|  |
| --- |
| cifp Juan de colonia |
| Sistemas de Gestión Empresarial |
| Memorias Proyecto Final Hospital |
|  |
| **Roberto Tiron & Alexis López Briongos Dam2t** |
| **23/02/2024** |

|  |
| --- |
|  |

*Índice*

[1. Introducción 2](#_Toc159631745)

[2. Contexto del Hospital 2](#_Toc159631746)

[3. Diseño del módulo 3](#_Toc159631747)

[4. Funcionalidades 8](#_Toc159631748)

[5. Modelo de datos 13](#_Toc159631749)

[6. Vistas y Formularios 20](#_Toc159631750)

[7. Proceso de desarrollo 24](#_Toc159631751)

[8. Conclusiones 25](#_Toc159631752)

# Introducción

* El proyecto consiste en el desarrollo de un módulo para Odoo que se enfoca en la gestión integral de un hospital o ambulatorio. Este módulo tiene como objetivo optimizar los procesos administrativos y clínicos dentro de la institución médica, permitiendo una gestión eficiente de los recursos y una atención de calidad para los pacientes.

Objetivos:

* **Mejorar la eficiencia administrativa**: Automatizando tareas como la gestión de citas médicas, registro de pacientes, asignación de camas, entre otros, se busca agilizar los procesos administrativos del hospital.
* **Facilitar la atención médica**: Proporcionando a los profesionales de la salud herramientas para acceder de manera rápida y precisa a la información de los pacientes, historiales médicos, diagnósticos, tratamientos, etc., se pretende mejorar la calidad y la rapidez en la atención médica.
* **Centralizar la información**: Integrando todos los aspectos de la gestión hospitalaria en un único sistema, se busca centralizar la información y evitar la duplicación de datos, lo que contribuye a una toma de decisiones más informada y a una mejor coordinación entre los diferentes departamentos del hospital.

# Contexto del Hospital

* El módulo se está desarrollando para su implementación en un entorno hospitalario moderno y dinámico.
* Este hospital puede variar en tamaño, desde pequeños ambulatorios hasta grandes centros médicos, y busca optimizar sus procesos mediante el uso de tecnología.
* Se espera que el sistema sea escalable y adaptable a las necesidades específicas de cada institución, permitiendo una fácil configuración y personalización según los requerimientos del hospital.

# Diseño del módulo

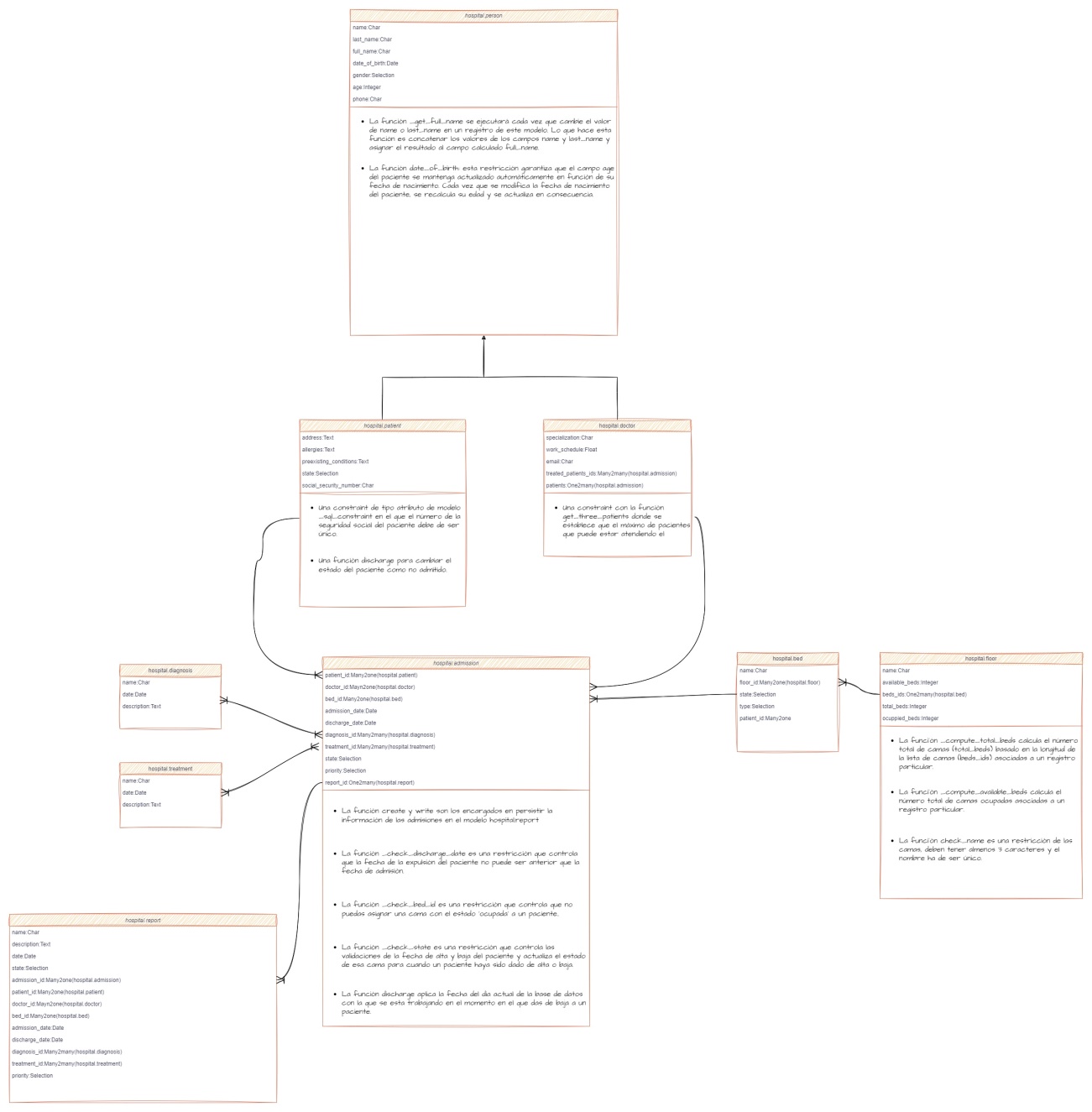
* Este módulo se compone de 6 csv, 9 modelos y 10 vistas:

