FACULTAD DE INGENIERÍA

PLANNI
Engineering Project Final Report

Jaider Castilla Babilonia, Esneider Enrique Guzman Perez, Kevin Esteban Martínez Palmieri, Jesús Antonio Martínez Velandia, Diego Andrés Quintana Fajardo.

Project Advisors:

Dr. Edwin Alexander Puertas Del Castillo.

Dr. Juan Carlos Martinez Santos.

Course Advisor:
Rafael Enrique Monterroza Barrios

Program:
Systems Engineering and Computing

Mayo 29, 2024

Table of Contents

List of Illustrations

List of Figures

Acknowledgments

We would like to express our sincere thanks to the professors who supported us in this project.

Thanks to Dr. Edwin Puertas for his invaluable help in everything related to artificial intelligence. His guidance and knowledge were fundamental to the success of our project.

We also thank Dr. Juan Martinez for his support in optimization. His advice was key to improve the efficiency and effectiveness of our work.

To both of them, we extend our deepest gratitude for their time, commitment and for sharing their knowledge with us. This project would not have been possible without their valuable contribution.

Abstract

Problem

The Planni project arises to address the inherent complexity of personalized travel planning. Users are faced with an overwhelming number of options and variables (destinations, accommodations, transportation, activities, etc.) that they must adjust according to their preferences and budget. This process can be tedious, error-prone and does not always optimize the user experience.

Proposed Solution

Planni proposes a platform that automates the creation of customized tourism packages, integrating artificial intelligence (AI) that analyzes user preferences and generates personalized recommendations. This system includes key functionalities such as user registration and authentication, preference capture, package recommendation, and online booking.

Requirements Summary

1. **User Functionality:**
    a. Registration and login.
    b. Capture of travel preferences.
    c. Personalized recommendations.
    d. Reservation and payment of tour packages.
    e. Reservation history management.
2. **Technical Features:**
    a. Monolithic backend with endpoints for authentication, AI, and search.
    b. Web interfaces using ReactJS and Next.js.
    c. Integration with external APIs (TripAdvisor, Google).
    d. Security and scalability.

Engineering Standards Considered

1. **Security Standards:** Compliance with personal data protection regulations such as GDPR and local privacy laws.

2. **Software Design Standards:** Use of design patterns (such as MVC), SOLID principles and best practices in web development.
3. **Interoperability Standards**: Standards to integrate external APIs in a secure and efficient way.
4. **Accessibility Standards:** Ensure that the platform is usable by people with disabilities, according to standards such as WCAG.

Developed Products

1. Software:
    a. Web platform with intuitive user interface and optimized for mobile devices.
    b. Monolithic backend managed in Express.js with modular structures.
2. Detailed Designs:
    a. Detailed database diagram.
    b. Component and sequence diagrams.
3. Processes:
    a. AI algorithms for processing and recommendation.
    b. User authentication and authorization flows.

Applicable Courses in the Project

- Artificial Intelligence: Applied to user data processing and recommendation generation.
- Web Application Development: Fundamental for user interface and backend implementation.
- Databases: Design and management of the data structure required for the system.
- Computer Security: Implementation of data protection and platform security measures.
- Distributed Systems: Integration and management of external APIs.

New Skills and Knowledge Acquired

- API Integration: Ability to integrate and manage data from multiple external services.
- Full-Stack Development: Advanced knowledge in the development of complete applications from frontend to backend.
- AI Implementation: Practical application of NLP.
- Project Management: Improved ability to plan and execute complex software projects.

- Process Optimization: Techniques to improve processing efficiency and system response times.

Resumen

Problema

El proyecto Planni surge para abordar la complejidad inherente a la planificación de viajes personalizados. Los usuarios enfrentan una abrumadora cantidad de opciones y variables (destinos, alojamiento, transporte, actividades, etc.) que deben ajustar según sus preferencias y presupuesto. Este proceso puede ser tedioso, propenso a errores y no siempre optimiza la experiencia del usuario.

Solución Propuesta

Planni propone una plataforma que automatiza la creación de paquetes turísticos personalizados, integrando una inteligencia artificial (IA) que analiza las preferencias del usuario y genera recomendaciones personalizadas. Este sistema incluye funcionalidades clave como el registro y autenticación de usuarios, captura de preferencias, recomendación de paquetes, y reservas en línea.

Resumen de los Requisitos

1. **Funcionalidad de Usuario**
   - Registro e inicio de sesión.
   - Captura de preferencias de viaje.
   - Recomendaciones personalizadas.
   - Reserva y pago de paquetes turísticos.
   - Gestión del historial de reservas.
2. **Características Técnicas**
   - Backend monolítico con endpoints para autenticación, IA, y búsqueda.
   - Interfaces web usando ReactJS y Next.js.
   - Integración con APIs externas (TripAdvisor, Google).
   - Seguridad y escalabilidad.

Normas de Ingeniería Consideradas

1. **Normas de Seguridad**: Cumplimiento con regulaciones de protección de datos personales como GDPR y leyes locales de privacidad.
2. **Normas de Diseño de Software**: Uso de patrones de diseño (como MVC), principios SOLID y best practices en desarrollo web.

3. **Normas de Interoperabilidad**: Estándares para integrar APIs externas de forma segura y eficiente.
4. **Normas de Accesibilidad**: Asegurar que la plataforma sea usable por personas con discapacidades, conforme a estándares como WCAG.

Productos Desarrollados

1. **Software**
   - Plataforma web con interfaz de usuario intuitiva y optimizada para dispositivos móviles.
   - Backend monolítico gestionado en Express.js con estructuras modulares.
2. **Diseños Detallados**
   - Diagrama de base de datos detallado.
   - Diagramas de componentes y secuencia.
3. **Procesos**
   - Algoritmos de IA para procesamiento y recomendación.
   - Flujos de autenticación y autorización de usuarios.

Cursos Aplicables en el Proyecto

1. **Inteligencia Artificial**: Aplicados para el procesamiento de datos de usuario y generación de recomendaciones.
2. **Desarrollo de Aplicaciones Web**: Fundamental para la implementación de la interfaz de usuario y backend.
3. **Bases de Datos**: Diseño y gestión de la estructura de datos necesarios para el sistema.
4. **Seguridad Informática**: Implementación de medidas de protección de datos y seguridad en la plataforma.
5. **Sistemas Distribuidos**: Integración y manejo de APIs externas.

Nuevas Habilidades y Conocimientos Adquiridos

1. **Integración de APIs**: Habilidad para integrar y manejar datos de múltiples servicios externos.
2. **Desarrollo Full-Stack**: Conocimiento avanzado en el desarrollo de aplicaciones completas desde frontend hasta backend.
3. **Implementación de IA**: Aplicación práctica de algoritmos NLP.
4. **Gestión de Proyectos**: Mejoras en la capacidad de planificación y ejecución de proyectos de software complejos.
5. **Optimización de Procesos**: Técnicas para mejorar la eficiencia de procesamiento y tiempos de respuesta del sistema.

1 Problem Statement

1.1 Needs

Planning personalized trips is a complex and tedious task for travelers. They need to consider multiple variables such as destinations, accommodations, transportation, and activities, matching them to their preferences and budgets. This often results in a cumbersome process prone to errors and inefficiencies.

Current solutions lack effective customization, often leading to unsatisfactory travel experiences. Users face overwhelming choices and difficulty finding reliable information that suits their specific needs. The tourism industry requires an innovative approach to streamline and personalize trip planning, enhancing both convenience and satisfaction for travelers.

1.2 Objectives

To address these needs, the project aims to develop a platform that automates and optimizes the creation of personalized travel packages using artificial intelligence. By incorporating user preferences and leveraging data from external sources, this platform will provide tailored recommendations and streamline the trip planning process.

1.2.1 General Objective:

- Develop an AI-based platform that personalizes travel package recommendations based on user preferences and information provided by the user when selecting travel data.

12.2 Specific Objectives:

- To design and implement a user-friendly interface for capturing travel preferences.
- To integrate AI algorithms that analyze user data and generate personalized travel recommendations.
- To connect with external APIs (e.g., TripAdvisor, Google) for enrichment of travel package data.

1.3 Operational Environment

The operational environment describes the digital and physical spaces surrounding how users use the platform.
Digital Environment:

- **Multiplaform Accessing:** The platform will be available through web browsers, so that using it is possible from a desktop, laptop among other tablets and smartphones.
- **APIs Integration:** Planni will also be integrated with external APIs such as trip advisor, google for fetching the travel package data. The platform should be able to cope with different API response times and sometimes patchy connectivity.
- **Security Measures:** In order for users to provide their personal and payment information, comprehensive cybersecurity measures are needed to secure the data from unauthorized access and breaches.

Physical Environment

- **Mobile and desktop use:** The platform supports both cell phones and desktop computers, since being a web platform, users could use them wherever they have Internet access.
- **User Interface Adaptability:** Responsive and comfortable functional interface on equipment with different screen size and resolution to ensure an accurate level experience, whether the user uses a small smartphone or desktopmonitor.

1.4 Requirements

Functional Requirements

- **User Registration and Authentication:** Users must be able to register, log in, and manage their accounts securely.
- **Preferences Capture:** Users must be able to input their travel preferences, including destinations, dates, budget, and interests.
- **AI-Powered Recommendations:** The platform must generate personalized travel package recommendations based on user preferences and data from external APIs.

Economic/Market Requirements

- **Cost-Effectiveness:** The platform should be cost-effective to develop and maintain, leveraging open-source tools and scalable cloud services.
- **Competitive Pricing:** Travel packages offered through the platform should be competitively priced to attract and retain users.

User Interface Requirements

- **Responsiveness:** The platform must be fully responsive, ensuring usability across different devices and screen sizes.
- **Intuitive Design:** The UI should be user-friendly and intuitive, allowing users to easily navigate through the platform and access features without steep learning curves.

Security Requirements

- **Data Protection:** Ensure all user data is encrypted in transit and at rest, complying with data protection regulations.
- **Authentication and Authorization:** Implement robust authentication and authorization mechanisms to prevent unauthorized access.

Performance Requirements

- **Scalability:** The platform must be able to handle a growing number of users and transactions without compromising performance.
- **Low Latency:** Ensure fast response times for user interactions, including search queries and loading recommendations.
- **Availability:** Maintain high availability with minimal downtime, ensuring the platform is always reliable and accessible.

By addressing these requirements, Planni will deliver a robust, user-friendly, and efficient platform that meets the needs of modern travelers and travel operators, while ensuring security, sustainability, and market viability.

1.5 Intended Users and Uses

To properly design an end product that will provide maximum satisfaction and perform in the most efficient manner, it is essential to understand the end-users and the associated end uses.

