Palmer Technology
palmertechn@gmail.com
Cochabamba – Bolivia

MANUAL TÉCNICO

SISTEMA DE RESERVA DE AMBIENTES
“RESERVA FÁCIL”

Palmer Technology
palmertechn@gmail.com
Cochabamba – Bolivia

1. Introducción

Con el fin de ofrecer una guía exhaustiva para comprender el diseño y funcionamiento del sistema en cuestión, se ha elaborado este presente manual. Su propósito principal es la de proporcionar al lector “usuario” una vista detallada de como fue concebido el sistema “Reserva fácil”, así como instrucciones precisas para interactuar con el programa de manera efectiva y llevar a cabo su actualización o mantenimiento adecuado en caso de presentar fallos.

En este sentido, se ha estructurado el contenido de manera que abarque todos los aspectos relevantes del sistema. Se incluyeron el código fuente utilizado, así como cualquier otro elemento que pueda resultar de interés para el programador encargado de su gestión.

Mediante una exposición clara y concisa, se pretende facilitar el entendimiento del sistema y proporcionar las herramientas necesarias para su correcta administración y desarrollo continuo. Este documento se presenta como una herramienta fundamental para aquellos involucrados en el mantenimiento y evolución del sistema, brindando una base sólida para futuras actualizaciones y mejoras.

2. Alcance

Este manual está pensado para ser utilizado por profesionales de distintas áreas, como desarrolladores, ingenieros, personal técnico y administradores. Proporciona información técnica esencial sobre la estructura del sistema y prácticas recomendadas para el mantenimiento eficiente. Su enfoque general y adaptable lo hace relevante para una variedad de roles dentro del contexto del proyecto (Reserva de ambientes “Reserva fácil”).

3. Requerimientos técnicos

Con el fin de garantizar el éxito en la implementación del Sistema de Reserva de ambientes "Reserva fácil", diseñado para solicitar, reservar ambientes de la universidad mayor de san simon “UMSS”, se detallan los requisitos de software esenciales que constituirán la base sólida para asegurar una implementación efectiva y un rendimiento óptimo del sistema.

1) Sistema operativo.-

a. Sistemas Operativos Compatibles:

- **Windows:**
  - Windows 7 (SP1 o posterior)
  - Windows 8/8.1
  - Windows 10
  - Windows 11

- **Linux:**
  - Ubuntu 14.04 LTS o posterior
  - Otras distribuciones de Linux pueden ser compatibles, pero no han sido probadas oficialmente.

2) Base de datos.-

a. Para la gestión de la base de datos, con la capacidad de escalabilidad necesaria para manejar un gran volumen de datos.
- PostgreSQL 10.23

3) Lenguajes de Programación.-

a. Se requerirá PHP en su versión compatible 7.4.22
b. Se requerirá Laravel en su versión compatible 8
c. Se requerirá JavaScript

4) Manejador de Dependencias.-

a. Composer 2.7.2

5) Navegadores Web.-

a. **Google Chrome:** Versión 60 y posteriores.
b. **Mozilla Firefox:** Versión 55 y posteriores.
c. **Microsoft Edge:** Versión 79 y posteriores (basado en Chromium).
d. **Safari:** Versión 11 y posteriores.
e. **Opera:** Versión 47 y posteriores.

6) Conexión a Internet.-

a. Se requerirá una velocidad de internet mínima de 2MB para el acceso al sistema web.

7) Seguridad.-

a. Protección Contra Ataques Comunes
   i) Protección CSRF: Implementa tokens CSRF en formularios y solicitudes que cambian el estado del servidor para proteger contra ataques de falsificación de solicitudes entre sitios.
   ii) Protección XSS: Escapa y valida correctamente toda la entrada del usuario para proteger contra ataques de secuencias de comandos entre sitios (XSS).

iii) Protección SQL Injection: Utiliza declaraciones preparadas y consultas parametrizadas para proteger contra ataques de inyección SQL.

8) Servidor Web.-

a. Se utilizará un servidor web compatible con Laravel, como AWS, configurado correctamente para soportar el framework y las aplicaciones web. Para más información revisar el manual de instalación.

4. Herramientas utilizadas para el desarrollo:

Dentro de las herramientas utilizadas para el desarrollo de este sistema web se incluyen las siguientes:

- **Entorno de Desarrollo Integrado (IDE)**
  - Visual Studio Code 1.90.0: Es un IDE que proporciona un entorno integrado para escribir, depurar y ejecutar código.

- **Gestión de Versiones y Control de Código Fuente**
  - Git 2.42.0 (con GitHub): Herramienta para gestionar y controlar el código fuente del proyecto, permitiendo el trabajo colaborativo y el seguimiento de cambios.

- **Frameworks de Desarrollo**
  - Laravel 8, Bootstrap 5: Estos frameworks proporcionan una estructura y conjunto de herramientas para simplificar el desarrollo de aplicaciones web.

- **Gestión de Dependencias**
  - Composer 2.7.2 (para PHP): es una herramienta para gestionar las dependencias del proyecto y administrar bibliotecas y paquetes de terceros.