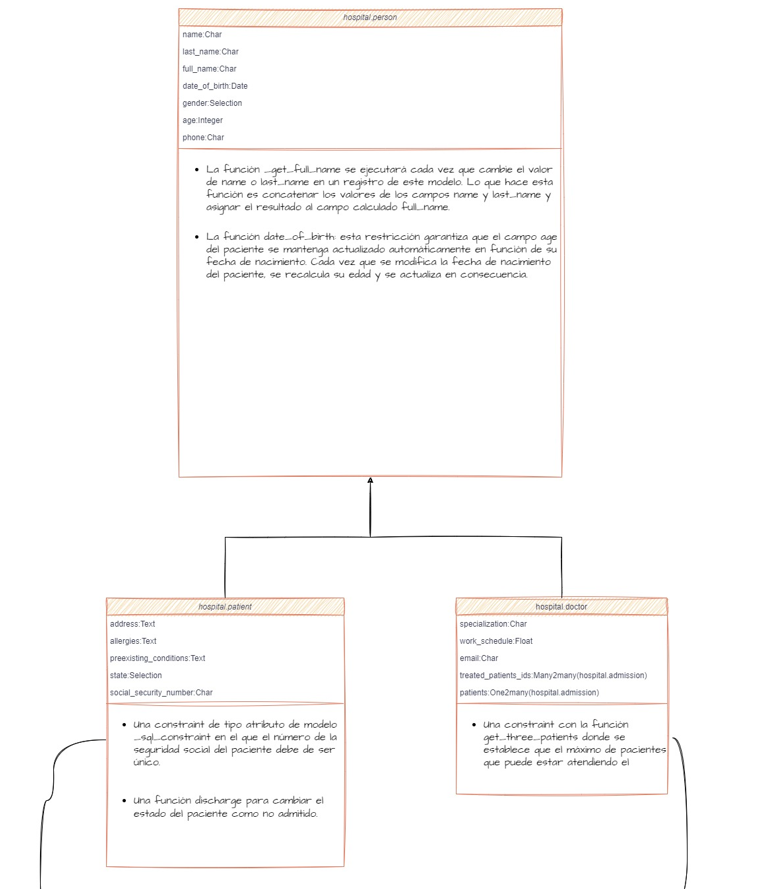
Estructura del proyecto

Hospital

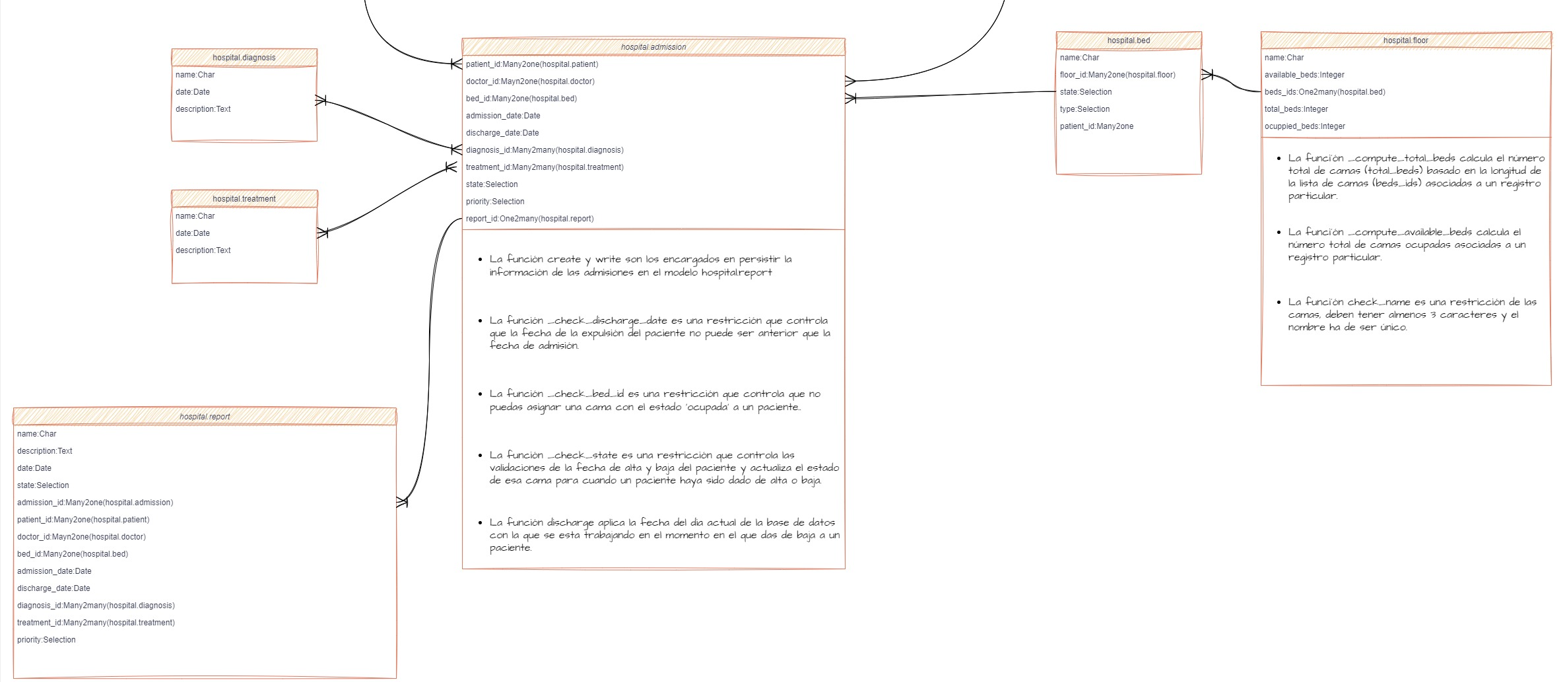
* controllers
  + \_init\_.py
  + controllers.py
* data
  + hospital.bed.csv
  + hospital.diagnosis.csv
  + hospital.doctor.csv
  + hospital.floor.csv
  + hospital.patient.csv
  + hospital.treatment.csv
* demo
  + demo.xml
* models
  + \_init\_.py
  + Admission.py
  + Bed.py
  + Diagnosis.py
  + Doctor.py
  + Floor.py
  + Patient.py
  + Person.py
  + Report.py
  + Treatment.py
* security
  + ir.model.access.csv
* static
  + description
    - icon.png
  + src
    - scss
      * menus.scss
      * views.scss
* views
  + Admission.xml
  + Bed.xml
  + Diagnosis.xml
  + Doctor.xml
  + Floor.xml
  + Menu.xml
  + Patient.xml
  + Report.xml
  + template.xml
  + Treatment.xml
* \_init\_\_.py
* \_manifest\_.py
* pythonBedsGenerateCSV.py
* pythonDiagnosisGenerateCSV.py
* pythonDoctorsGenerateCSV.py
* pythonFloorGenerateCSV.py
* pythonTreatmentGenerateCSV.py

Diagrama UML

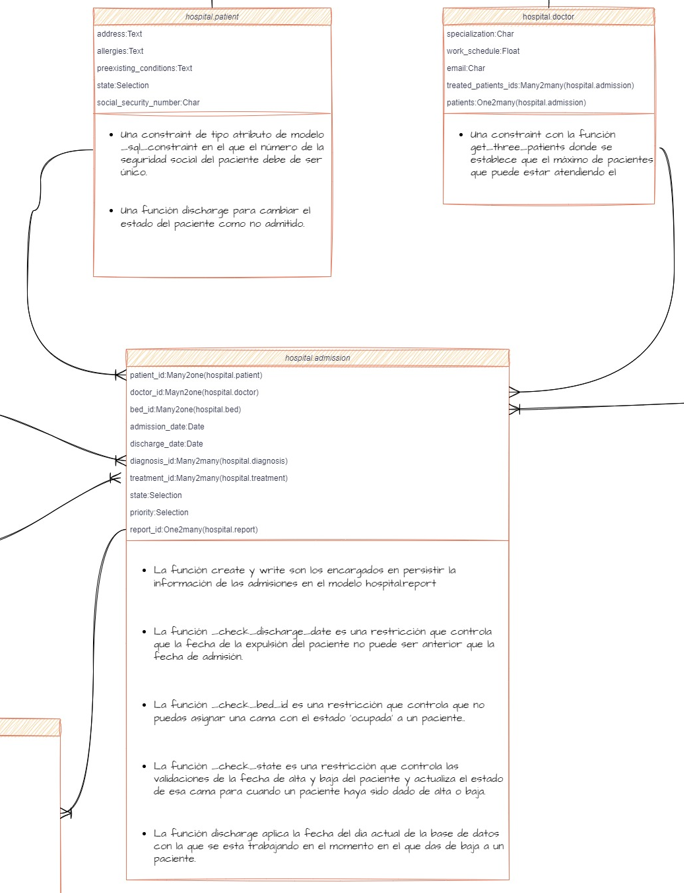




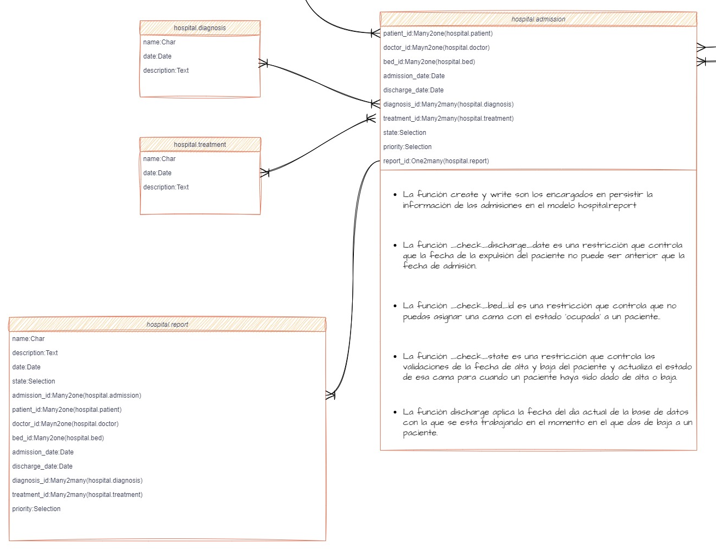
*Los modelos hospital.patient y hospital.doctor heredando de hospital.person.*

**

*El modelo hospital.admission relacionándose con todos los modelos necesarios para poder pasar una consulta y dar de alta a un paciente.*