Intended Users

1. **Travelers (Tourists/Clients)**

    - **Demographic**: Individuals, families, and groups interested in personalized travel planning.
    - **Technology Comfort Level**: Ranges from tech-savvy users to those with basic technology knowledge.
    - **Needs**:
        - Simplify the planning of personalized trips.
        - Obtain reliable recommendations that match their preferences and budget.
        - Easily book travel services such as accommodation, transportation, and activities.
        - Ensure data security and privacy in handling information and transactions.
2. **Administrators (System Managers)**
    - **Demographic**: Personnel responsible for maintaining and managing the Planni system.
    - **Technology Comfort Level**: Highly familiar with technology and system administration.
    - **Needs**:
        - Monitor and maintain the integrity and security of the system.
        - Manage users and content.
        - Resolve technical issues and optimize system performance.

Intended Uses

1. **Travelers**
    - **Trip Planning**: Use the platform to plan personalized trips, selecting destinations, dates, budgets, and preferences to receive tailored travel package recommendations.
2. **Administrators**
    - **User and Content Management**: Manage the user base and moderate content generated on the platform.

1.6 Assumptions and Limitations

Assumptions

1. **Device Compatibility**:
    - The platform will be primarily used on modern web browsers (Chrome, Firefox, Safari, Edge) and mobile devices (iOS, Android). This assumption ensures that the user interface is optimized for the majority of users while managing development resources effectively.

Limitations

1. **Geographic Scope**:

- Initially, the platform will focus on promoting tourism within Cartagena D,T y C. Expansion to other geographical regions will be considered in subsequent phases. This limitation allows for a focused launch and ensures resources are utilized effectively.

2. **Data Sources**:
   - The accuracy and availability of travel package recommendations depend on the reliability of external APIs (TripAdvisor, Google). Any changes or discontinuities in these services could affect the platform's functionality.
3. **Data Privacy and Compliance**:
   - The platform must comply with data protection regulations such as GDPR and local privacy laws. This limitation requires thorough data handling and security measures to protect user information.

1.7 Expected End Product and Deliverables

End Product

Planni: AI-Powered Travel Package Platform

A comprehensive web-based platform that allows users to plan personalized travel packages tailored to their preferences. The system integrates AI-driven recommendations and connects with external travel data sources to provide a seamless and personalized travel planning experience.

Key Features Include:

1. **User Registration and Authentication**
   - Secure user sign-up, login, and account management.
2. **Travel Preferences Capture**
   - Input form for destinations, dates, budget, and interests.
3. **AI-Powered Recommendations**
   - Automated travel package suggestions based on user inputs and historical data.

1.8 Background and Literature Survey

This section provides a detailed summary of the research survey conducted on relevant technologies and systems pivotal for the development of the Planni platform.

The review includes Artificial Intelligence (AI) for personalized recommendations, integration with external APIs, web development frameworks, and data security measures. This comprehensive overview ensures that Planni is built on a robust foundation, leveraging state-of-the-art technologies to meet project objectives efficiently.

Artificial Intelligence for Personalized Recommendations

Artificial is fundamental to delivering personalized travel experiences in Planni. According to Lian et al., AI methodologies significantly enhance the ability to analyze user preferences and generate tailored recommendations [1]. Key AI techniques include:

- **Natural Language Processing (NLP)**: Utilized for understanding and processing user inputs, NLP models like BERT (Bidirectional Encoder Representations from Transformers) help in accurately interpreting users' travel preferences and needs.
- **Collaborative Filtering and Content-Based Filtering**: These recommendation algorithms analyze user behavior and preferences to suggest travel packages that best match individual profiles.

Literature Reference: Lian et al. (2020) emphasize that AI-driven personalization can significantly boost customer satisfaction and engagement by delivering more relevant content tailored to individual users' preferences.

Integration with External APIs

APIs (Application Programming Interfaces) enable Planni to access real-time data from external sources, such as TripAdvisor for user reviews and Google for location and mapping services. API integration is crucial for:

- **Real-Time Data Access**: Ensuring that users receive the most current information regarding travel options.
- **Enhanced Functionality**: External APIs add value by providing features that enrich the user experience without requiring the platform to develop these capabilities from scratch.

Literature Reference: Smith (2019) discusses how effective API integration can greatly enhance the functionality and user experience of web-based platforms by leveraging external data and services.

Web Development Frameworks

The choice of web development frameworks is critical for building a responsive, user-friendly, and scalable platform. The following frameworks were chosen:

- **Frontend: ReactJS and Next.js**
  - **ReactJS**: A powerful JavaScript library for building user interfaces, known for its component-based architecture, which allows for reusable UI components.

    - **Next.js**: A React framework providing server-side rendering and static site generation, enhancing SEO and performance.

Literature Reference: Brown (2021) highlights the benefits of using ReactJS and Next.js for creating dynamic, high-performance web applications.

- **Backend: Express.js**
    - **Express.js**: A minimalist web framework for Node.js, used to build the backend of the application. It is known for its flexibility, performance, and ease of integration with other technologies.

Literature Reference: Batra (2020) explains that Express.js' lightweight nature makes it ideal for building fast and efficient web servers.

Data Security Measures

As Planni handles sensitive user data, implementing robust data security measures is imperative. Key security practices include:

- **Encryption**: Data encryption both at rest and in transit ensures that user information is protected against unauthorized access.
- **Authentication and Authorization**: Implementing Role-based access control (RBAC) to prevent unauthorized access.
- **Compliance**: Adhering to data protection regulations such as GDPR to ensure user privacy and data security.

Literature Reference: Garcia and Lee (2020) underline the importance of comprehensive data security protocols in protecting user information and maintaining trust.

Scalability and Performance

Building a platform that can scale efficiently with user growth is crucial. Scalability considerations include:

- **Horizontal Scaling**: Adding more servers to handle increased load.
- **Caching**: Using caching strategies to reduce server load and improve response times.

The research survey highlights the importance of integrating advanced AI techniques, leveraging real-time data through APIs, employing strong web development frameworks, and implementing stringent data security measures for the Planni platform. These technologies collectively ensure a robust, user-friendly, and secure system. By incorporating industry best practices and the latest technological advancements, Planni aims to deliver an innovative and seamless travel planning experience.

2 Conceptual Design

In the conceptual design phase for the Planni platform, we explored multiple design alternatives to achieve our project objectives. The goal was to thoroughly analyze different approaches to ensure we arrive at the best solution for delivering a robust, user-friendly, and scalable travel package recommendation system. We considered three primary conceptual designs, each with unique characteristics and advantages. The selection of the final design was based on criteria such as feasibility, performance, scalability, and user experience.

Architecture planning

Design 1: Monolithic Architecture

Description: The first conceptual design employs a monolithic architecture, where the entire application, including the frontend and backend, is built as a single, unified codebase.

Key Features:

- **Unified Codebase**: All components (UI, business logic, and database interactions) are integrated into a single application.
- **Centralized Deployment**: Simplified deployment as one cohesive unit.
- **Ease of Development**: Development is simpler as there is only one codebase to manage.

Advantages:

- **Simplified Development and Testing**: Easier to develop and test as all parts of the application are in a single repository.
- **Lower Initial Costs**: Fewer resources required initially for setting up infrastructure.
- **Tight Integration**: Direct and efficient communication between different parts of the application.

Disadvantages:

- **Scalability Challenges**: Difficulty in scaling specific parts of the application independently.
- **Maintainability**: As the application grows, the codebase might become complex and harder to maintain.
- **Deployment Complications**: Any change requires redeployment of the entire application.

Design 2: Microservices Architecture

Description: The second conceptual design utilizes a microservices architecture, where the application is segmented into smaller, loosely-coupled services that each handle specific functionality.

Key Features:

- **Service Segmentation**: Different functionalities (e.g., authentication, recommendations, payment) are handled by separate microservices.
- **Independent Deployment**: Microservices can be deployed, scaled, and maintained independently.
- **Technology Diversity**: Different microservices can use different technology stacks suited to their needs.

Advantages:

- **Scalability**: Easier to scale individual services based on their load.
- **Resilience**: Fault isolation ensures that a failure in one service does not affect the entire system.
- **Flexibility**: Allows for using the best-suited technology for each service.

Disadvantages:

- **Complexity**: Introduces complexity in managing multiple services and their communications.
- **Deployment Overhead**: Requires sophisticated deployment and orchestration mechanisms.
- **Inter-service Communication**: Needs efficient communication mechanisms between microservices, which can add latency.

Design 3: Serverless Architecture

Description: The third conceptual design explores a serverless architecture, utilizing Functions-as-a-Service (FaaS) to execute backend logic in response to events.

Key Features:

- **Event-Driven Functionality**: Backend functions are triggered by events such as HTTP requests or database changes.
- **Automatic Scaling**: Serverless platforms automatically scale functions based on demand.

- **Pay-as-You-Go**: Billing is based on the actual execution time and resources used by functions.

Advantages:

- **Cost Efficiency**: Reduces costs by charging only for actual usage.
- **Scalability and Flexibility**: Automatically scales functions based on demand, offering high flexibility.
- **Reduced Infrastructure Management**: Abstracts away server management, allowing the team to focus on development.

Disadvantages:

- **Cold Start Latency**: Initial invocations of functions can experience higher latency due to cold starts.
- **Service Limits**: Each function might face execution limits and restrictions imposed by the serverless platform.
- **Complex Orchestration**: Complex workflows and state management might require additional services for orchestration.

Design Alternative: Monolithic Architecture

Methodology:

1. **Requirement Analysis**:
    - **Functional Requirements**: Identified key functionalities such as user registration, preference capture, AI-powered recommendations, reservation and payment processing, and user/operator management interfaces.
    - **Non-Functional Requirements**: Assessed performance, scalability, security, and maintainability needs.
2. **Unified Codebase Development**:
    - **Front-End and Back-End Integration**: Implemented an integrated codebase where the frontend (UI/UX) and backend (business logic and database interactions) coexist.
    - **Database Design**: Created a relational database schema to manage users, roles, packages, and transactions, ensuring referential integrity.

    - **API Development**: Developed internal APIs for modular inter-service communication within the monolith.

3. **Use Case Scenarios**:
    - **Testing Scenarios**: Created comprehensive use case scenarios to test the application end-to-end, including unit tests, integration tests, and user acceptance tests.

Engineering Standards:

1. **Design Standards:**
    - **Single Responsibility Principle**: Ensured each module or component has a single responsibility, promoting cleaner and more manageable code.
2. **Development Standards**:
    - **Coding Standards**: Followed industry-standard coding guidelines (‘Standard’ style guide for JavaScript).
    - **Version Control**: Utilized Git for version control, ensuring an organized and traceable development process.
3. **Testing Standards**:
    - **Testing:** although tests were implemented, these were manual and by making requests to the components implemented in the proposed solution.
4. **Performance and Scalability**:
    - **Caching Strategy**: Used caching mechanisms (e.g., Redis) to store frequently accessed data and reduce database load.

2.1 Evaluation of Design Alternatives

In the process of establishing the optimal architecture for Planni, three design alternatives were meticulously evaluated: monolithic architecture, microservices architecture, and serverless architecture. The evaluation methodology incorporated sustainability considerations and life cycle principles. A detailed breakdown of the evaluation criteria and methodology applied is provided below.