- **Base de Datos**
  - pgAdmin 4.30 (para PostgreSQL 10.23): Herramientas para gestionar y administrar la base de datos del proyecto.

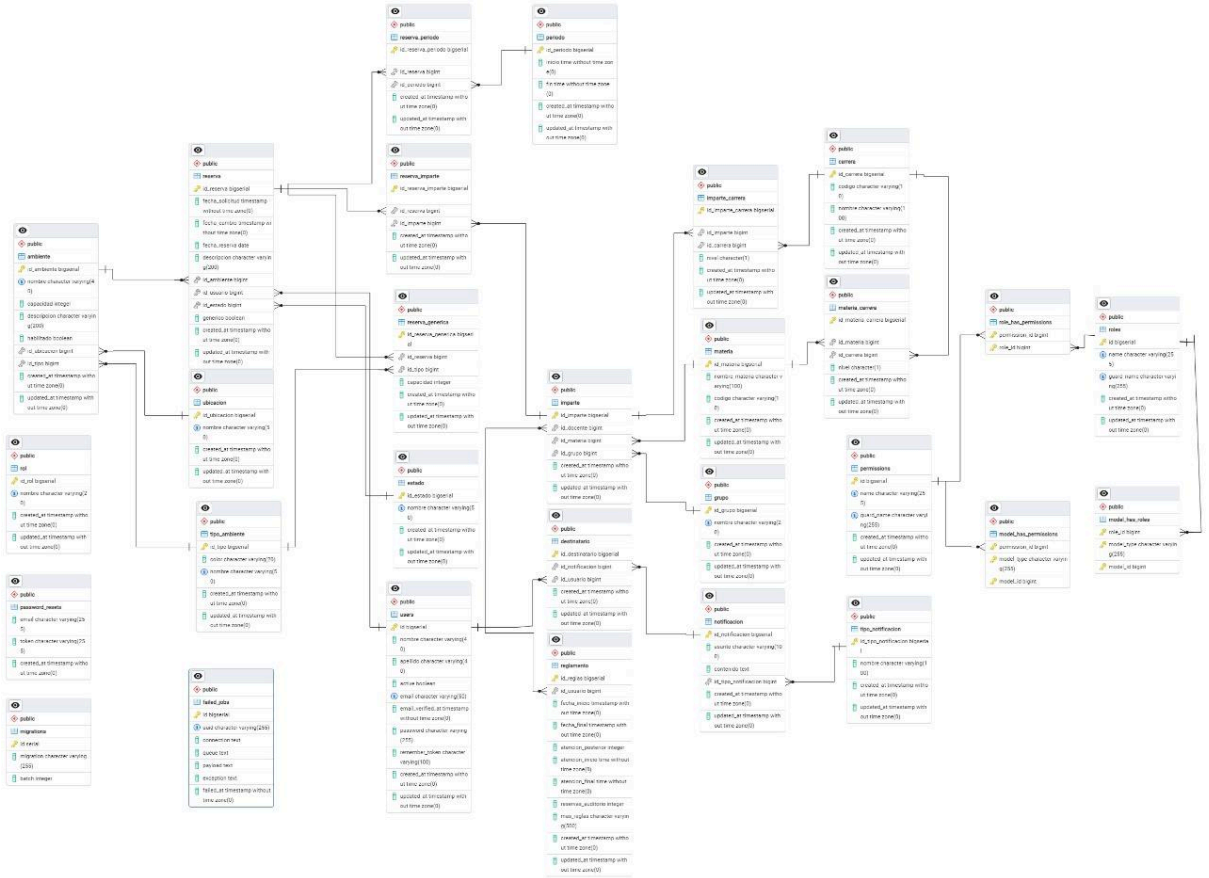
- **Despliegue y Gestión de Servidores**
  - AWS que permite a los desarrolladores implementar aplicaciones web. Para más información visitar el manual de instalación.

- **Documentación y Colaboración:**
  - Google Docs (para colaboración en documentos): Herramienta para documentar el código y facilitar la colaboración entre miembros del equipo.

5. Documentación de diseño

5.1. Diseño de base de datos:

5.1.1. Diagrama de Entidad-Relación (DER)



5.1.2. Análisis de datos para la migración en base de datos

Para este objetivo se diseñó un modelo de E-R (Entidad Relación), que nos permitió hacer el análisis de datos correspondientes a las migraciones que se fueron creando.

5.1.3. Migraciones creadas

Las migraciones creadas son:

2024_03_26_201759_create_ubicacions_table.php
Esta migración en Laravel crea una tabla llamada “ubicacion” en la base de datos, que se utiliza para almacenar ubicaciones de los respectivos ambientes.

- id_ubicacion: Identificador único y clave primaria.
- nombre: Nombre de la ubicación, debe ser único y tiene una longitud máxima de 50 caracteres.
- timestamps: Campos created_at y updated_at para rastrear cuándo se creó y se actualizó cada registro.

