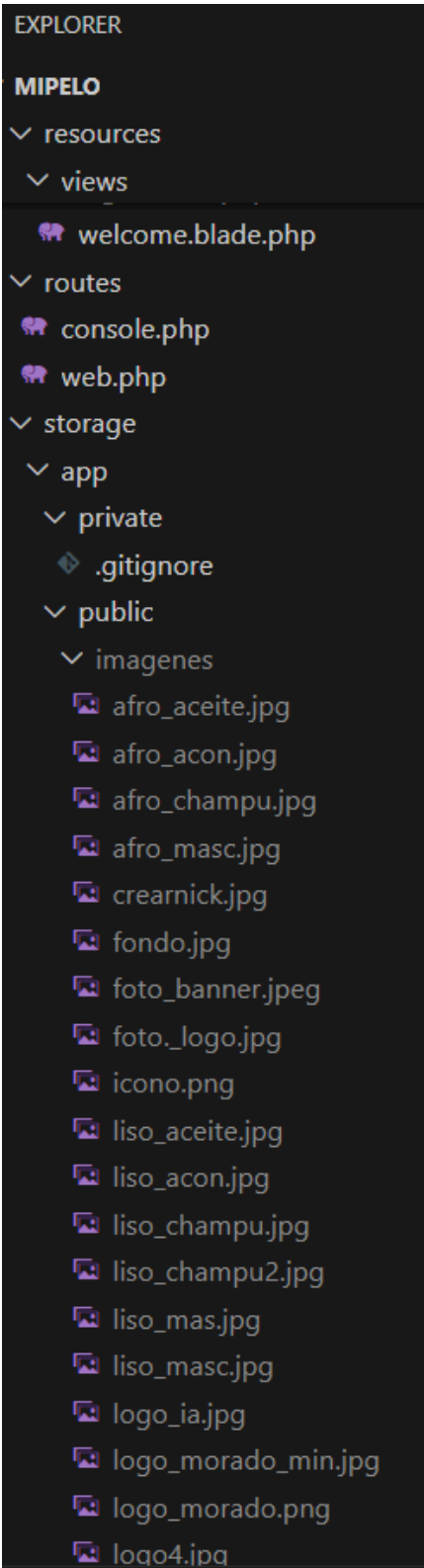


Figura 12: Estructura de Directorios Principal del Proyecto Laravel (referencia a sección 7.5 Implementación y Documentación).

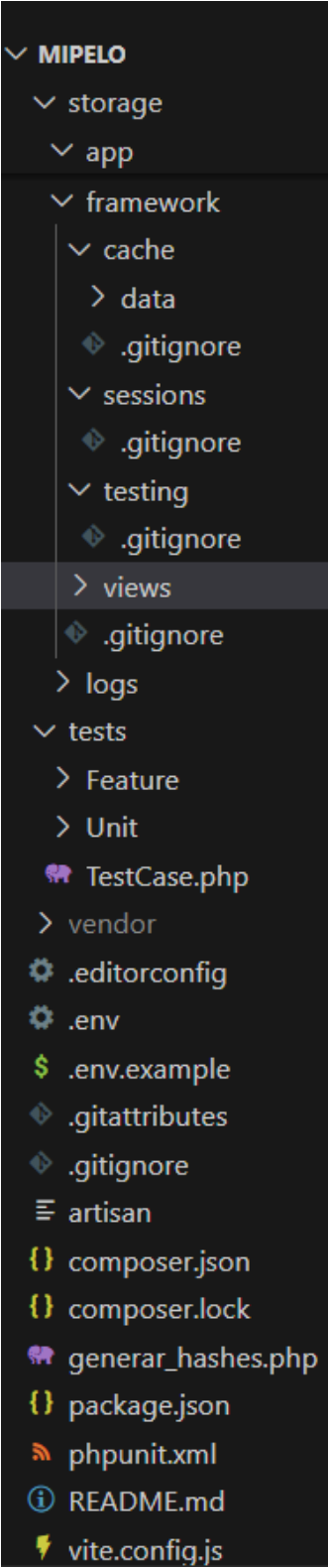




Figura 10: Estructura de Directorios Principal del Proyecto Laravel (referencia a sección 7.5 Implementación y Documentación).

Historial de Comits y Gestión de Ramas en Git

- mejoras en la función de la nueva estrategia de sel...  main 
- nueva inserción de la funcionalidad en las recomendaciones aissa
- cambio en la el controlador de cuestionari y en el gracias.blade...
- Comentarios y limpiar codigo aissa
- Incorporación de footers aissa
- imagenes traseras de las tarjetas aissa
- Hasheo de contraseña aissa
- diseño css en show, blade, crear nick,login aissa
- diseño css en crearnik, login aissa
- modificacion del nick aissa
- solucion de problema con las migraciones aissa
- ajustes en los productos recomendados aissa
- modificación estetica del botón aissa
- gesstion del boton de la última vista para volver a la última pre...
- modificaciones varias aissa
- modificacion y diseño de blade crearnick aissa
- Corrección del nick en el login aissa
- Actualización del frontend del blade gracias aissa
- Actualización del frontend del cuestionario a dinamico aissa
- Actualización del frontend del cuestionario a dinamico aissa
- Actualización del frontend del panel aissa
- Actualización del frontend del panel aissa
- Actualización del frontend del login aissa
- Actualización del frontend del login aissa
- integracion de volver al menu aissa
- insercion de recomendaciones aissa
- Implementar redirección a página de gracias con método y ruta...
- feat: ✨ introduccion .env aissa
- feat: commit de prueba aissa
- feat: ✨ introduccion del .env aissa