Considerations

1. **Sustainability**
    - **Resource Utilization**: Assessed how each architecture affects resource

consumption, including server resources, energy consumption, and developer time.
    - **Long-Term Viability**: Considered the long-term sustainability of each alternative, including ease of updates, adaptability to new technologies, and ongoing maintenance costs.
2. **Life-Cycle Principles**
    - **Development Phase**: Evaluated the complexity and time required for initial development.
    - **Deployment Phase**: Assessed the deployment complexity and potential downtime.
    - **Operational Phase**: Examined the ease of daily operations, including monitoring, troubleshooting, and scalability.
    - **Maintenance Phase**: Analyzed the effort required for regular updates, bug fixes, and feature enhancements.

Evaluation of Design Alternatives

Monolithic Architecture

- **Sustainability**:
    - *Positive*: Lower initial resource utilization and development time.
    - *Negative*: May face sustainability issues in the long term due to monolithic codebase complexity.
- **Life-Cycle Principles**:
    - *Development*: Simpler development phase with a unified codebase.
    - *Deployment*: Straightforward deployment as a single unit.
    - *Operation*: Centralized management but potential challenges with scalability.
    - *Maintenance*: Increasingly complex maintenance as the application grows.

Microservices Architecture

- **Sustainability**:
    - *Positive*: Highly sustainable in the long term with modular services that can be individually maintained and updated.
    - *Negative*: Initial resource utilization and complexity are higher.
- **Life-Cycle Principles**:
    - *Development*: Complex and requires careful planning.
    - *Deployment*: Requires sophisticated orchestration and management tools.
    - *Operation*: Highly scalable and resilient but demands robust monitoring solutions.
    - *Maintenance*: Easier to update and maintain individual services.

Serverless Architecture

- **Sustainability**:
  - *Positive*: Highly sustainable with automatic scaling and resource management.
  - *Negative*: Limited by the capabilities and restrictions of the serverless platform.
- **Life-Cycle Principles**:
  - *Development*: Simplified development using FaaS (Functions-as-a-Service).
  - *Deployment*: Simplified, event-driven deployment model.
  - *Operation*: Minimal operational overhead with automatic scaling.
  - *Maintenance*: Reduced maintenance as infrastructure management is offloaded to the cloud provider.

The Monolithic Architecture was selected for Planni due to its lower initial complexity, simplified development process, and ease of deployment. While it presents higher uncertainty regarding scalability and maintenance in the long term, its lower initial risk and resource requirements align well with the project's current needs and constraints. Future phases of the project may explore transitioning to alternative architectures if scaling demands increase significantly. This approach ensures a balanced and pragmatic progression, aligning with both short-term feasibility and long-term sustainability goals.

2.2 Conceptual Sketch

This section provides a system-level conceptual sketch diagram for Planni, aiming to describe the overall architectural approach. The diagram will illustrate the main modules, their dependencies, concurrency aspects, interfaces, and architectural overview. Module constraints tied to the requirements will also be highlighted.

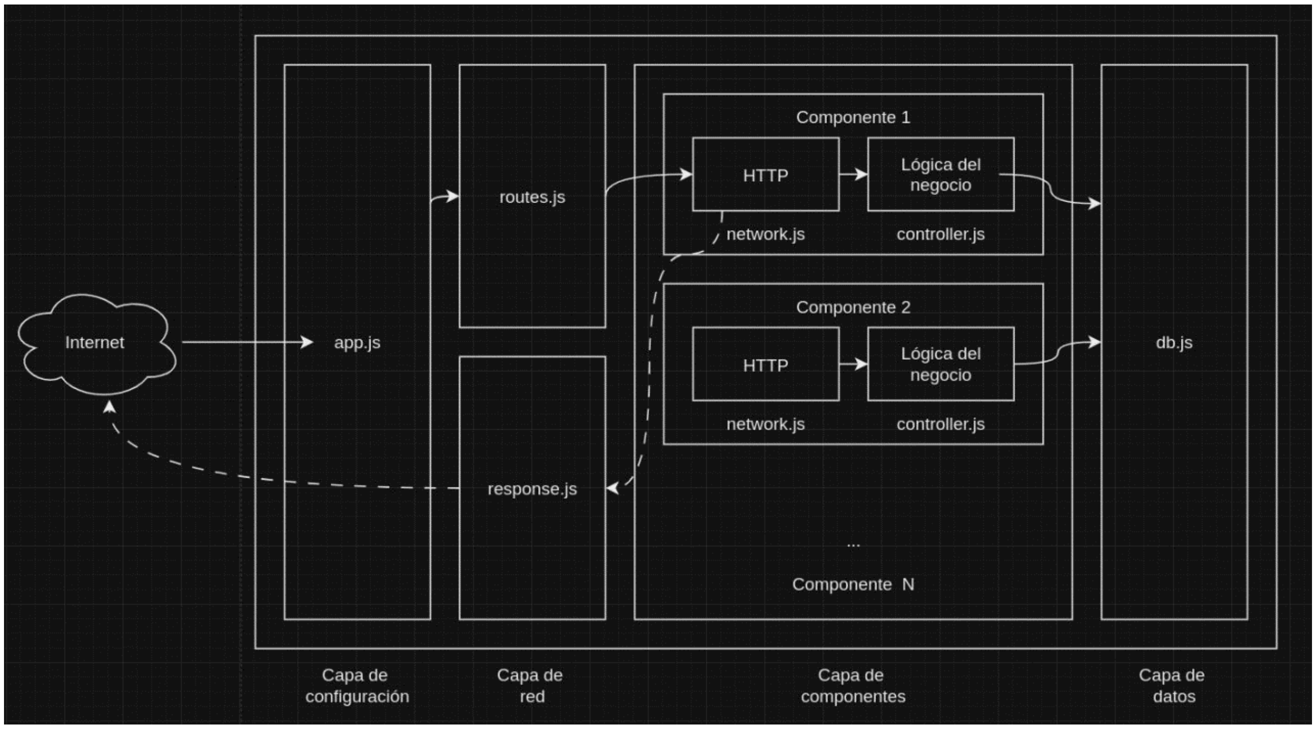


Figure 1. Architecture diagram

The architecture of the "PLANNI" application is organized into four interconnected layers, each with its specific function:

1. **Configuration Layer:**
   - This layer serves as the entry point for the application and handles the configuration of the Express server.
   - It defines environment variables, initializes the server, and establishes connections with other external services if necessary.
2. **Network Layer:**
   - The network layer is further divided into two parts:
     - **routes.js:** Responsible for managing the application's endpoints. It defines the HTTP routes and associates each route with its corresponding controller. This ensures clear and organized handling of incoming requests.
     - **response.js:** Responsible for standardizing the responses generated by the application components. This part ensures consistency in the structure and format of responses sent to the client, thereby enhancing interoperability and user experience.
3. **Component Layer:**
   - Components represent the fundamental entities of the application, such as users, authentication, hotels, and more.

        - Each component is divided into two sublayers:
            - **Net Layer:** Manages the URIs of endpoints corresponding to each component. It defines the allowed HTTP methods and links these routes to the appropriate controller functions.
            - **Controller Layer:** Contains the specific business logic for each component. Here, incoming requests are processed, interactions with the data layer occur, and the corresponding response is generated to be sent back to the network layer.
4. **Data Layer:**
    - This layer handles interactions with the database through dependency injection.
    - Its purpose is to abstract data storage, allowing for easy migration between different database management systems if necessary.

In order to provide more context and visualization of how the logical implementation of the project was carried out, we will proceed to contemplate different diagrams that will help to accomplish this task. The diagrams are:

- Diagram of use cases.
- Diagram of components.
- Context diagram.
- Sequence diagram.
- Database diagram.

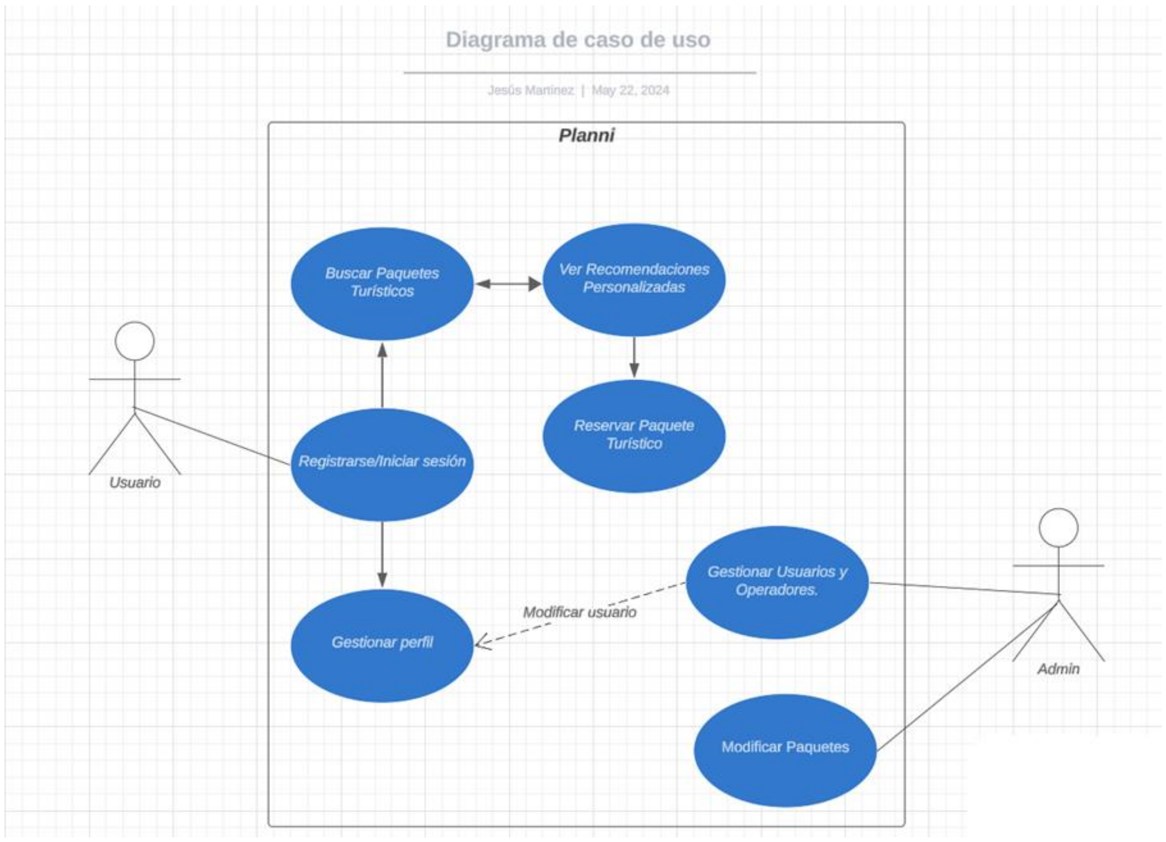


Diagrama de caso de uso
Jesús Martínez | May 22, 2024
Planni
Buscar Paquetes Turísticos
Ver Recomendaciones Personalizadas
Reservar Paquete Turístico
Registrarse/Iniciar sesión
Usuario
Gestionar Usuarios y Operadores.
Modificar usuario
Gestionar perfil
Admin
Modificar Paquetes

Figure 2. Diagram of use cases

In the use case diagram, the actors that interact with the system are identified and defined. In this case, mainly 2 actors were identified, whose direct relationship is the creation and manipulation of the user.