2024_03_27_134201_create_tipo_ambientes_table.php
Esta migración en Laravel crea una tabla llamada “tipo_ambiente” en la base de datos, utilizada para almacenar diferentes tipos de ambientes que sean necesarios para cada tipo de ambiente.

- id_tipo: Identificador único y clave primaria.
- color: Color asociado al tipo de ambiente, es opcional y tiene una longitud máxima de 20 caracteres.
- nombre: Nombre del tipo de ambiente, debe ser único y tiene una longitud máxima de 50 caracteres.
- timestamps: Campos created_at y updated_at para rastrear cuándo se creó y se actualizó cada registro.

2024_03_27_140423_create_ambientes_table.php
Esta migración en Laravel crea una tabla llamada “ambiente” en la base de datos, la cual se utiliza para almacenar información sobre diferentes ambientes.

- id_ambiente: Identificador único y clave primaria.
- nombre: Nombre del ambiente, único y con una longitud máxima de 40 caracteres.
- capacidad: Capacidad del ambiente, con restricciones de 10 a 300 (deberían implementarse en el modelo).
- descripcion: Descripción del ambiente, opcional y con una longitud máxima de 200 caracteres.
- habilitado: Estado de habilitación, con un valor predeterminado de true.
- id_ubicacion: Clave foránea que referencia a id_ubicacion en la tabla ubicacion.
- id_tipo: Clave foránea que referencia a id_tipo en la tabla tipo_ambiente.
- timestamps: Campos created_at y updated_at.

2024_03_27_143218_create_rols_table.php
Esta migración crea la tabla rol con los siguientes campos:

- id_rol: Identificador único y clave primaria.
- nombre: Nombre del rol, único y con una longitud máxima de 20 caracteres.
- timestamps: Campos created_at y updated_at.

2024_03_27_152639_create_materias_table.php
Esta migración crea la tabla materia con los siguientes campos:

- id_materia: Identificador único y clave primaria.
- nombre_materia: Nombre de la materia, con una longitud máxima de 100 caracteres.
- codigo: Código de la materia, con una longitud máxima de 10 caracteres.
- timestamps: Campos created_at y updated_at.

2024_03_28_000000_create_users_table.php
La tabla users se utiliza para almacenar información sobre los usuarios del sistema. Esta migración crea la tabla users con los siguientes campos:

- id: Identificador único y clave primaria.
- nombre: Nombre del usuario, con una longitud máxima de 40 caracteres y requerido.
- apellido: Apellido del usuario, con una longitud máxima de 40 caracteres y requerido.
- active: Estado activo del usuario, con un valor predeterminado de 1.
- email: Correo electrónico del usuario, con una longitud máxima de 50 caracteres, único y requerido.
- email_verified_at: Fecha y hora de verificación del correo electrónico, puede ser nulo.
- password: Contraseña del usuario.
- remember_token: Token de "recordar sesión".
- timestamps: Campos created_at y updated_at.

2024_04_05_131950_create_carreras_table.php
Esta migración crea la tabla carrera con los siguientes campos:

- id_carrera: Identificador único y clave primaria.
- codigo: Código de la carrera, con una longitud máxima de 10 caracteres.
- nombre: Nombre de la carrera, con una longitud máxima de 100 caracteres.
- timestamps: Campos created_at y updated_at.

2024_04_05_132454_create_materia_carreras_table.php
Esta migración en Laravel crea una tabla de unión llamada materia_carrera para establecer una relación de muchos a muchos entre las tablas materia y carrera.

- id_materia_carrera: Identificador único y clave primaria.
- id_materia: Clave foránea que referencia a id_materia en la tabla materia, con eliminación en cascada.
- id_carrera: Clave foránea que referencia a id_carrera en la tabla carrera, con eliminación en cascada.
- nivel: Campo de tipo char de longitud 1 para indicar el nivel.
- timestamps: Campos created_at y updated_at.

2024_04_15_221239_create_permission_tables.php
Esta migración crea múltiples tablas para manejar roles y permisos en la aplicación usando el paquete spatie/laravel-permission. Las tablas y sus campos son los siguientes:

- **permissions**
  - id: Identificador único y clave primaria.
  - name: Nombre del permiso.
  - guard_name: Nombre del guard.
  - timestamps: Campos created_at y updated_at.

- **roles**
  - id: Identificador único y clave primaria.
  - team_foreign_key (opcional): Clave foránea para equipos.
  - name: Nombre del rol.
  - guard_name: Nombre del guard.
  - timestamps: Campos created_at y updated_at.

- **model_has_permissions**
  - permission_id: Clave foránea a permissions (eliminación en cascada).
  - model_type: Tipo de modelo.
  - model_id: Identificador del modelo.
  - team_foreign_key (opcional): Clave foránea para equipos.

- **model_has_roles**
  - role_id: Clave foránea a roles (eliminación en cascada).
  - model_type: Tipo de modelo.
  - model_id: Identificador del modelo.
  - team_foreign_key (opcional): Clave foránea para equipos.