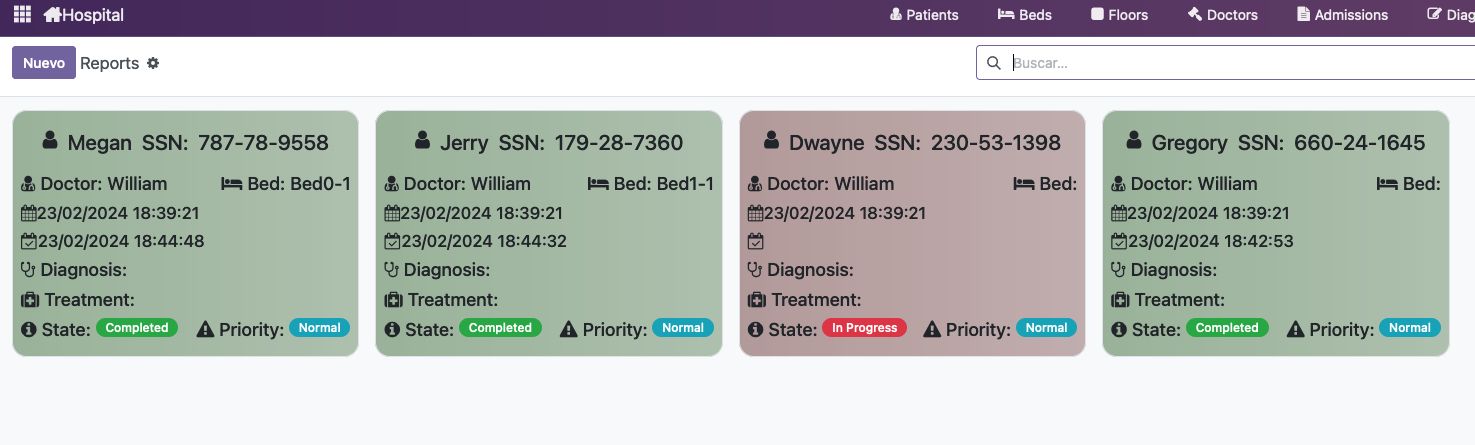
*El modelo hospital.admission relacionado con hospital.patient y hospital.doctor(Many2one)*

**

*Relación entre el modelo hospital.admission y hospital.report para poder persistir los datos del historial de admisiones.*

Diseño

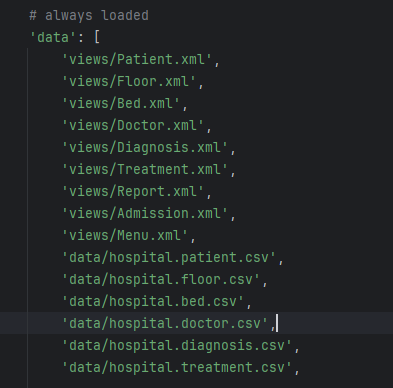
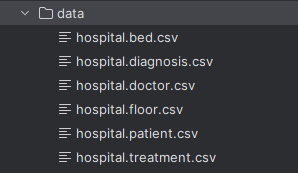
* Todos los modelos de este módulo soportan todas las vistas disponibles en Odoo para mostrar las tablas de la base de datos, pero en este caso nos hemos centrado en desarrollar una interfaz atractiva e intuitiva para el usuario basado en la vista Kanban

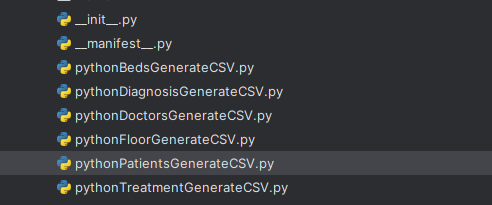


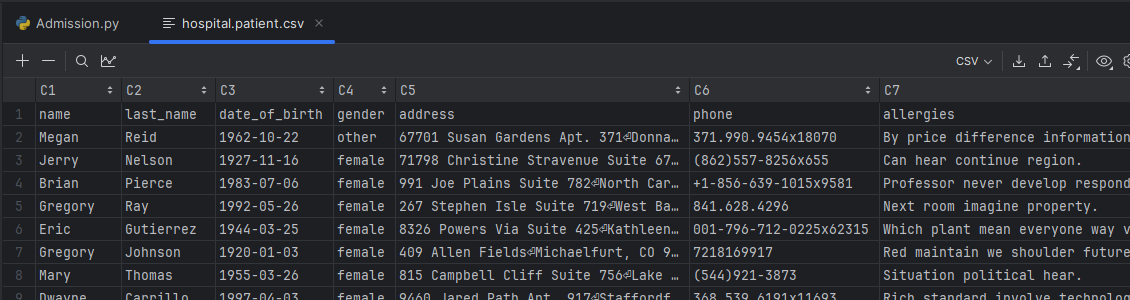
*Ejemplo de la vista kanban del modelo hospital.report*

## Funcionalidades

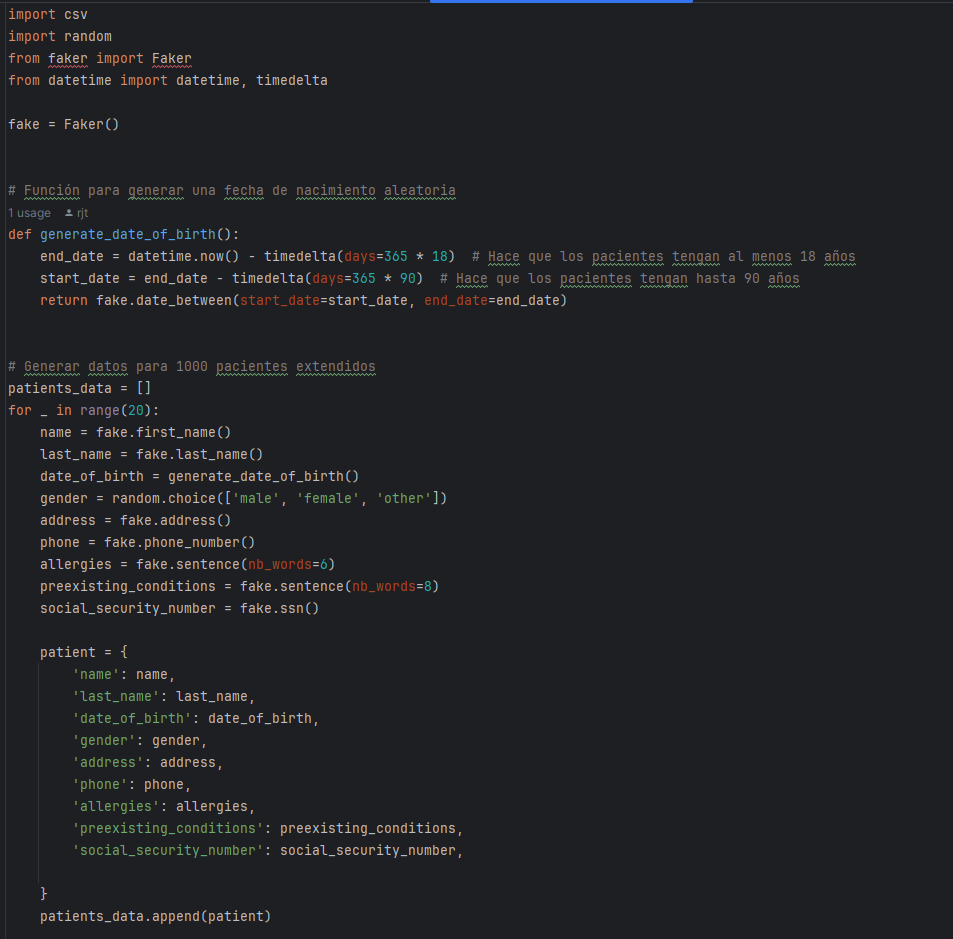
* Este módulo está organizado por áreas o módulos relacionados (por ejemplo, gestión de pacientes, citas médicas, historial médico, etc.).