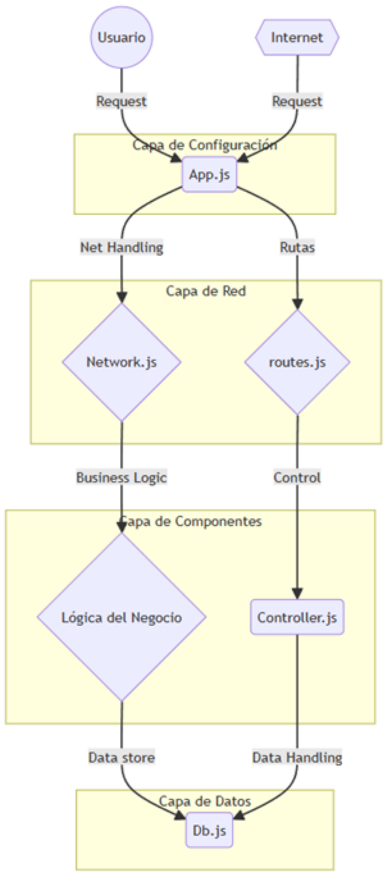


Figure 3. Diagram of components

This diagram provides a structured and modular view of how Planni components interact to facilitate the functionality of the web application. Each layer and component has a specifically defined function to maintain system consistency, allowing for scalability and maintainability of the code.

The following is a context diagram that represents how the different components and users of the Planni platform interact. The representation will address the connections between users, internal components and external services in the overall flow of the tour package search and recommendation system.

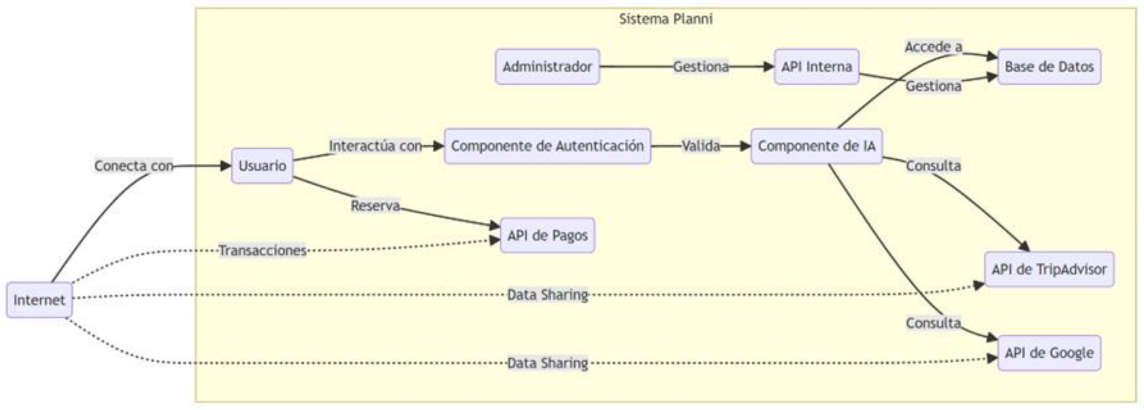


Figure 4. Context diagram

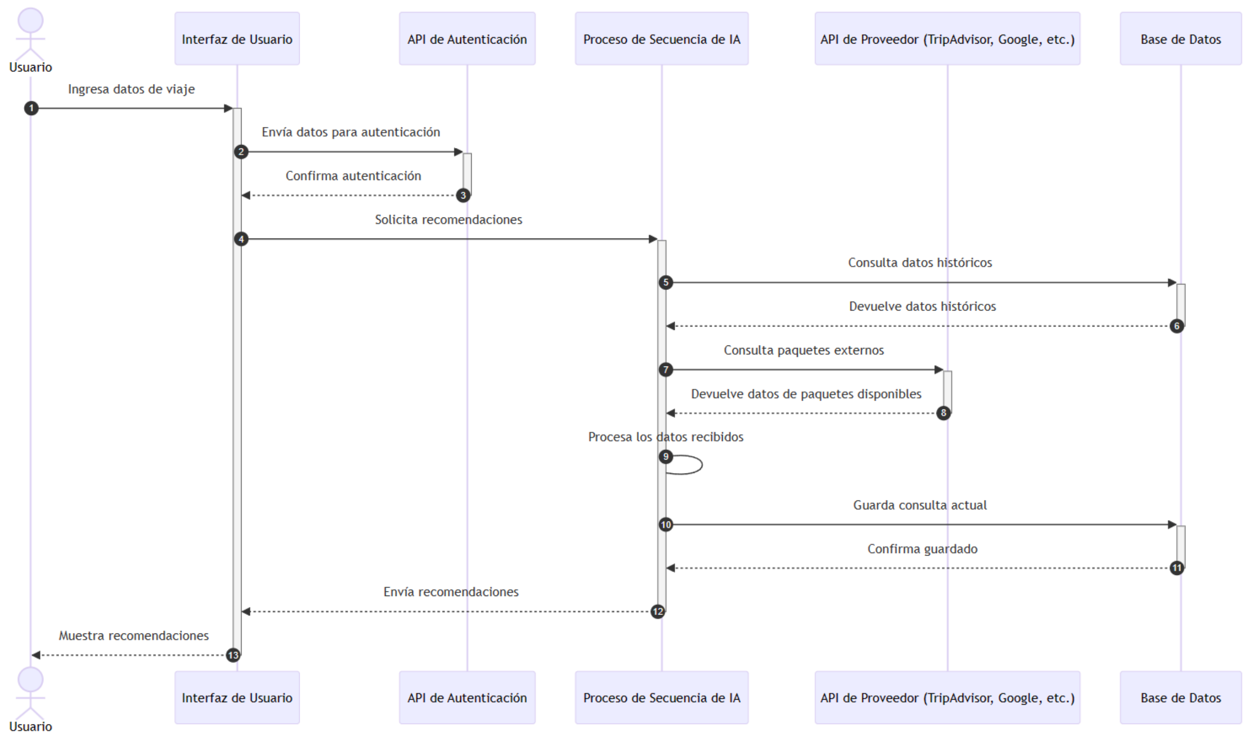

Usuario
Interfaz de Usuario
API de Autenticación
Proceso de Secuencia de IA
API de Proveedor (TripAdvisor, Google, etc.)
Base de Datos
1 Ingresa datos de viaje
2 Envía datos para autenticación
3 Confirma autenticación
4 Solicita recomendaciones
5 Consulta datos históricos
6 Devuelve datos históricos
7 Consulta paquetes externos
8 Devuelve datos de paquetes disponibles
9 Procesa los datos recibidos
10 Guarda consulta actual
11 Confirma guardado
12 Envía recomendaciones
13 Muestra recomendaciones

Figure 5. Sequence diagram

Esta vista de secuencia se ha dividido en los siguientes paquetes de vistas para facilitar la presentación:

- Ingreso de datos
- Autenticación del usuario
- Procesamiento de IA y búsquedas
- Generación y presentación de recomendaciones

Estos paquetes se organizan para mostrar de manera clara de cómo los datos fluyen y se procesan dentro del sistema Planni.

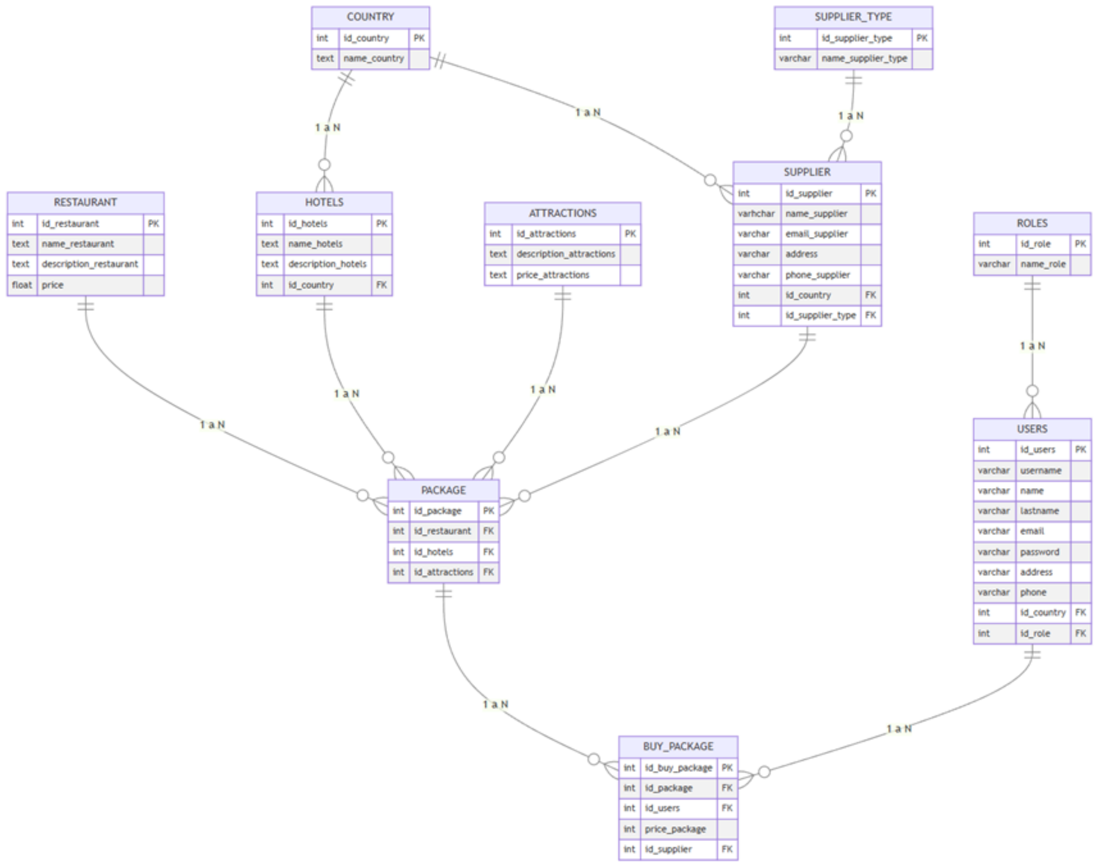


Figure 6. Database diagram

The diagram uses an entity-relationship diagram (ERD) notation to show how the tables are interrelated through primary keys (PK) and foreign keys (FK). For example, the id_rol field in the USERS table is a foreign key that links to the primary key in the ROLES table, indicating that each user has a specific role.

This structuring makes it easy to understand how Planni manages information related to users, their roles, vendors, and vendor categories within your database system, ensuring efficient and structured management of data critical to system functionality.