- **role_has_permissions**
  - permission_id: Clave foránea a permissions (eliminación en cascada).
  - role_id: Clave foránea a roles (eliminación en cascada).

2024_04_22_194055_create_imparte_table.php

Esta migración crea la tabla imparte para gestionar las relaciones entre docentes, materias y grupos

- **imparte**
  - id_imparte: Identificador único y clave primaria.
  - id_docente: Clave foránea que referencia al docente en la tabla users.
  - id_materia: Clave foránea que referencia a la materia en la tabla materia.
  - id_grupo: Clave foránea que referencia al grupo en la tabla grupo.
  - timestamps: Campos created_at y updated_at.
- **Relaciones**
  - id_docente referencia la tabla users con eliminación en cascada.
  - id_materia referencia la tabla materia con eliminación en cascada.
  - id_grupo referencia la tabla grupo con eliminación en cascada.

2024_05_10_201007_create_estado_table.php

Esta migración crea la tabla estado para almacenar estados o estatus con un identificador único y un nombre.

- id_estado: Identificador único y clave primaria.
- nombre: Nombre del estado, con una longitud máxima de 50 caracteres y debe ser único.
- timestamps: Campos created_at y updated_at.

2024_05_11_200818_create_Periodo_table.php
Esta migración crea la tabla periodo para almacenar períodos de tiempo con un identificador único y los tiempos de inicio y fin.

- id_periodo: Identificador único y clave primaria.
- inicio: Campo de tiempo que representa el inicio del período.
- fin: Campo de tiempo que representa el fin del período.
- timestamps: Campos created_at y updated_at para el seguimiento de la creación y actualización de registros.

2024_05_11_203014_create_reserva_table.php
Esta migración crea la tabla reserva para almacenar información sobre reservas de ambientes, con campos que incluyen fechas, descripción y referencias a otras entidades como usuarios, ambientes y estados.

- id_reserva: Identificador único y clave primaria de la reserva.
- fecha_solicitud: Fecha y hora en que se realiza la solicitud de reserva.
- fecha_cambio: Fecha y hora de la última modificación de la reserva (opcional, puede ser nulo).
- fecha_reserva: Fecha de la reserva.
- descripcion: Descripción opcional de la reserva.
- id_ambiente: Clave externa que referencia al ambiente reservado (nullable, puede no estar asignado).
- id_usuario: Clave externa que referencia al usuario que realiza la reserva.
- id_estado: Clave externa que referencia al estado actual de la reserva.
- generico: Campo booleano que indica si la reserva es genérica o no, con valor predeterminado false.
- timestamps: Campos created_at y updated_at para el seguimiento de la creación y actualización de registros.

2024_05_12_114943_create_reserva_imparte_table.php
Esta migración crea la tabla reserva_imparte, que establece una relación muchos a muchos entre las entidades reserva e imparte. Esta tabla permite asociar múltiples registros de reserva con múltiples registros de imparte.

- Relaciones
    - id_reserva: Clave foránea que referencia a la columna id_reserva en la tabla reserva. Establece una relación de muchos a uno con la tabla reserva, indicando que un registro en reserva_imparte pertenece a una reserva específica.
    - id_imparte: Clave foránea que referencia a la columna id_imparte en la tabla imparte. Establece una relación de muchos a uno con la tabla imparte, indicando que un registro en reserva_imparte está asociado a una relación de impartición específica.

2024_05_15_162542_create_reserva_periodo_table.php
Esta migración crea la tabla reserva_periodo, que establece una relación muchos a muchos entre las entidades reserva y periodo. Esta tabla permite asociar múltiples registros de reserva con múltiples registros de periodo, indicando los períodos de tiempo en los que se realizan las reservas.

- **Relaciones**
    - id_reserva: Clave foránea que referencia a la columna id_reserva en la tabla reserva. Establece una relación de muchos a uno con la tabla reserva, indicando que un registro en reserva_periodo pertenece a una reserva específica.
    - id_periodo: Clave foránea que referencia a la columna id_periodo en la tabla periodo. Establece una relación de muchos a uno con la tabla periodo, indicando que un registro en reserva_periodo está asociado a un período de tiempo específico.

2024_05_16_190032_create_imparte_carrera_table.php
Esta migración crea la tabla imparte_carrera, la cual establece una relación muchos a muchos entre las entidades imparte y carrera. Esta tabla permite asociar múltiples registros de imparte con múltiples registros de carrera, indicando los niveles de enseñanza en los que los docentes imparten materias dentro de una carrera específica.

- Relaciones
    - id_imparte: Clave foránea que referencia a la columna id_imparte en la tabla imparte. Establece una relación de muchos a uno con la tabla imparte, indicando que un registro en imparte_carrera pertenece a una enseñanza específica impartida por un docente.
    - id_carrera: Clave foránea que referencia a la columna id_carrera en la tabla carrera. Establece una relación de muchos a uno con la tabla carrera, indicando que un registro en imparte_carrera está asociado a una carrera específica.