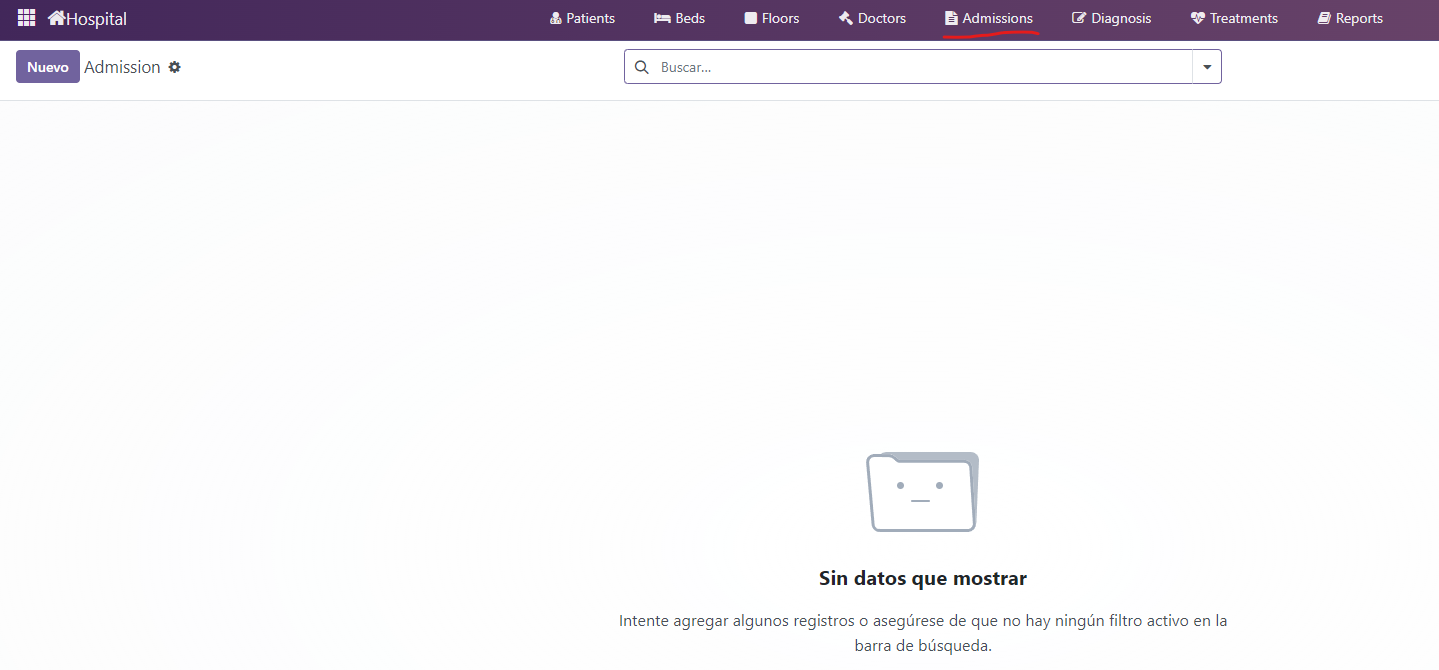


* Para obtener los datos de los registros utilizaremos documentos csv que deberán de estar denominados que igual que los modelos y que se cargarán cada vez que se ejecute el módulo añadiéndolos en el \_manifest\_.py.
* También hemos realizado la creación de una clase específica para generar los valores de los registros de cada modelo.

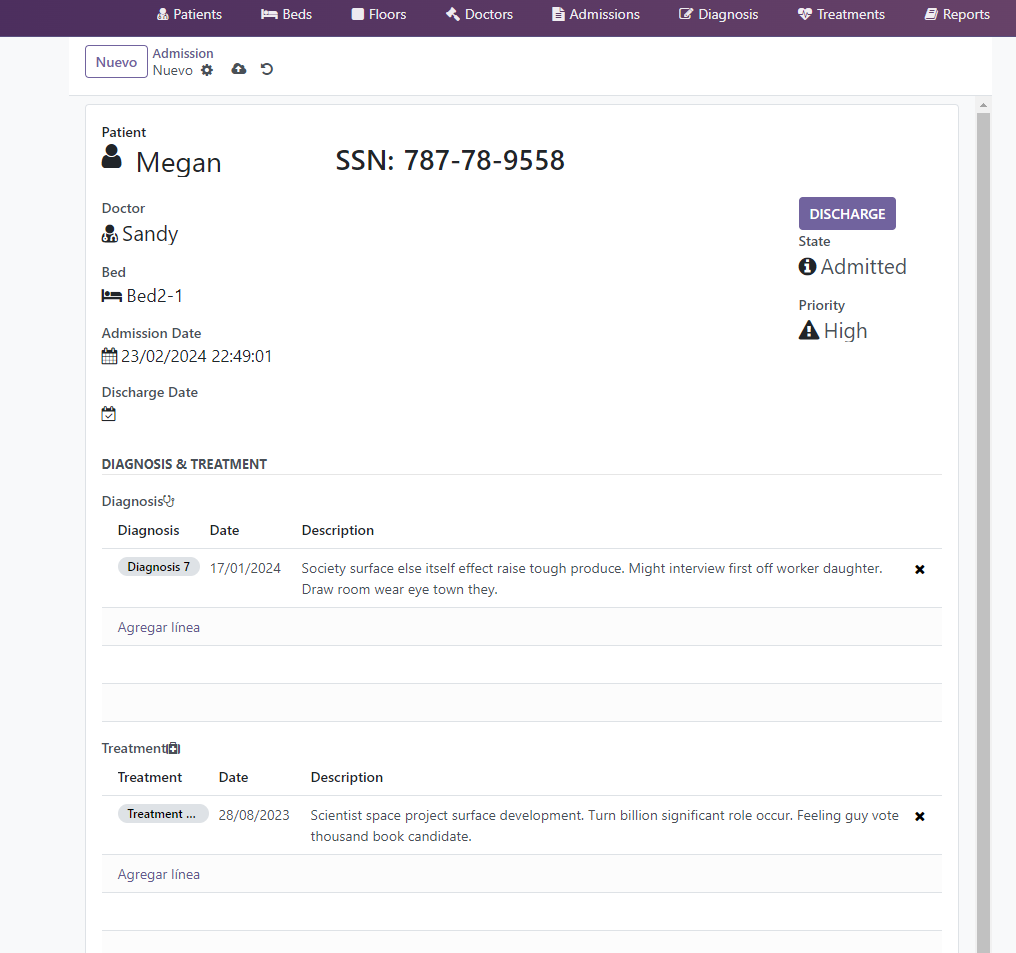


*Clase encargada de generar valores para los registros del hospital.patient.csv*

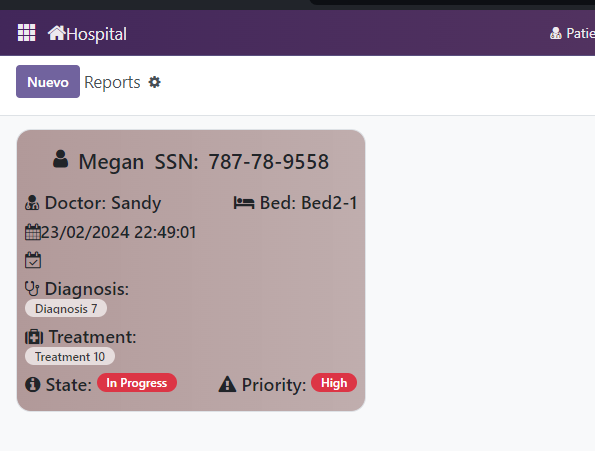
* Para poder generar los datos del csv utilizaremos la librería Faker.
* Esta librería tiene métodos para poder generar datos aleatorios pero específicos como pueden ser fechas, direcciones o seleccionar entre los valores que tú le indiques.



* Después de tener registros en todos los modelos anteriormente mencionados, la funcionalidad principal de este modelo es crear una solicitud de admisión para un paciente en el centro sanitario.



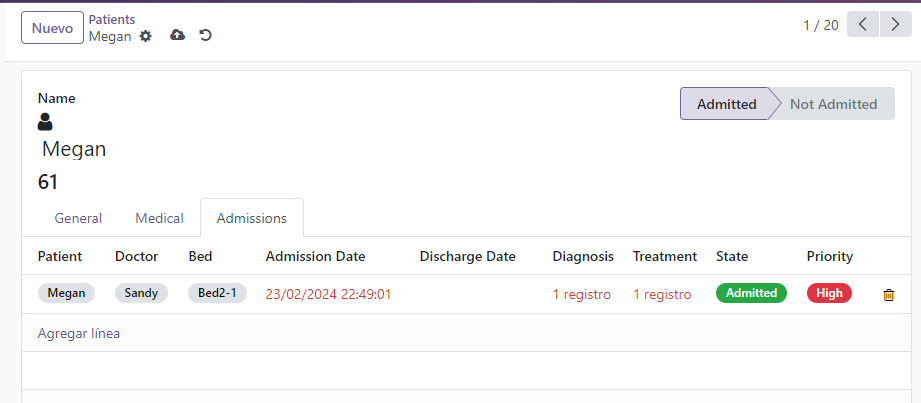
* Aquí podemos visualizar todos los campos necesarios como el doctor que lo atiende, la fecha, la prioridad, etc.
* Para realizar la gestión de una admisión de un nuevo paciente al centro sanitario.