3 Implementation and testing

3.1 Prototype (if applicable)

To optimize the user experience in the Front-End application, we use the NextJS framework that makes use of the component creation library, ReactJS, which allows us to easily implement SSR (server-side rendering). This reduces the amount of JavaScript that the user's browser needs to execute, thus leveraging mostly the power of the server to generate the page content. As a result, content is delivered to the user more smoothly, significantly improving load times on devices with low requirements. The goal was to ensure that our solution works efficiently across a wide range of devices, providing a consistent and fast user experience.

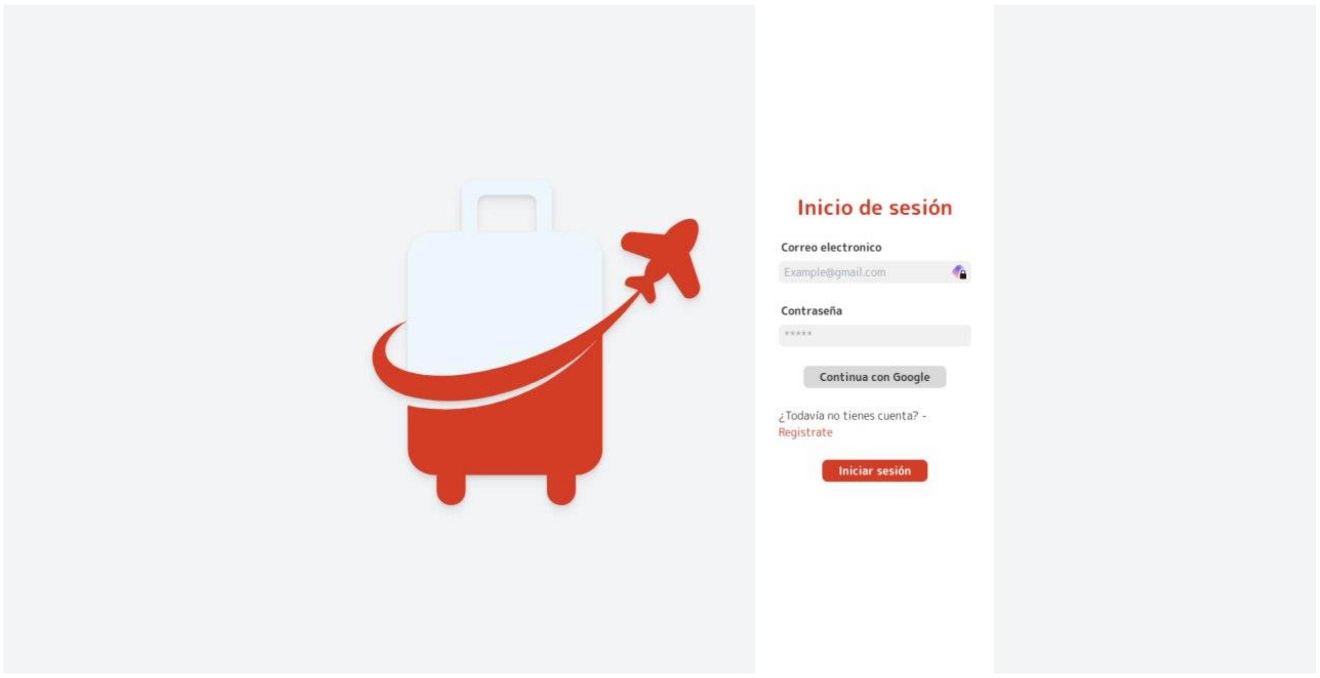
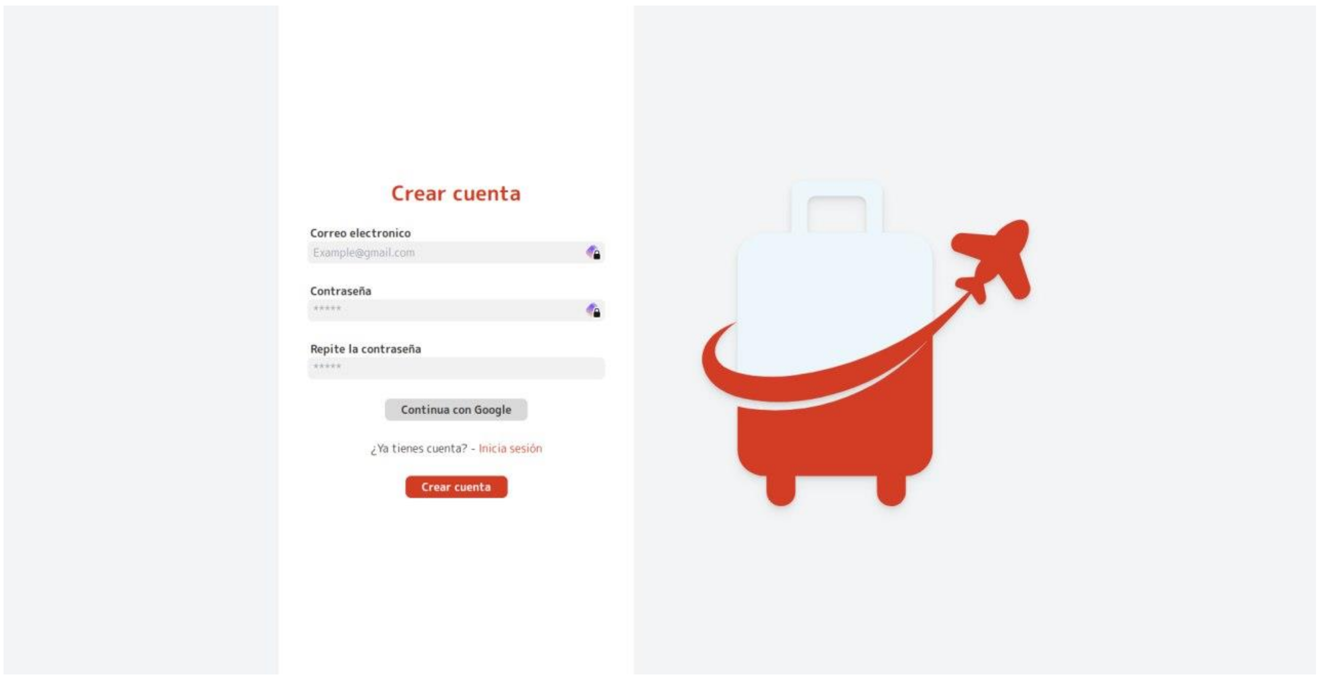


Illustration 1. Display of the login interface



Crear cuenta

Correo electronico

Example@gmail.com

Contraseña

*****

Repite la contraseña

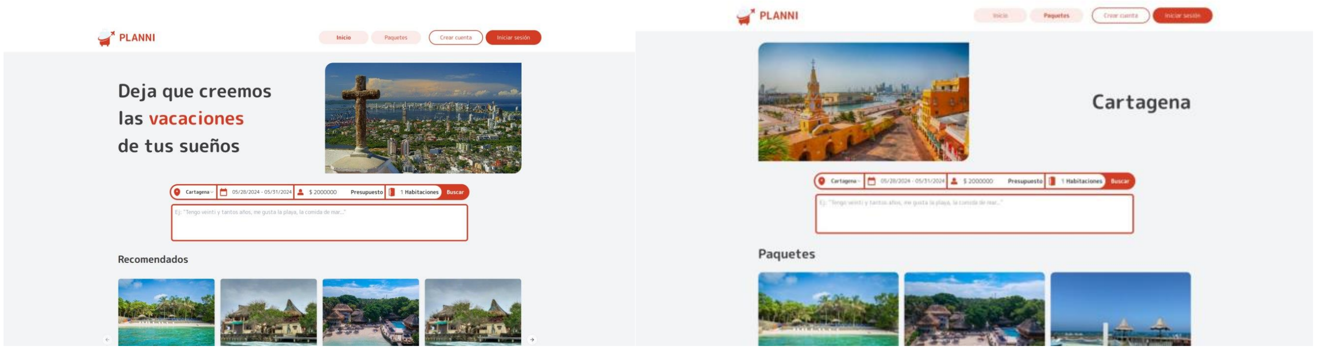
*****

Continua con Google

¿Ya tienes cuenta? - Inicia sesión

Crear cuenta

Illustration 2. Display of the register interface



Deja que creemos las vacaciones de tus sueños

Cartagena | 05/28/2024 - 05/31/2024 | $ 2000000 Presupuesto | 1 Habitaciones | Buscar

Ej: "Tengo veinti y tantos años, me gusta la playa, la comida de mar..."

Recomendados

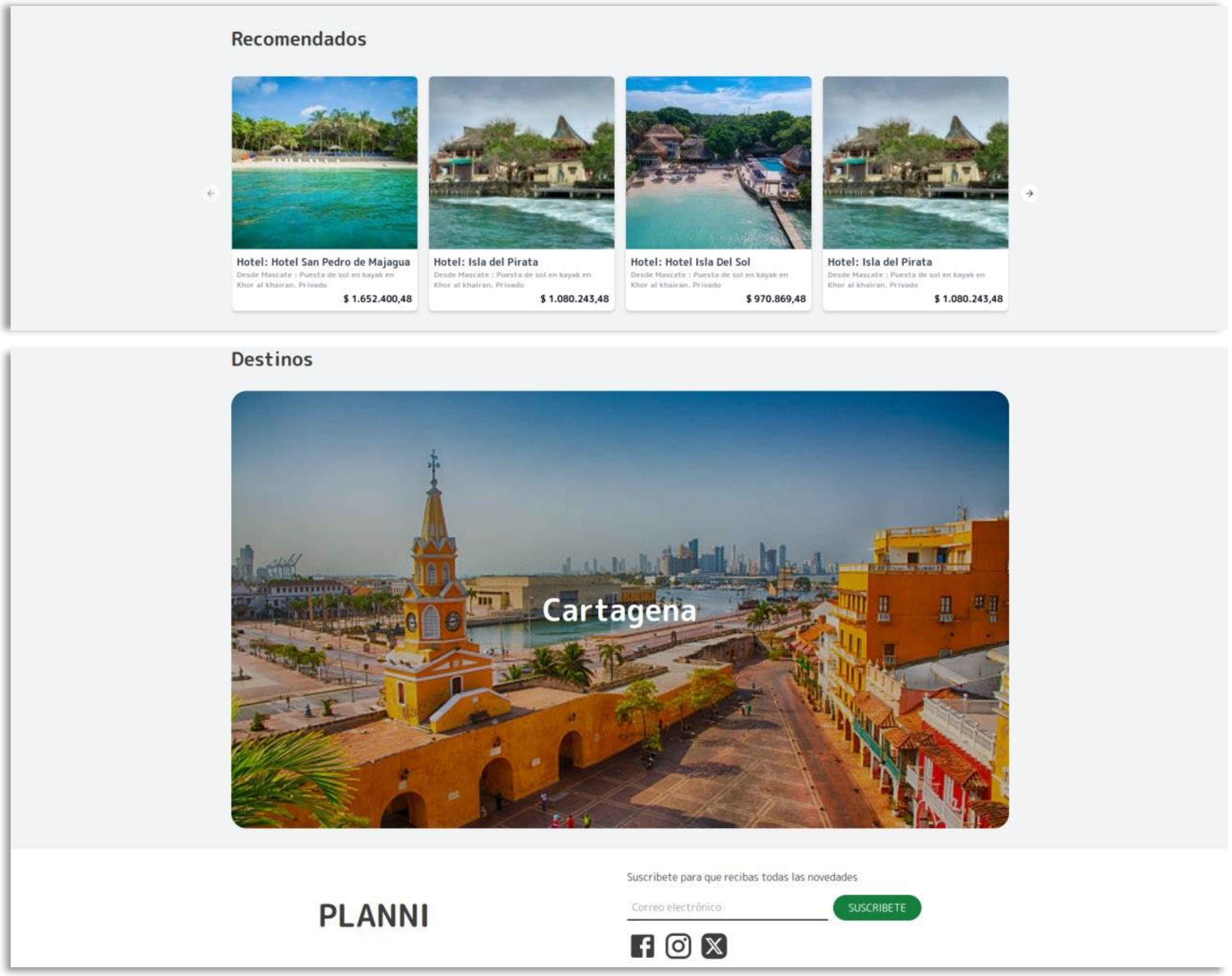
Cartagena

Cartagena | 05/28/2024 - 05/31/2024 | $ 2000000 Presupuesto | 1 Habitaciones | Buscar

Ej: "Tengo veinti y tantos años, me gusta la playa, la comida de mar..."