2024_05_30_090938_create_tipo_notificacions_table.php
Esta migración crea la tabla tipo_notificacion con el objetivo de almacenar tipos de notificaciones disponibles en el sistema.

- id_tipo_notificacion: Clave primaria autoincremental de tipo entero grande (bigIncrements).
- nombre: Campo de tipo cadena de caracteres (string) con una longitud máxima de 100 caracteres. Almacena el nombre del tipo de notificación.
- timestamps: Dos campos automáticos para registrar la fecha y hora de creación (created_at) y la última fecha y hora de actualización (updated_at) de cada registro en la tabla.

2024_05_30_091501_create_notificacions_table.php
La tabla notificación está diseñada para almacenar y gestionar las notificaciones del sistema. Cada registro de esta tabla representa una notificación específica enviada a los usuarios.

- id_notificacion: Campo de tipo bigIncrements, que representa el identificador único y clave primaria de cada notificación.
- asunto: Campo de tipo string con una longitud máxima de 100 caracteres. Almacena el asunto o título de la notificación.
- contenido: Campo de tipo text, diseñado para almacenar el contenido extenso de la notificación.
- id_tipo_notificacion: Clave externa (unsignedBigInteger) que referencia el identificador único en la tabla tipo_notificacion.
- timestamps: Campos automáticos created_at y updated_at para registrar la fecha y hora de creación y actualización de cada registro en la tabla.

2024_05_30_095313_create_destinatarios_table.php
Esta migración crea la tabla destinatario, que sirve para registrar la relación entre las notificaciones y los usuarios que las reciben.

- Relaciones:
    - id_notificacion: Clave foránea que establece una relación con la tabla notificacion, indicando a qué notificación pertenece este destinatario.
    - id_usuario: Clave foránea que establece una relación con la tabla users, indicando qué usuario recibe esta notificación.

2024_05_31_233215_create_reserva_generica_table.php
Esta migración crea la tabla reserva_generica, que permite almacenar reservas genéricas de ambientes.

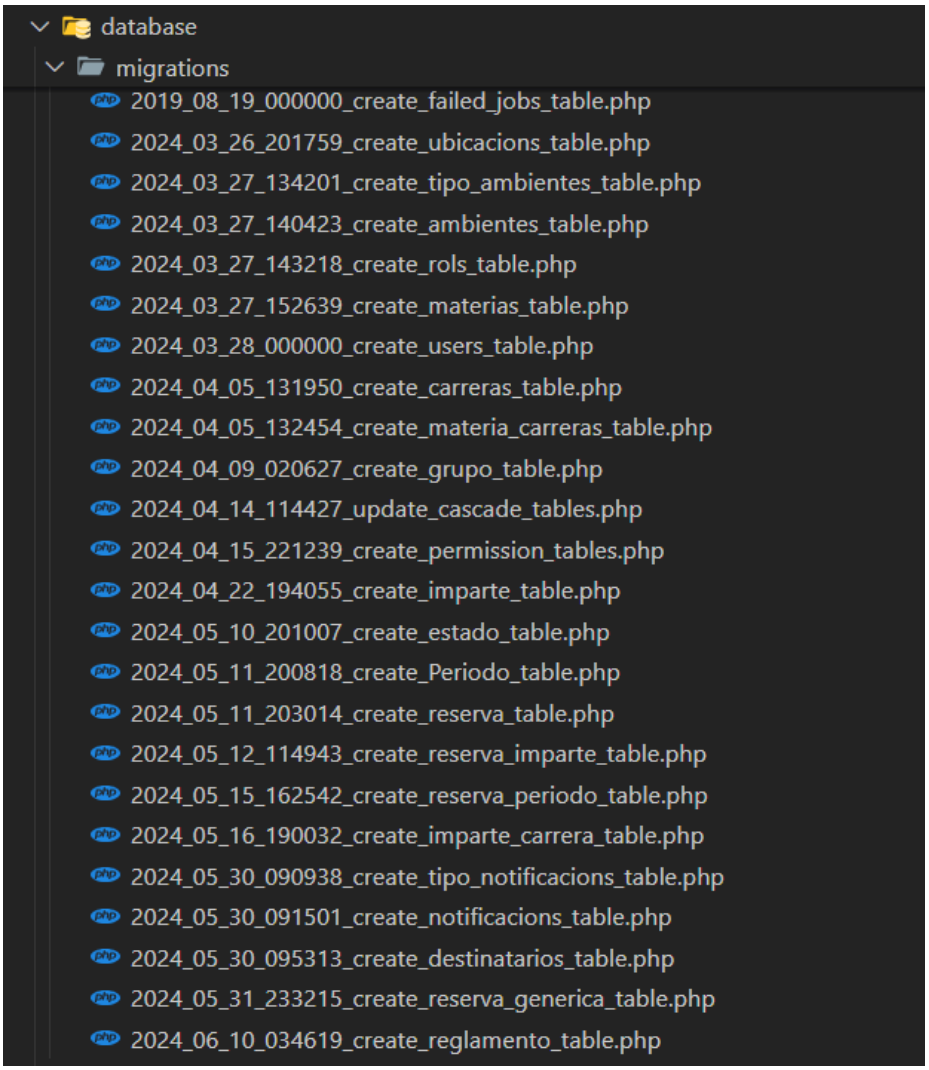
- id_reserva_generica: Identificador único y clave primaria de la tabla.
- id_reserva: Clave externa que referencia al identificador único de una reserva en la tabla reserva.
- id_tipo: Clave externa que referencia al identificador único de un tipo de ambiente en la tabla tipo_ambiente.
- capacidad: Capacidad de personas que puede albergar la reserva genérica.
- timestamps: Campos created_at y updated_at para el seguimiento de la creación y actualización de registros.