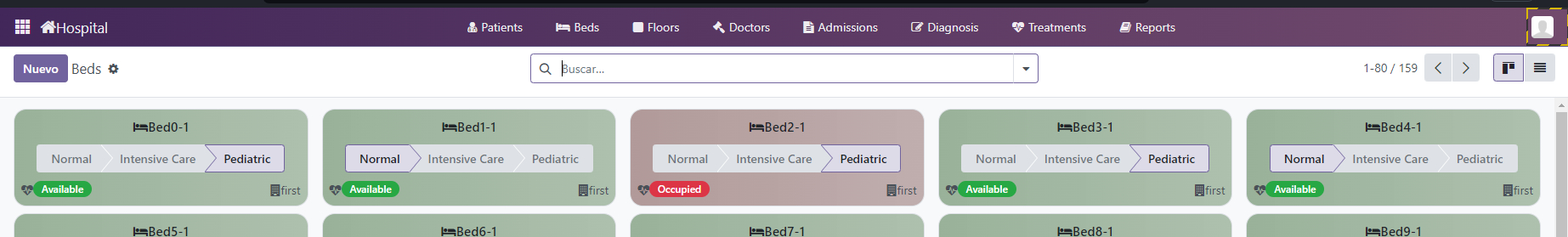
* Después en el modelo de hospital.report podemos visualizar las consultas de los pacientes.



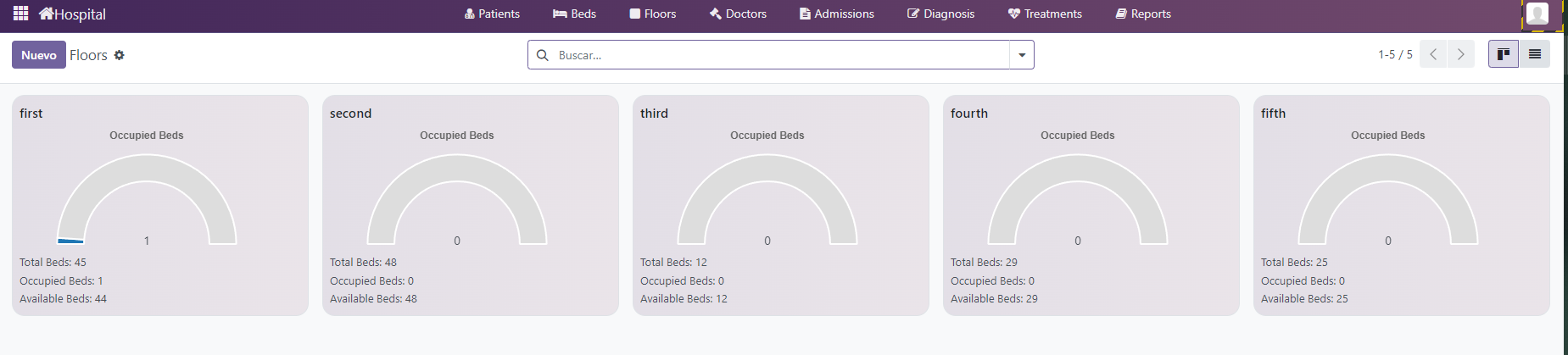
* En el modelo de hospital.doctor podemos visualizar los doctores todos los con pacientes activos.



* También podemos visualizar las admisiones activas en el modelo hospital.patient.



* En el modelo hospital.bed podemos ver que camas están ocupadas.

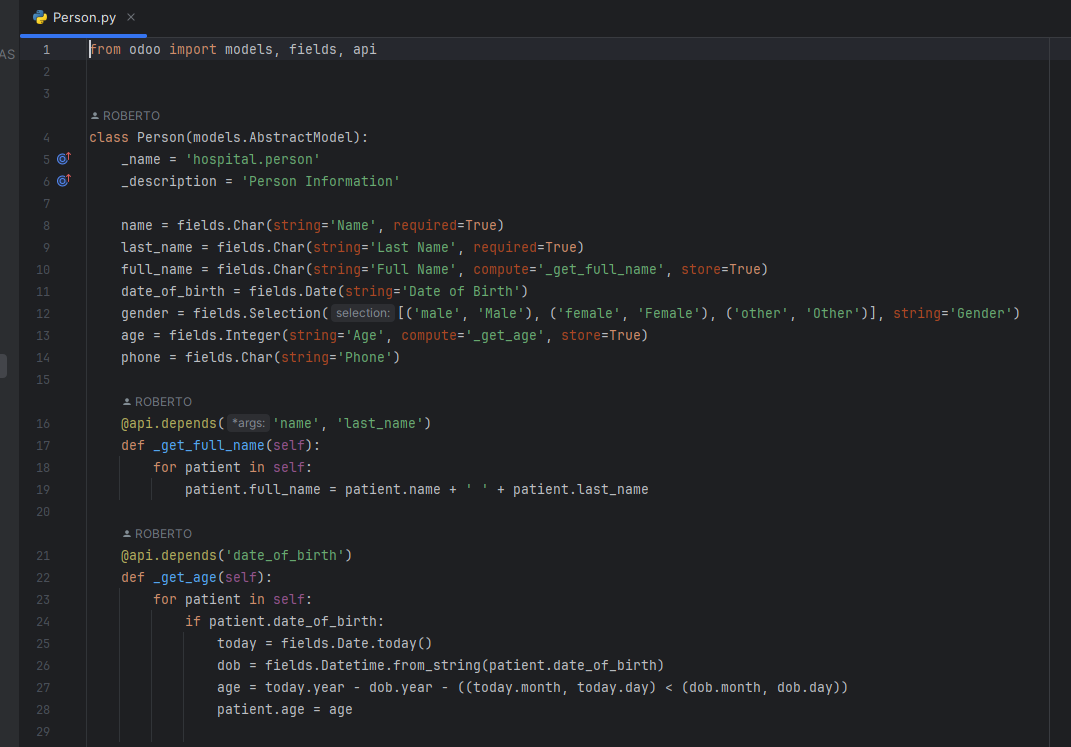


* En el modelo hospital.floor podemos visualizar todas las plantas y las capas ocupadas en cada planta.

## Modelo de datos

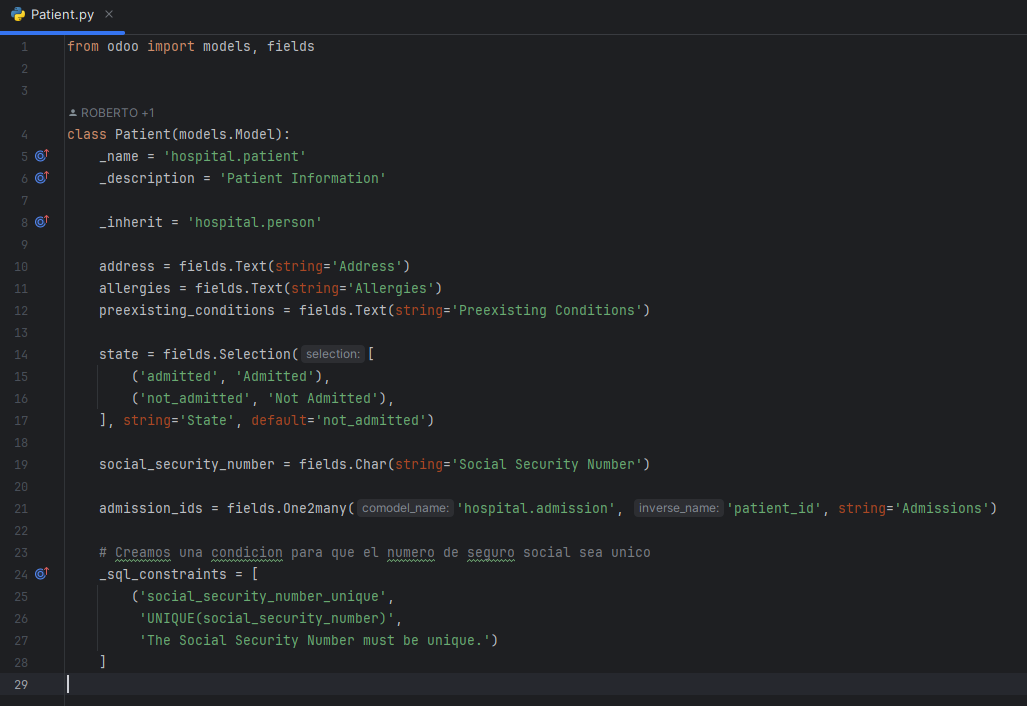
* El módulo se compone de 9 modelos que iremos comentando a continuación.