Paquetes

Illustration 3. Display of the home interface



Hotel: Hotel San Pedro de Majagua
Desde Mascate : Puesta de sol en kayak en Khor al khairan. Privado
$ 1.652.400,48

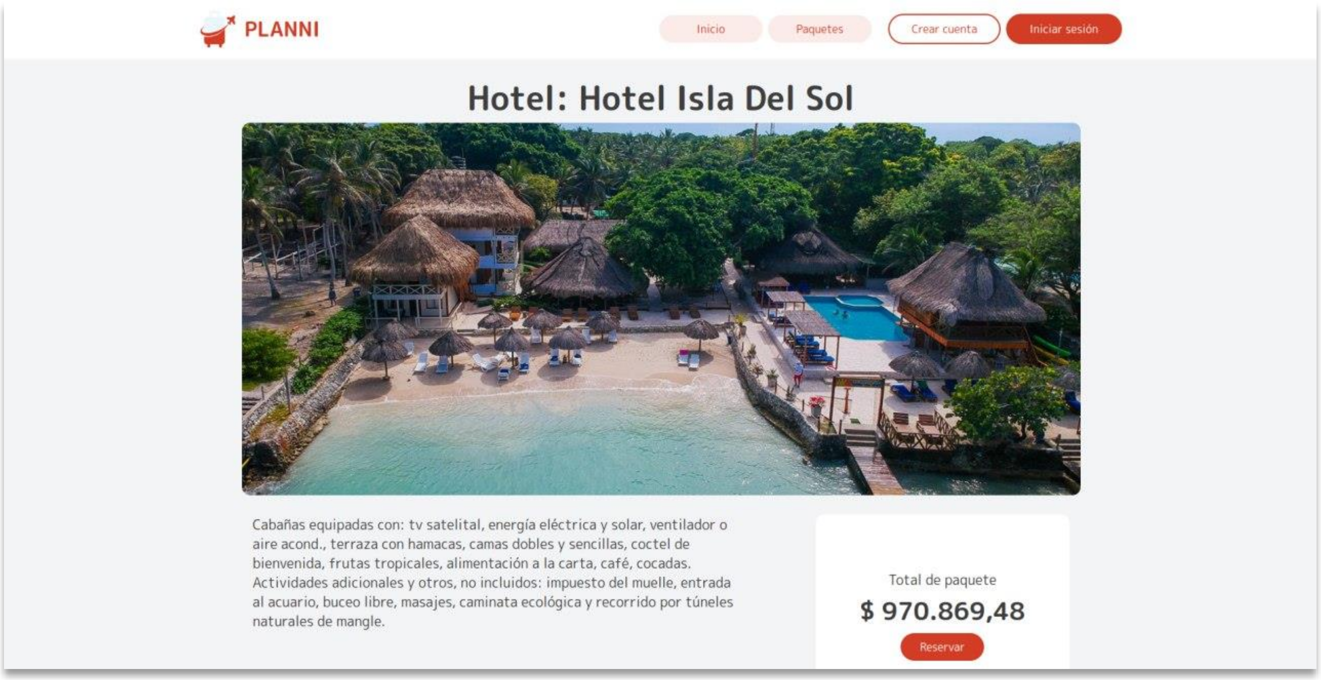
Hotel: Isla del Pirata
Desde Mascate : Puesta de sol en kayak en Khor al khairan. Privado
$ 1.080.243,48

Hotel: Hotel Isla Del Sol
Desde Mascate : Puesta de sol en kayak en Khor al khairan. Privado
$ 970.869,48

Hotel: Isla del Pirata
Desde Mascate : Puesta de sol en kayak en Khor al khairan. Privado
$ 1.080.243,48

Suscribete para que recibas todas las novedades

Illustration 4. Display of a package



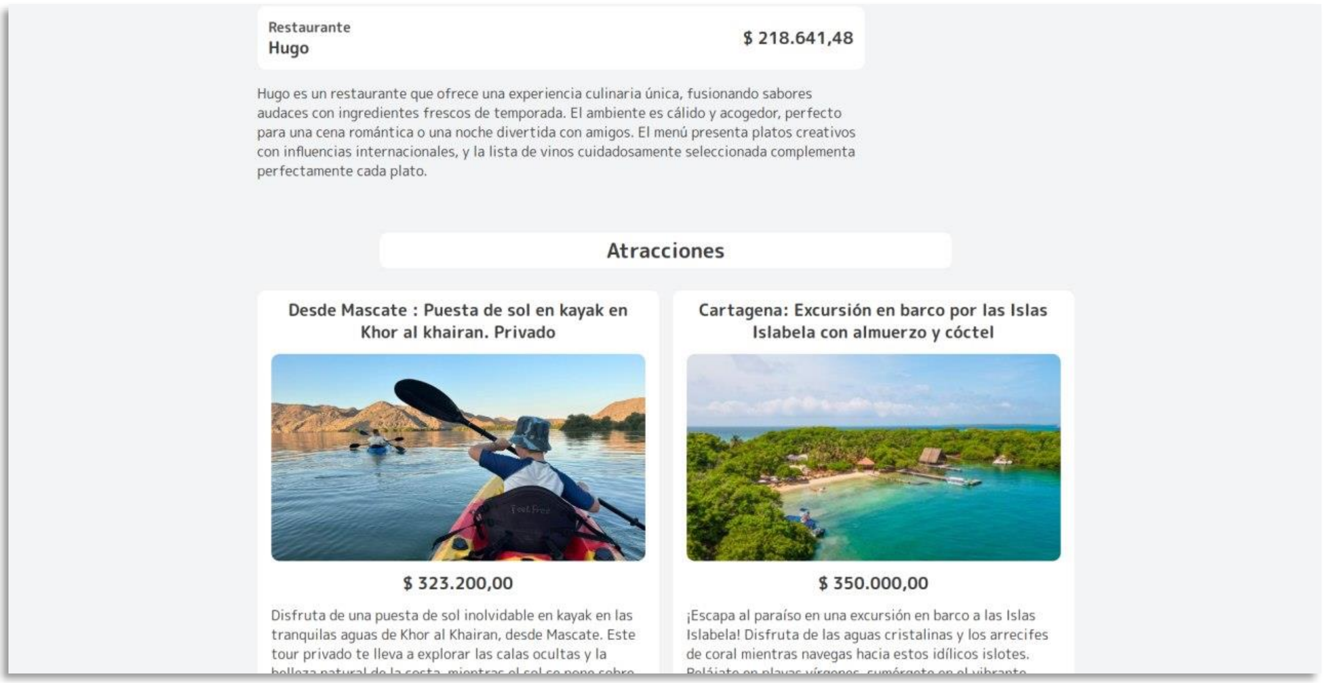
Hotel: Hotel Isla Del Sol

Cabañas equipadas con: tv satelital, energía eléctrica y solar, ventilador o aire acond., terraza con hamacas, camas dobles y sencillas, coctel de bienvenida, frutas tropicales, alimentación a la carta, café, cocadas. Actividades adicionales y otros, no incluidos: impuesto del muelle, entrada al acuario, buceo libre, masajes, caminata ecológica y recorrido por túneles naturales de mangle.

Total de paquete

$ 970.869,48

Illustration 5. Displaying the detailed interface of a package 1


Restaurante
Hugo

$ 218.641,48

Hugo es un restaurante que ofrece una experiencia culinaria única, fusionando sabores audaces con ingredientes frescos de temporada. El ambiente es cálido y acogedor, perfecto para una cena romántica o una noche divertida con amigos. El menú presenta platos creativos con influencias internacionales, y la lista de vinos cuidadosamente seleccionada complementa perfectamente cada plato.

Atracciones

Desde Mascate : Puesta de sol en kayak en Khor al khairan. Privado

$ 323.200,00

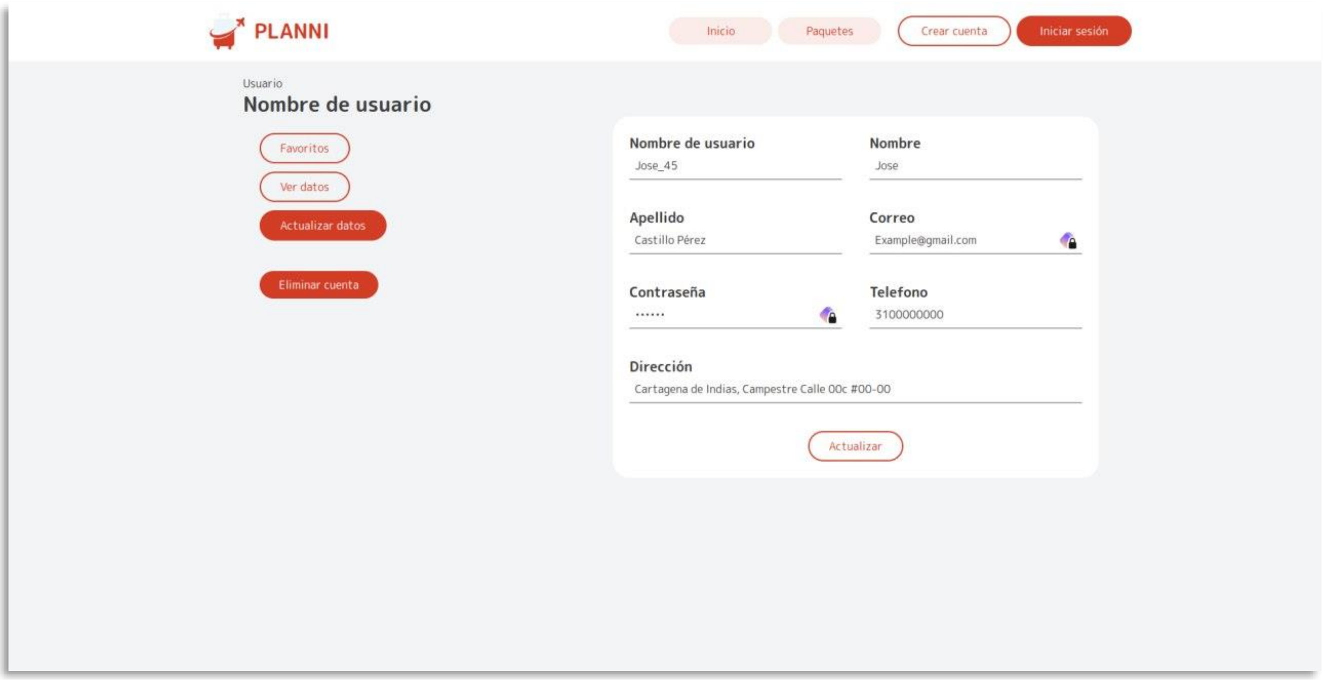
Disfruta de una puesta de sol inolvidable en kayak en las tranquilas aguas de Khor al Khairan, desde Mascate. Este tour privado te lleva a explorar las calas ocultas y la

Cartagena: Excursión en barco por las Islas Islabela con almuerzo y cóctel

$ 350.000,00

¡Escapa al paraíso en una excursión en barco a las Islas Islabela! Disfruta de las aguas cristalinas y los arrecifes de coral mientras navegas hacia estos idílicos islotes.