2024_06_10_034619_create_reglamento_table.php
Esta migración crea la tabla reglamento, la cual permite almacenar reglas y normativas relacionadas con la gestión de reservas en un sistema.

- id_reglas: Identificador único y clave primaria de la tabla.

- id_usuario: Clave externa que referencia al identificador único de un usuario en la tabla users, indicando el usuario que establece las reglas.
- fecha_inicio: Fecha y hora de inicio de vigencia de las reglas, campo requerido.
- fecha_final: Fecha y hora de fin de vigencia de las reglas, campo requerido.
- atencion_posterior: Número entero mínimo de 0 que indica la atención posterior necesaria para las reservas, campo requerido.
- atencion_inicio: Hora de inicio de atención para las reservas, campo requerido.
- atencion_final: Hora de fin de atención para las reservas, campo requerido.
- reservas_auditorio: Número entero mínimo de 0 que indica la cantidad de reservas permitidas para el auditorio, campo requerido.
- mas_reglas: Texto opcional que permite agregar más detalles o reglas adicionales (máximo 500 caracteres).

Las migraciones están en la carpeta de database, migrations



5.1.4. Arquitectura de sistema Reserva Fácil

Está diseñado siguiendo una arquitectura de tres capas, basada en el patrón Modelo-Vista-Controlador (MVC). Esta elección arquitectónica proporciona modularidad, flexibilidad y facilita el mantenimiento del sistema a medida que evoluciona.

5.1.4.1. Componentes Principales:

Capa de Presentación (Vista):

- Responsable de la interfaz de usuario y la interacción con el usuario.
- Desarrollada utilizando las plantillas Blade de Laravel y estilos de Bootstrap para una apariencia moderna y responsiva.

Capa de Lógica de Negocios (Controlador):

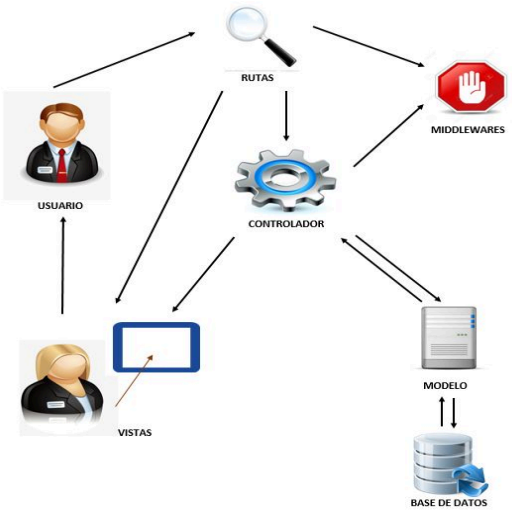
- Contiene la lógica de aplicación y la gestión de datos.
- Implementada en Laravel, un framework PHP que sigue el patrón MVC.
- Utiliza controladores para procesar las solicitudes del usuario y gestionar la lógica de negocio.

Capa de Datos (Modelo):

- Responsable de la interacción con la base de datos.
- Utiliza el ORM Eloquent de Laravel para simplificar las operaciones de base de datos.
- Define modelos que representan las entidades del sistema, como Competencias y Participantes.

5.1.4.2. Cuadro de estructura

La siguiente imagen explica la interacción de los componentes y el funcionamiento del Framework.



5.1.4.3. El proyecto presenta la siguiente estructura de folders y archivos:

- app/
- bootstrap/
- config/
- database/
- public/
- resources/
- routes/
- scss/
- storage/
- stubs/
- tests/
- vendor/
- .editorconfig
- .env
- .env.example
- .gitattributes
- .gitignore
- .styleci.yml
- artisan
- composer.json
- composer.lock
- package-lock.json
- package.json
- phpunit.xml
- README.md
- server.php
- .webpack.mix.js

6. Configuración

El sistema ya está configurado con valores predeterminados que son adecuados para la mayoría de los casos de uso. Por lo tanto, no es necesario realizar ningún ajuste adicional, ya que el sistema viene preconfigurado y listo para su uso inmediato.

7. Análisis

Es importante realizar un análisis exhaustivo para garantizar su funcionamiento óptimo y verificar que cumple con las necesidades y expectativas del usuario. Sigue estos pasos para llevar a cabo el análisis:

1. **Pruebas de Funcionalidad:**

   Realiza pruebas exhaustivas en todas las funciones y características del sistema para verificar que funcionen según lo esperado.