hospital.person



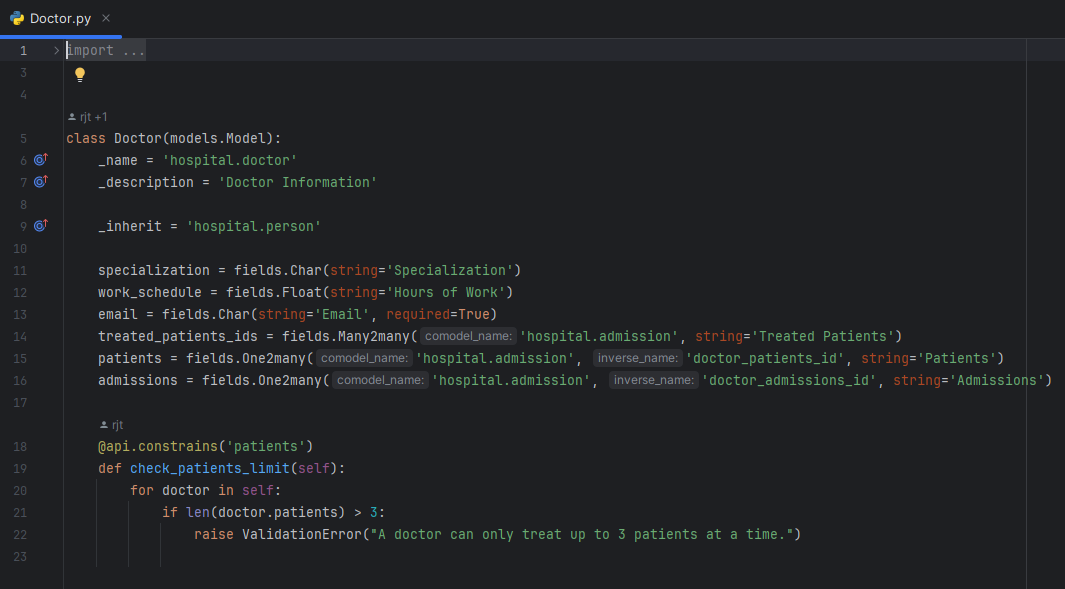
* La clase hospital.person proporciona una estructura básica para almacenar información sobre personas en un entorno hospitalario, incluyendo campos para atributos clave y métodos para calcular valores derivados.
* Las clases que heredan de esta clase pueden extenderla según sea necesario para adaptarse a requisitos específicos, como pacientes, médicos, personal administrativo, etc.

hospital.patient



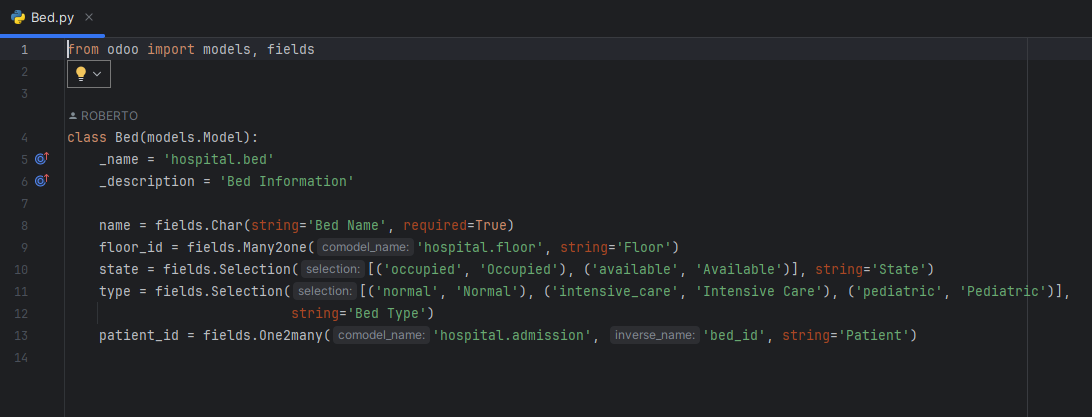
* La clase Patient extiende la funcionalidad de la clase hospital.person para representar específicamente a los pacientes en un sistema hospitalario, agregando campos adicionales y estableciendo relaciones relevantes para gestionar la información de los pacientes de manera efectiva.

hospital.doctor



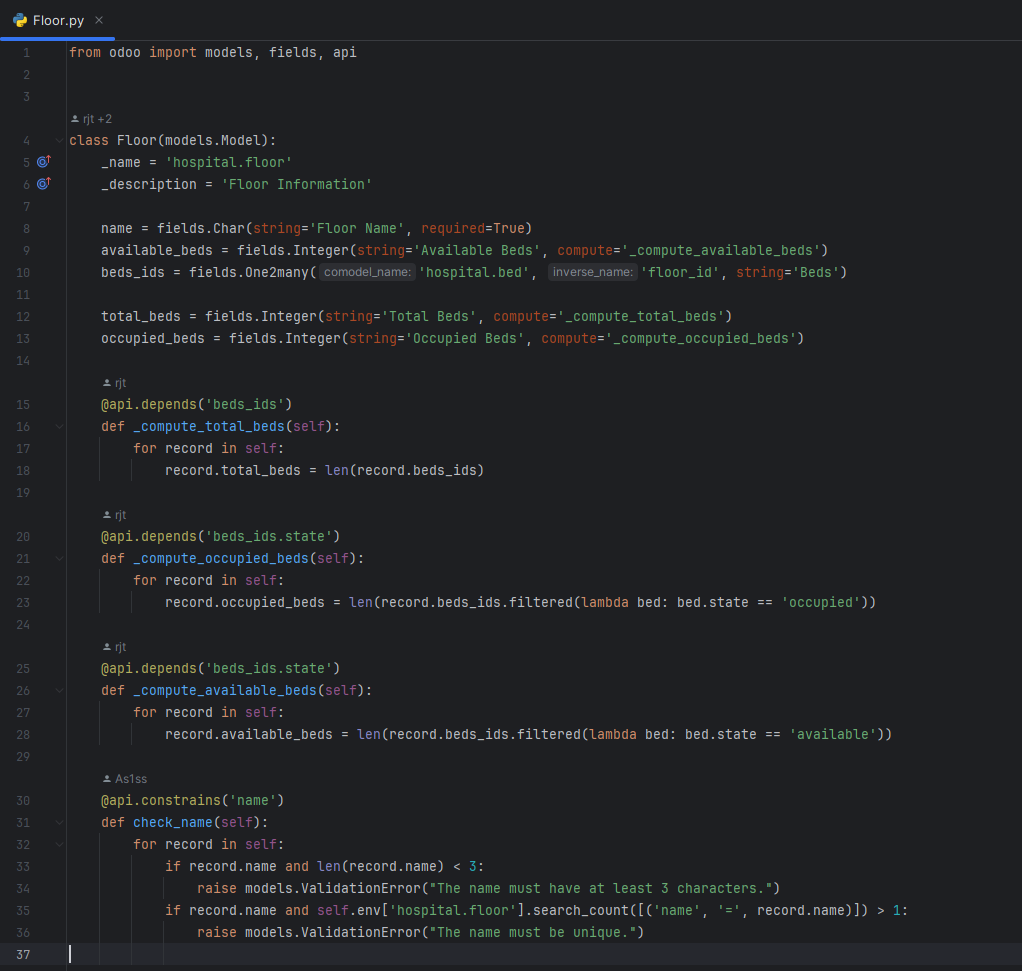
* En resumen, la clase Doctor en el modelo de datos del hospital extiende la funcionalidad básica de una persona para representar a los médicos, agregando campos específicos de médicos y estableciendo relaciones relevantes para gestionar la información de los médicos y sus interacciones con los pacientes y las admisiones hospitalarias.

hospital.bed



* La clase Bed en el modelo de datos del hospital representa las camas del hospital, proporcionando información sobre su nombre, estado, tipo y la relación con la planta del hospital en la que se encuentran, así como la información del paciente que ocupa la cama en un momento dado.

hospital.floor



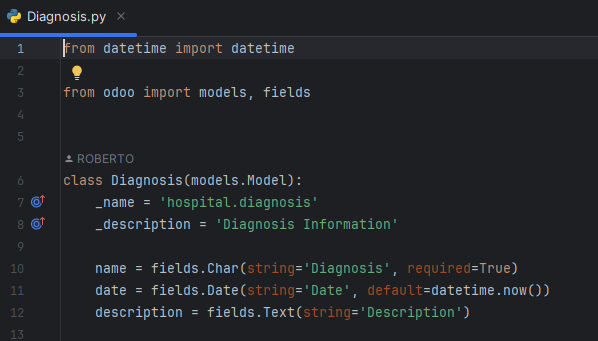
* La clase Floor en el modelo de datos del hospital representa los pisos del hospital, proporcionando información sobre el nombre del piso, el número total de camas, el número de camas ocupadas y disponibles, y la relación con las camas ubicadas en ese piso. Además, incluye restricciones para garantizar la integridad de los datos, como la unicidad del nombre del piso y la longitud mínima del nombre.

hospital.treatment



* La clase Treatment en el modelo de datos del hospital almacena información sobre tratamientos médicos, incluyendo el nombre del tratamiento, la fecha en que se realizó y una descripción opcional del tratamiento. Esto permite llevar un registro de los tratamientos médicos administrados a los pacientes en el hospital.