11.3.3. Capturas de Pantalla de la Aplicación (GUI en Funcionamiento)

- **Contenido:** Imágenes reales de tu aplicación en acción, mostrando su interfaz gráfica.

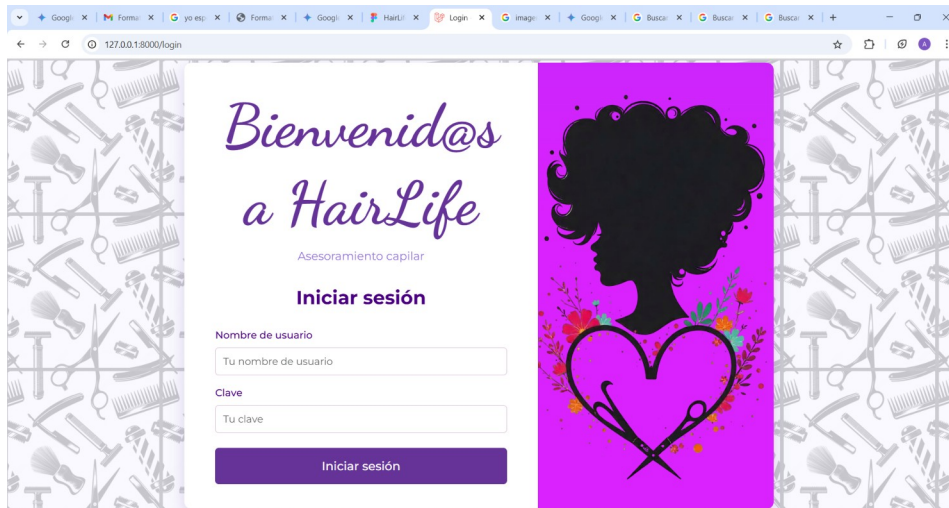


Figura 13: Login

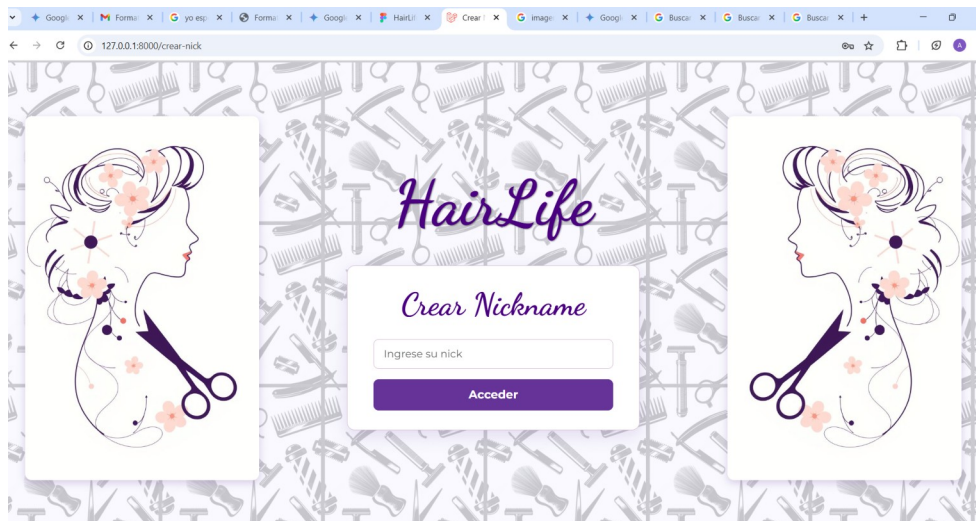


Figura 14: Creación de nick

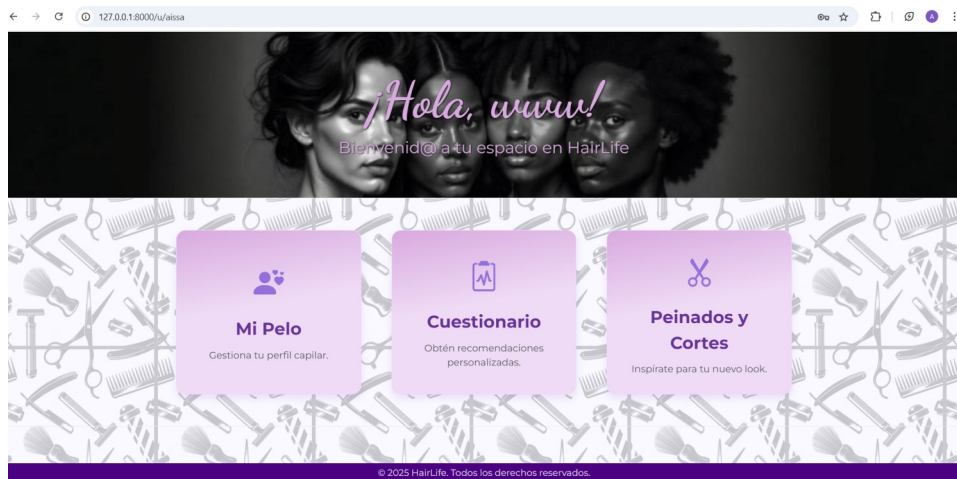


Figura 15: Panel de Usuario

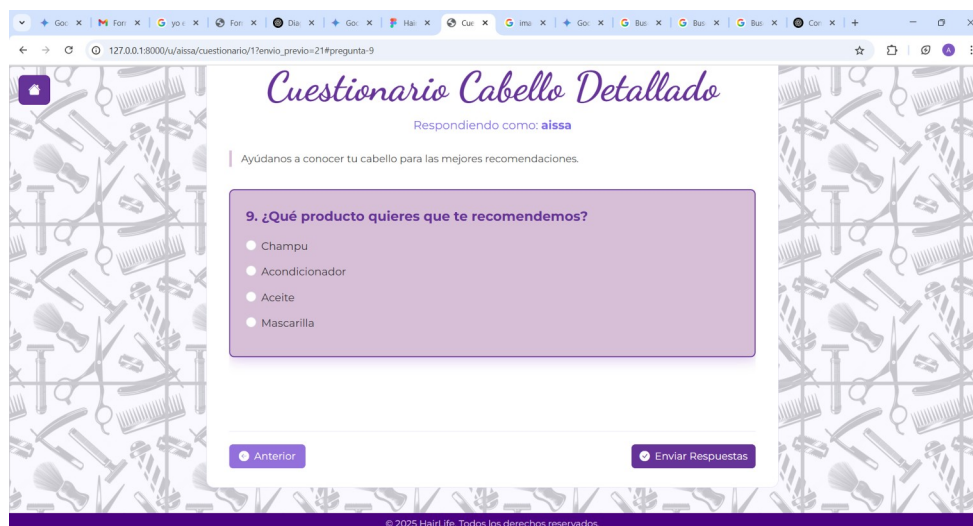


Figura 16: Cuestionario

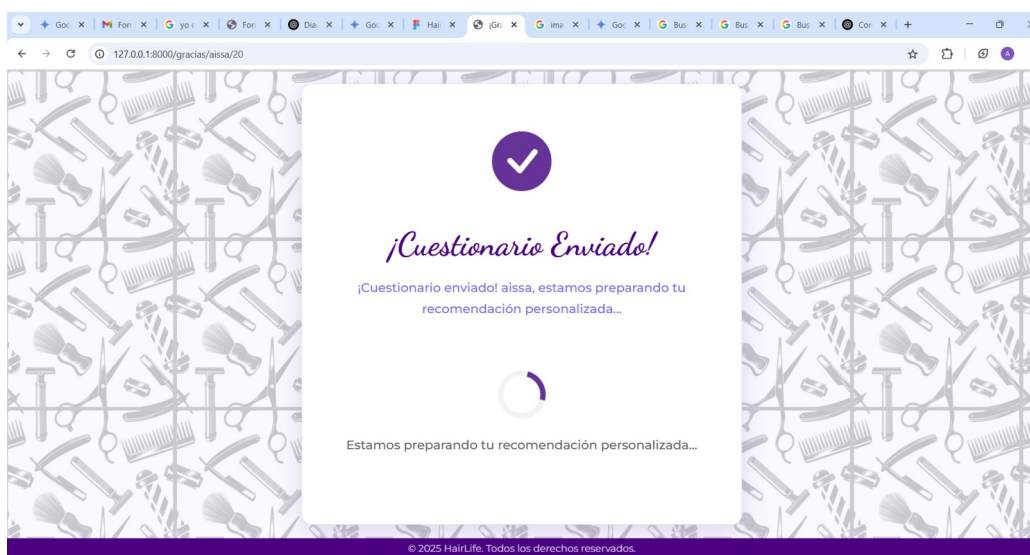


Figura 17: Loading

[Volver al cuestionario](#)

Hair Food Smoothing Macadamia

Marca: Garnier | Categoría: Mascarilla



Descripción y Modo de Uso

Mascarilla nutritiva para cabello liso (normal a grueso) o con frizz. Suaviza y controla. Indicada para cuero cabelludo seco y cabello con alta porosidad por su nutrición. Modo de uso: Como mascarilla, aplicar sobre cabello húmedo, dejar actuar 3-5 minutos y aclarar. También como acondicionador o leave-in (poca cantidad). Frecuencia: 1-2 veces por semana como mascarilla.

[Ver o Comprar Producto](#)

Figura 18: Recomendación de producto

Fin de la memoria



Autor: Aissa Diallo González

Fecha Fin: 13/06/2025