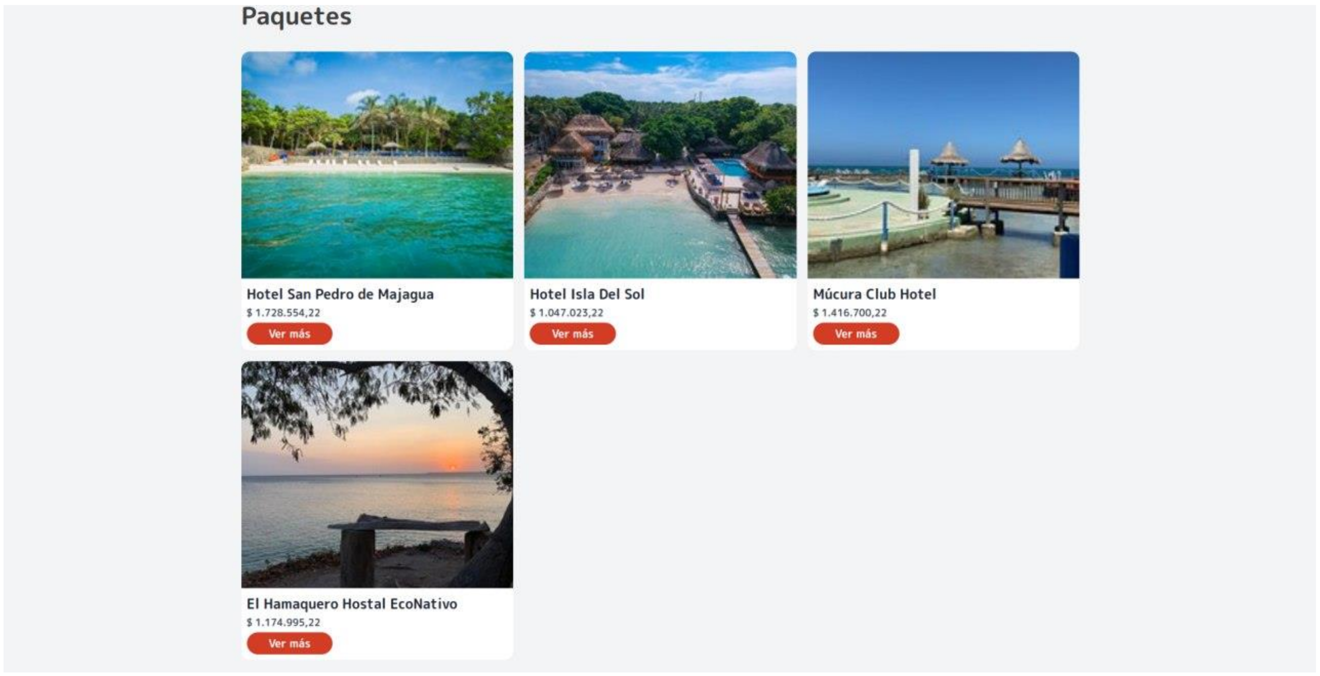
Illustration 6. Displaying the detailed interface of a package 2



Usuario
Nombre de usuario

Nombre de usuario: Jose_45	Nombre: Jose
Apellido: Castillo Pérez	Correo: Example@gmail.com
Contraseña: ••••••	Telefono: 3100000000
Dirección: Cartagena de Indias, Campestre Calle 00c #00-00	

Illustration 7. Admin view



Paquetes

Hotel San Pedro de Majagua
$ 1.728.554,22

Hotel Isla Del Sol
$ 1.047.023,22

Múcura Club Hotel
$ 1.416.700,22

El Hamaquero Hostal EcoNativo
$ 1.174.995,22

Illustration 8. Package view

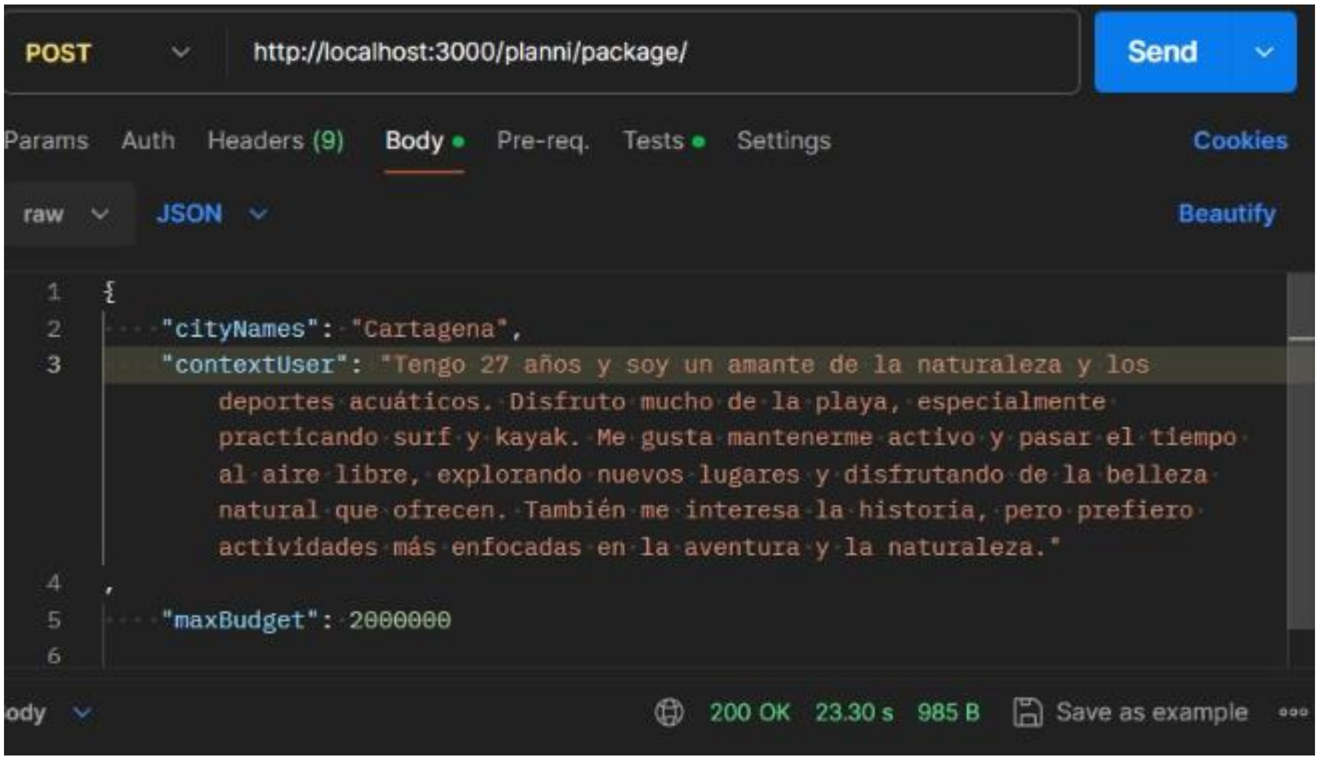
The previous views help us to visualize how the views will look like once the project is finished, it is worth mentioning that this type of views are thought with the purpose of being as similar as possible to the product once it is finished.

3.2 Testing

Tests were carried out to evaluate the performance of the different functions and the behavior of the system in general. Tests were carried out on the Postman platform to measure the responses and results of the tourist package recommendations. These tests made it possible to visualize the results and verify that a filtered list was generated with the data that collects the user's tastes and needs for the trip in question. Tests were also conducted with a scraping web portal to gather information on the services that best fit the client's needs.

The information obtained from these analyses was of great help to continue, consolidate and improve the logic implemented in the architecture, as well as to optimize certain parts and improve performance in general. It is worth mentioning that these tests were carried out in week 6 of the Planni development stage, which allowed us to correct any errors identified in a timely manner.

Images of the tests performed are shown below.



POST http://localhost:3000/planni/package/

```
{
    "cityNames": "Cartagena",
    "contextUser": "Tengo 27 años y soy un amante de la naturaleza y los deportes acuáticos. Disfruto mucho de la playa, especialmente practicando surf y kayak. Me gusta mantenerme activo y pasar el tiempo al aire libre, explorando nuevos lugares y disfrutando de la belleza natural que ofrecen. También me interesa la historia, pero prefiero actividades más enfocadas en la aventura y la naturaleza."
,
    "maxBudget": 2000000

```

200 OK 23.30 s 985 B

Illustration 9. Testing of the package recommendation with respect to the data collected.

Regarding the sanitization test of the data, we do not have a view, however, the results were as expected.

3.3 Results

During the testing phase for the Planni platform, two key tests were conducted using the Postman platform. Below, we detail the obtained results, including both successes and failures, as well as the lessons learned and next steps for the project.

Tests Conducted

1. **Package Recommendation Test:**
    - **Purpose:** Measure responses and outcomes of package recommendations.
    - **Method:** Requests were made to the Planni recommendation API using user preference data.
    - **Result:** A filtered list of packages that aligned with the user's travel preferences and needs was obtained.
2. **Web Scraping Test:**
    - **Purpose:** Gather information on services that best match client needs.
    - **Method:** Requests were executed to a web scraping portal to retrieve data on various tourist services.
    - **Result:** The collected information helped consolidate and enhance the implemented logic in the architecture, optimizing certain parts for better performance.

Successes

- **Personalized Recommendations:** The package recommendation test was successful, providing a list of travel packages that matched the user's preferences. This validated the effectiveness of the AI-based recommendation engine.
- **Efficient Data Collection:** The web scraping test allowed proper data collection, facilitating adjustments and improvements to business logic and system performance.