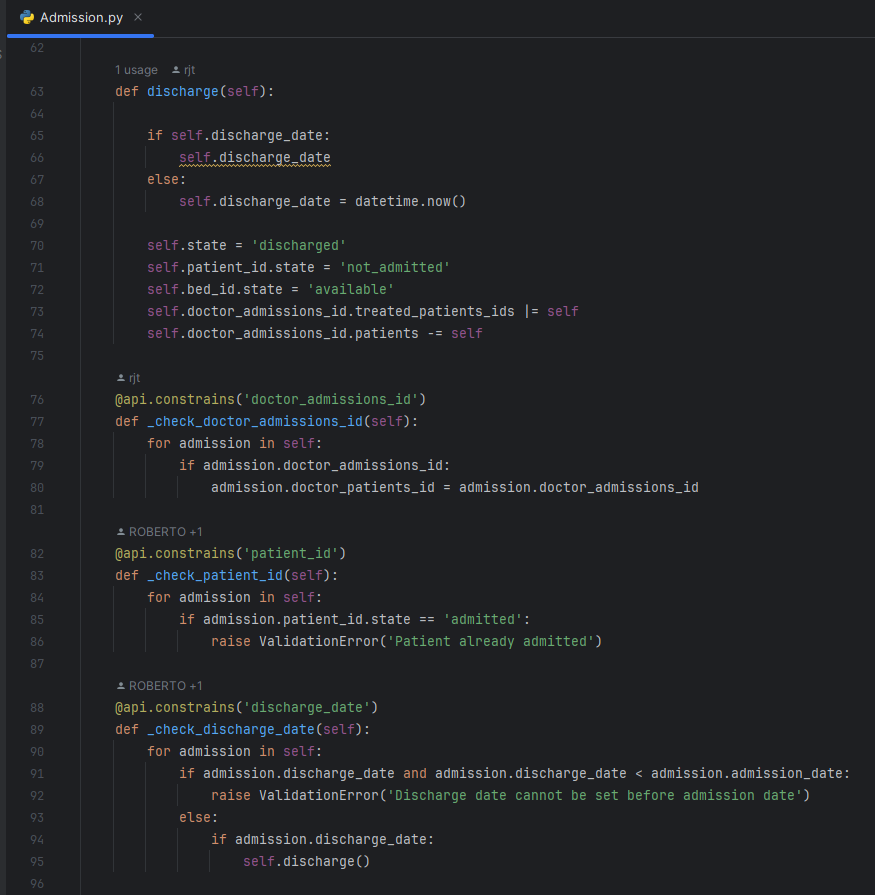
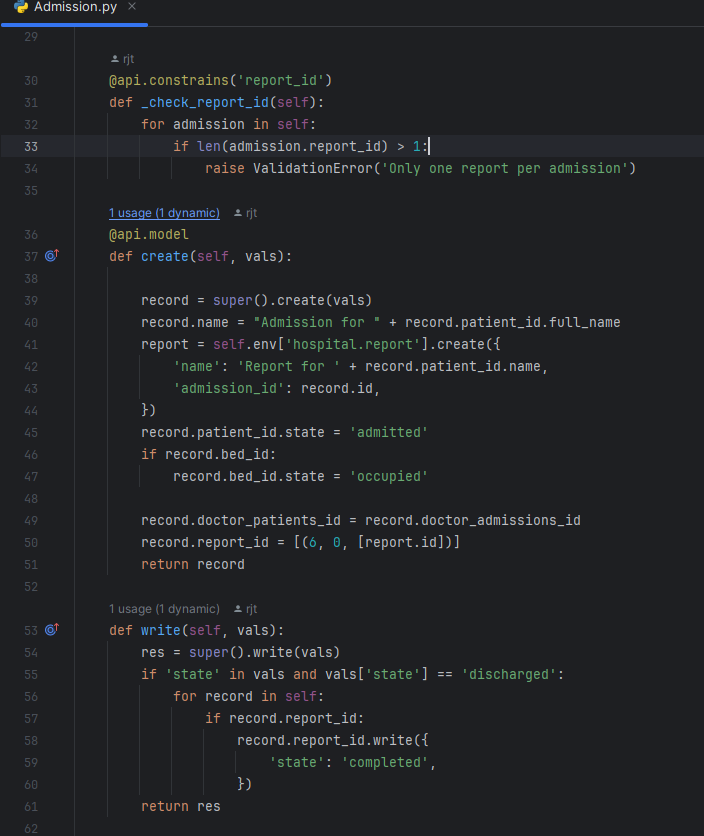
hospital.diagnosis



* La clase Diagnosis en el modelo de datos del hospital almacena información sobre diagnósticos médicos, incluyendo el nombre del diagnóstico, la fecha en que se realizó y una descripción opcional del mismo. Esto permite llevar un registro de los diagnósticos médicos realizados a los pacientes en el hospital.

hospital.admission

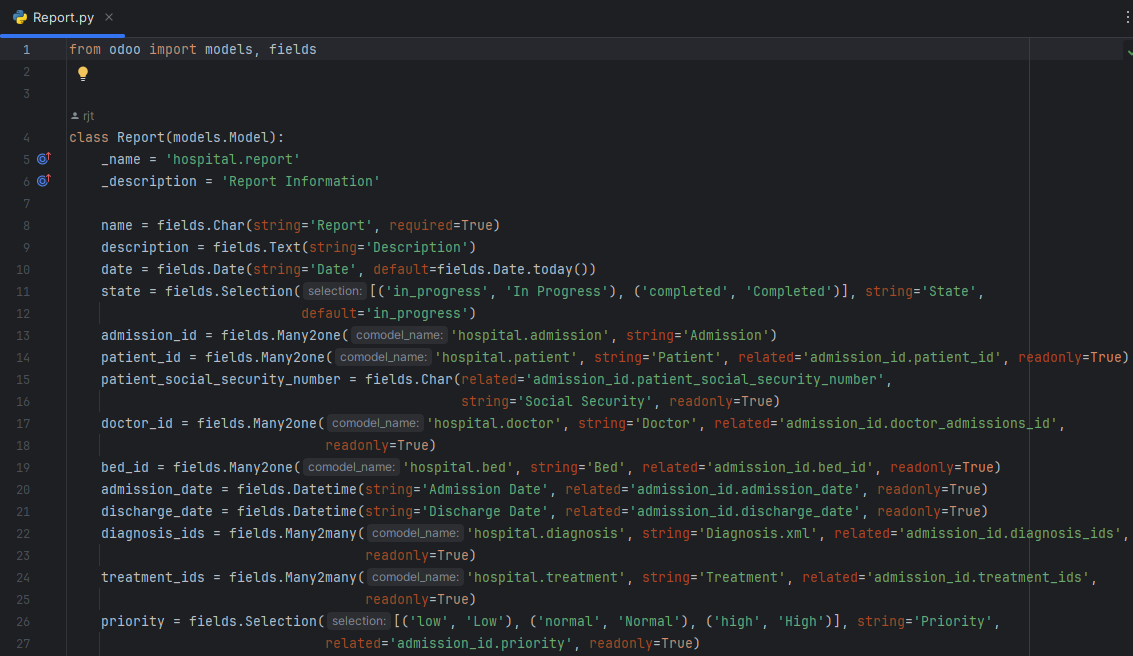






* La clase Admission en el modelo de datos del hospital gestiona la información relacionada con las admisiones hospitalarias, incluyendo los pacientes admitidos, los médicos encargados de la admisión, las camas asignadas, los diagnósticos y tratamientos asociados, el estado de la admisión y los informes generados durante la admisión.
* También proporciona métodos para realizar acciones como la creación de una admisión, la actualización de su estado y el alta del paciente.