   Asegúrate de probar tanto las funciones básicas como las avanzadas del sistema para detectar posibles errores o problemas de funcionamiento.

2. **Pruebas de Rendimiento:**

   Realiza pruebas de rendimiento para evaluar la velocidad y la eficiencia del sistema bajo diferentes cargas de trabajo.

   Verifica que el sistema pueda manejar múltiples solicitudes simultáneas sin experimentar retrasos o caídas en el rendimiento.

3. **Pruebas de Seguridad:**

   Realiza pruebas de seguridad para identificar posibles vulnerabilidades y asegurarte de que el sistema esté protegido contra amenazas externas.

   Verifica que todas las medidas de seguridad implementadas, como SSL/TLS y autenticación adecuada, estén funcionando correctamente.

4. **Pruebas de Usabilidad:**

   Solicita retroalimentación de los usuarios beta o realiza pruebas de usabilidad para evaluar la facilidad de uso y la experiencia del usuario.

   Identifica cualquier área del sistema que pueda ser confusa o difícil de navegar y realiza los ajustes necesarios para mejorar la usabilidad.

5. **Análisis de Resultados:**

   Analiza los resultados de las pruebas realizadas y documenta cualquier problema o área de mejora identificada.

   Utiliza los resultados del análisis para realizar ajustes adicionales en la configuración del sistema y mejorar su rendimiento y usabilidad.

Una vez completado el análisis y realizado cualquier ajuste necesario, el sistema estará listo para ser desplegado en el servidor y puesto en funcionamiento de manera oficial.

8. Desarrollo

El código fuente del sistema se encuentra alojado en un repositorio Git en GitHub. Este repositorio actúa como el almacenamiento centralizado de todo el código del proyecto, permitiendo a los desarrolladores colaborar de manera efectiva, realizar un seguimiento de los cambios realizados a lo largo del tiempo y mantener una versión estable del software.

- Código fuente: https://github.com/DarCkDev/reserva-facil

El despliegue del sistema está configurado en AWS (Amazon Web Services), una plataforma de servicios en la nube que ofrece una amplia gama de servicios para ayudar a las organizaciones a escalar y crecer. AWS es ampliamente utilizado en el ámbito del desarrollo de software debido a su confiabilidad, escalabilidad y flexibilidad.

- Despliegue sistema: http://3.134.115.18/

9. Privilegios “Permisos”

El sistema está diseñado con varios tipos de privilegios de usuarios para satisfacer diferentes roles y responsabilidades dentro de la plataforma. Incluye roles predefinidos como administrador y docente, así como la flexibilidad para crear tipos de usuarios personalizados según necesidades específicas.

- Administrador: Tiene acceso completo y privilegios para administrar todas las funciones del sistema, incluyendo la gestión de usuarios, configuraciones, acceso a todas las funcionalidades críticas, etc.

- Docente: Este rol está diseñado para docentes de la universidad mayor de san simón con privilegios que le permiten desde solicitar reservas hasta listar sus solicitudes entre otras.

- Usuarios personalizados: Además de los roles predefinidos, el sistema permite la creación de tipos de usuarios personalizados. Esto significa que las organizaciones pueden definir roles específicos con permisos y accesos únicos que se alinean con sus procesos internos y necesidades particulares. Por ejemplo, podrían crear roles como "Coordinador de Programas", "Asistente Administrativo", o cualquier otro perfil que requiere un conjunto específico de privilegios y responsabilidades dentro del sistema.

Este enfoque modular y flexible no solo facilita la administración de usuarios en el sistema, sino que también permite una personalización precisa de los niveles de acceso y control. Cada tipo de usuario puede tener atributos y permisos únicos que reflejen su función y contribución dentro del entorno educativo o administrativo en el que se implementa el sistema.

Palmer Technology
palmertechn@gmail.com
Cochabamba – Bolivia

10. Soporte

Cuando ocurre algún problema con el sistema en el cliente, es importante proporcionar un soporte eficiente y efectivo para resolver la situación lo más rápido posible. Para ello, vamos a utilizar una combinación de herramientas de colaboración remota y de gestión de problemas.

1. Elementos que conforman el soporte al cliente:

El apoyo al cliente comprende varios elementos complementarios, esta elección varía según la organización, su tamaño, su sector y objetivos.

A continuación, se detallan algunas herramientas que vamos a emplear:

- **Llamada telefónica:**
  Esta solución permite a los clientes comunicarse directamente con nuestros agentes de soporte al cliente. Los cuales son capaces de resolver sus problemas y dar respuestas satisfactorias.

  Si el cliente tiene problemas básicos, el agente del centro le redirige a la herramienta de autoayuda de la empresa. Sin embargo, puede ocurrir que el problema no pueda ser resuelto por la herramienta de autoservicio. En este caso, el asesor debe dar una respuesta personalizada al cliente.