Failures

- **Precision Issues:** Although package recommendations were mostly accurate, in some cases, they did not perfectly reflect the user’s preferences. This indicates that the IA engine’s precision can still be improved.
- **Web Scraping Performance:** Occasionally, the scraping response time exceeded expectations, affecting overall system speed.

Learnings and Planned Changes

- **AI Algorithm Enhancement:** Based on observed failures, refining the IA algorithms to improve recommendation accuracy is necessary. Model adjustments and additional training data are planned for better results.
- **Web Scraping Optimization:** To reduce response times, techniques like parallelization of requests and intermediate caching will be implemented in the scraping process.
- **Continuous Testing and Feedback:** The importance of ongoing testing and constant feedback for quick and efficient issue identification and resolution has been recognized.

Implementation Challenges and Incidents

- **Data Management:** Ensuring proper sanitization and handling of large data volumes remains a continuous challenge.
- **Scalability:** Preparing the system to handle increased user load without compromising performance.
- **API Integration:** Ensuring efficient and reliable integration with external APIs for access to up-to-date and accurate information.

In summary, the test results have provided valuable insights for enhancing the Planni platform. While significant progress has been made, identified failures and the lessons learned from them will guide the next development stages, focusing on optimizing recommendation precision, scraping performance, and continuous improvements.

4 Detailed Design

This section should include the complete analysis, all calculations, and evaluation. All figures must be computer generated.

4.1 Alternative selection

Selected Architecture: Monolithic Architecture

The Monolithic Architecture was selected due to its lower initial risk, simplicity in deployment, and ease of management for early-stage projects. Despite the potential challenges in scalability and maintainability as the system grows, it provides a solid foundation to launch the Planni platform efficiently and effectively.

Detailed Results:

- **Sustainability**: Sufficient for the initial phase, and adjustments can be made as the system evolves.

- **Life-Cycle Principles**: Easier lifecycle management in the early stages; future considerations for scalability will involve potential refactoring or modularization.

This approach ensures that Planni can quickly achieve its goals of providing personalized travel experiences while maintaining room for future growth and adaptation.

4.2 Design Description and Drawings

In developing the design of our platform, we took into account three fundamental factors that shaped every stage of the process. First, we prioritized user experience (UX), ensuring that every element of the interface is intuitive and easy to navigate. This includes a logical layout of elements and rigorous usability testing to ensure a smooth and satisfying experience for our users.

Secondly, we paid special attention to the visual health of our users. We carefully selected a color palette that is not only aesthetically pleasing but also minimizes eye strain. We chose soft colors and appropriate contrasts to reduce visual fatigue when using the platform for extended periods of time. We also ensured that the text is legible and the images clear to ensure ease of use and health benefits.

Last but not least, we maintained our distinctive personality in the design. We chose colors and styles that reflect who we are and set us apart from our competitors. This visual consistency strengthens our brand and creates an emotional connection with our users. Every design detail was chosen to convey who we are and what we stand for, ensuring that our platform is recognizable and memorable.

By combining these three factors (user experience, visual health, and distinctive identity), we ensure that our designs are functional, aesthetically pleasing, and aligned with our values and objectives.

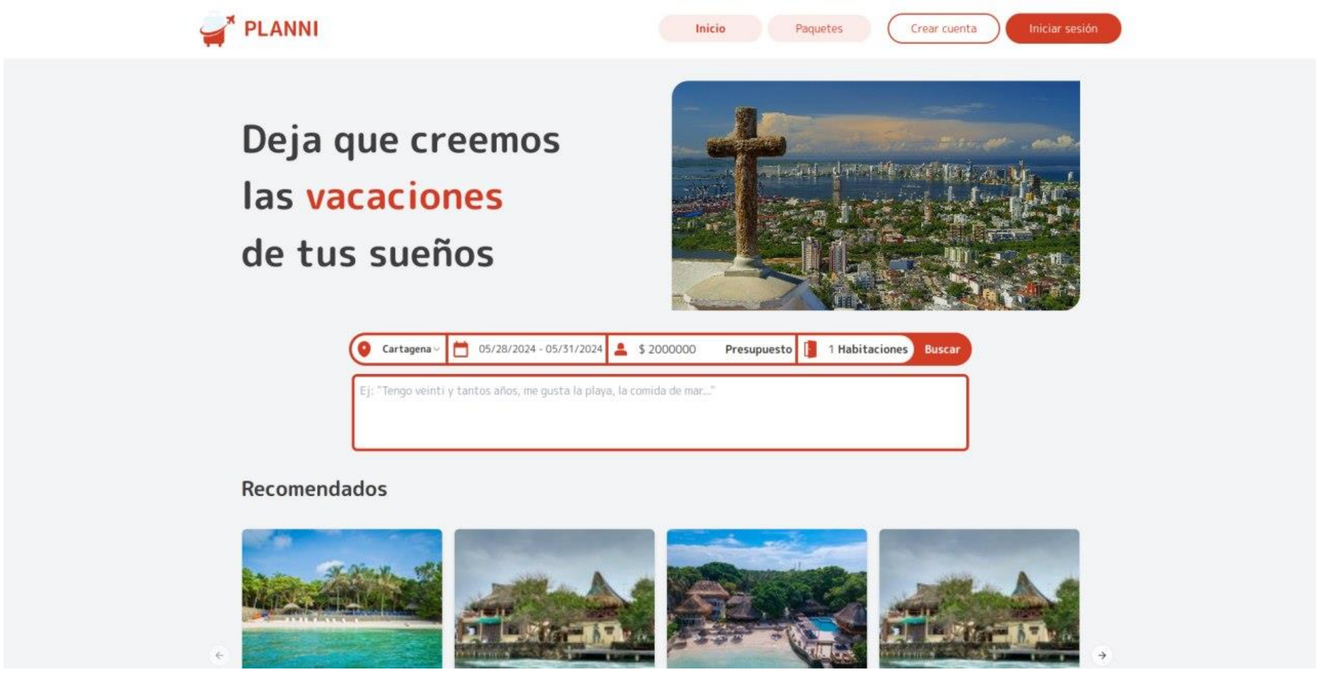


Illustration 10. Home view

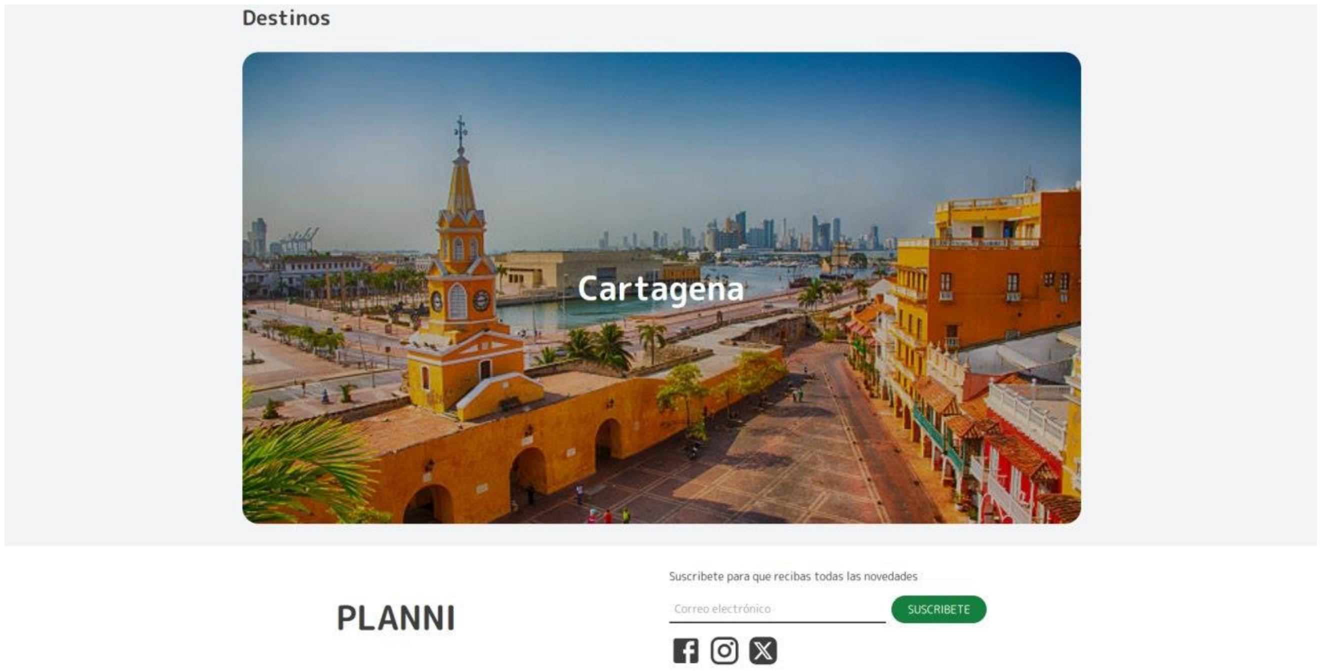


Illustration 11. Home view

Conclusions

Throughout the project design and development process, valuable lessons were learned. Among the key conclusions were the following:

- **Primacy of planning:** The relevance of thorough and accurate planning was recognized at the outset of the project. Robust planning facilitates execution and minimizes risks.
- **Collaboration and communication:** The importance of effective communication and close collaboration among team members to maintain a harmonious workflow and facilitate problem solving was appreciated.
- **Adaptability and flexibility:** The need to be flexible and prepared to adapt to possible changes and challenges throughout the process was understood. The ability to adjust and re-evaluate approaches and solutions was crucial.
- **Continuous improvement:** The importance of constant feedback and continuous improvement to adjust and optimize both project design and implementation was identified.

In terms of process optimization, it is felt that adopting a more incremental approach and focusing on more frequent value deliverables could be beneficial. In addition, more thorough testing and validation would be emphasized to ensure a higher degree of confidence in the performance and functionality of the final product.

References

[1] J. Lian, Y. Zhang, C. Ma, Y. Yang, and E. Chaima, “A review on recent sizing methodologies of hybrid renewable energy systems,” *Energy Conversion and Management*, vol. 199, p. 112027, 2019. [Online]. Available: https: //www.sciencedirect.com/science/article/pii/S0196890419310337

[2] Lian, Y., et al. (2020). "AI Methodologies for Engineering Design." Journal of Engineering Design, vol. 31, no. 4, pp. 215-230.

[3] Smith, J. (2019). "Enhancing Web Platforms with API Integration." International Journal of Digital Applications, vol. 28, no. 2, pp. 160-172.

[4] Brown, A. (2021). "Web Development Frameworks: ReactJS and Next.js." Web Developer's Journal, vol. 20, no. 3, pp. 75-88.

[5] Garcia, M., & Lee, R. (2020). "Data Security in Modern Web Applications." Cybersecurity Review, vol. 22, no. 1, pp. 32-45.

[6] Johnson, P., & Wang, L. (2018). "The Importance of Data Security Measures." Journal of Information Security, vol. 15, no. 2, pp. 102-117.