- **TeamViewer:**
  Esta herramienta permite el acceso remoto al sistema del cliente para diagnosticar y solucionar problemas directamente desde nuestro equipo. Podemos ver la pantalla del cliente en tiempo real y realizar acciones para solucionar problemas de manera rápida y eficiente.

- **Google meet:**
  Para casos en los que sea necesario realizar una reunión virtual con el cliente para discutir el problema y posibles soluciones, Zoom es una herramienta útil. Permite realizar videollamadas, compartir pantalla y colaborar en tiempo real.

- **GitHub/GitLab:**
  Si el problema está relacionado con el código fuente del sistema, vamos a utilizar plataformas de alojamiento de código como GitHub o GitLab para colaborar en la resolución del problema. Los desarrolladores pueden crear ramas específicas, realizar cambios y enviar solicitudes de extracción para revisión y fusión.

- **Google Drive:**
  Para compartir documentos, manuales de usuario, registros de errores y otros archivos relevantes con el cliente y el equipo de soporte técnico, Google Drive es una opción conveniente. Permite compartir archivos de forma segura y colaborar en tiempo real en la creación y edición de documentos.

Estas herramientas de colaboración remota nos permiten proporcionar un soporte rápido y eficiente al cliente, resolver problemas de manera efectiva y mantener una comunicación transparente y fluida con todas las partes involucradas en la resolución del problema.

11. Mantenimiento

El objetivo fundamental del mantenimiento en el sistema de reserva de ambientes o aulas “Reserva fácil” es prevenir fallos para que el sistema se pueda usar el mayor tiempo posible.

Sin embargo, a veces no se consigue y acaban por terminar con fallos. En ese caso, las labores de mantenimiento estarán destinadas a reparar estos fallos o errores.

Por supuesto, es fundamental considerar y cuidar los criterios de mantenimiento tanto cuando el sistema está en producción como cuando se está manejando de forma local.

1. Mantenimiento en entorno local

a. **Respaldo Antes de Modificaciones:**
   Antes de realizar cualquier modificación en el código o la configuración del sistema en el entorno local, se debe realizar un respaldo completo de todos los archivos y datos pertinentes. Esto garantiza que uno pueda revertir cualquier cambio que cause problemas sin perder datos importantes.

b. **Pruebas Exhaustivas:**
   Realizar pruebas exhaustivas de todas las modificaciones y mejoras en el entorno local para identificar y corregir errores antes de implementar los cambios en el entorno de producción. Esto ayuda a minimizar el riesgo de interrupciones o problemas inesperados cuando se despliegan los cambios.

c. **Optimización y Mejoras Continuas:**
   Utiliza el entorno local como un espacio para experimentar y probar nuevas funcionalidades, mejoras de rendimiento y optimizaciones del sistema. Esto te permite iterar rápidamente y validar los cambios antes de implementarlos en producción.

d. **Documentación Detallada:**
Mantén una documentación detallada de todas las modificaciones realizadas en el entorno local, incluyendo los cambios en el código, la configuración del sistema y cualquier problema encontrado y solucionado durante el proceso de desarrollo y pruebas.

2. Mantenimiento en entorno de producción:

a. **Respaldos Regulares:**
Se establecerá un programa regular de respaldo de todos los datos críticos del sistema en el entorno de producción, incluyendo la base de datos, los archivos del sistema y cualquier otro dato relevante. Los respaldos periódicos garantizan que puedas restaurar el sistema en caso de pérdida de datos o problemas graves.

b. **Monitorización Continua:**
Implementar herramientas de monitorización continua para supervisar el rendimiento del sistema, la disponibilidad y la seguridad en el entorno de producción. Esto permite identificar y abordar cualquier problema o anomalía de manera proactiva antes de que afecte a los usuarios finales.

c. **Parches y Actualizaciones de Seguridad:**
Mantener el sistema actualizado con las últimas versiones de software y parches de seguridad para mitigar los riesgos de vulnerabilidades y ataques cibernéticos en el entorno de producción.

d. **Gestión de Incidentes:**
Establece un proceso formal de gestión de incidentes para registrar, priorizar y resolver problemas de manera eficiente en el entorno de producción. Esto incluye la comunicación con los usuarios afectados, la asignación de recursos adecuados y la documentación detallada de las acciones tomadas para resolver el problema.

e. **Plan de Continuidad del Negocio:**
Desarrolla un plan de continuidad del negocio que incluya medidas para responder a situaciones de emergencia, como fallas del sistema, ataques cibernéticos o desastres naturales. Esto garantiza que el negocio pueda continuar operando con el menor tiempo de inactividad posible en caso de un evento catastrófico.

12. Información de Soporte

Palmer Technology S.R.L. Grupo Empresa
Av. Melchor Pérez de Olguín, entre la calle E. Camargo y la Av. Simón López
Cochabamba, Bolivia
Teléfonos: (+591) 67681761
Pagina Web: www.palmertechnology.com
Correo de Soporte: palmertechn@gmail.com