hospital.report

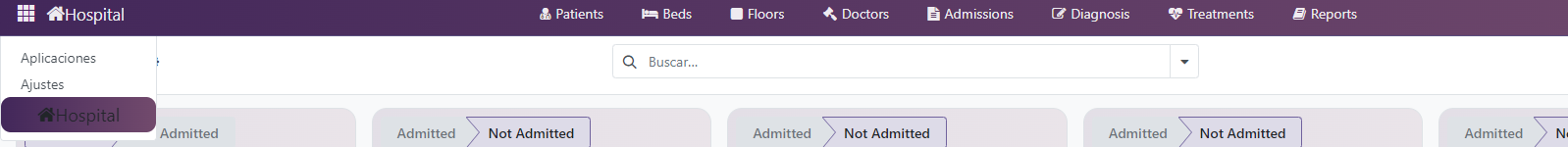


* La clase Report en el modelo de datos del hospital almacena información sobre los informes asociados a las admisiones hospitalarias. Esto incluye detalles sobre el paciente, el médico, la cama, la fecha de admisión y alta, los diagnósticos y tratamientos asociados, así como el estado del informe.

## Vistas y Formularios

* Este módulo ofrece soporte para diferentes tipos de vistas, incluyendo listas y formularios. Sin embargo, nos centraremos en resaltar las vistas tipo kanban, ya que son visualmente más atractivas y ofrecen una presentación óptima de la información.

Hospital



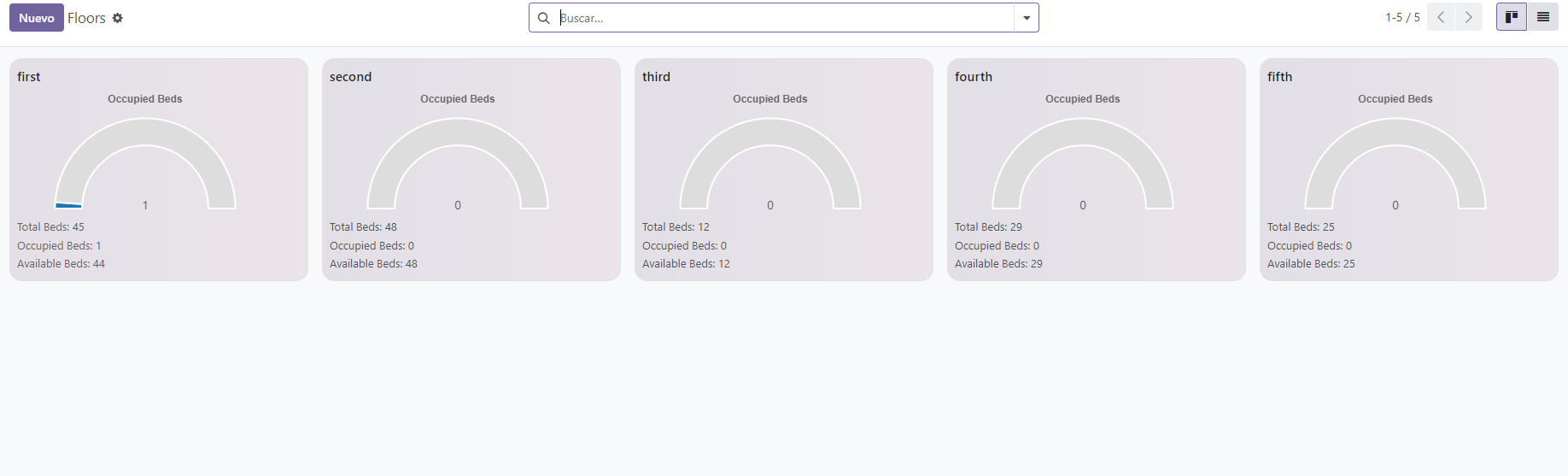
Patient



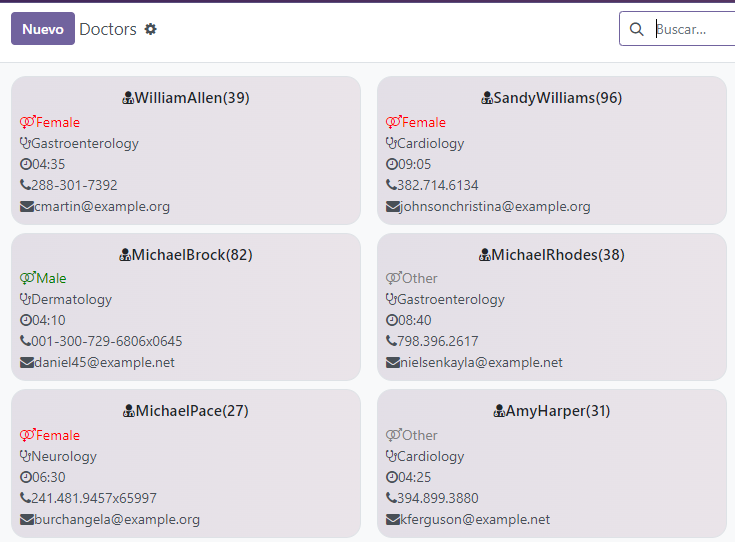
Bed



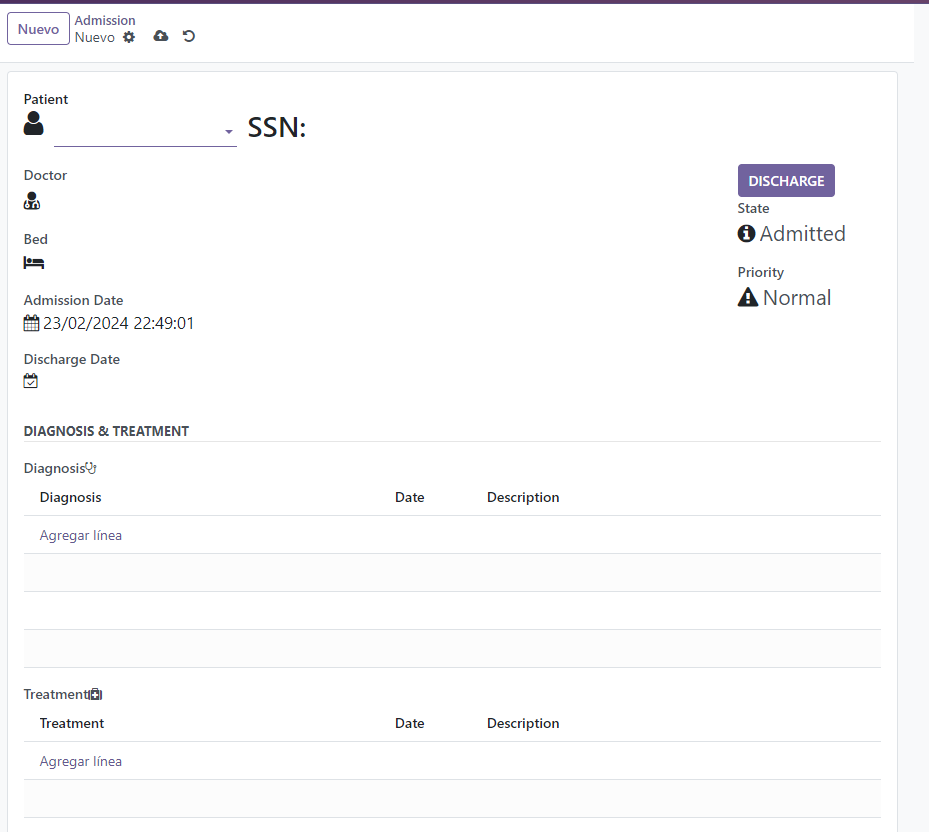
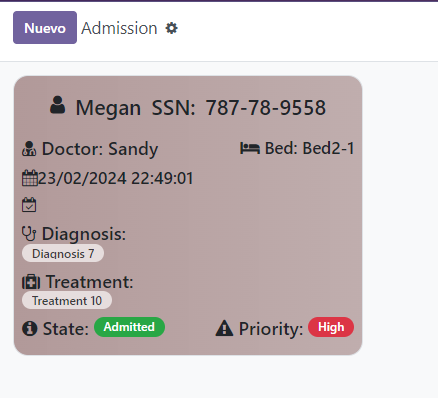
Floor



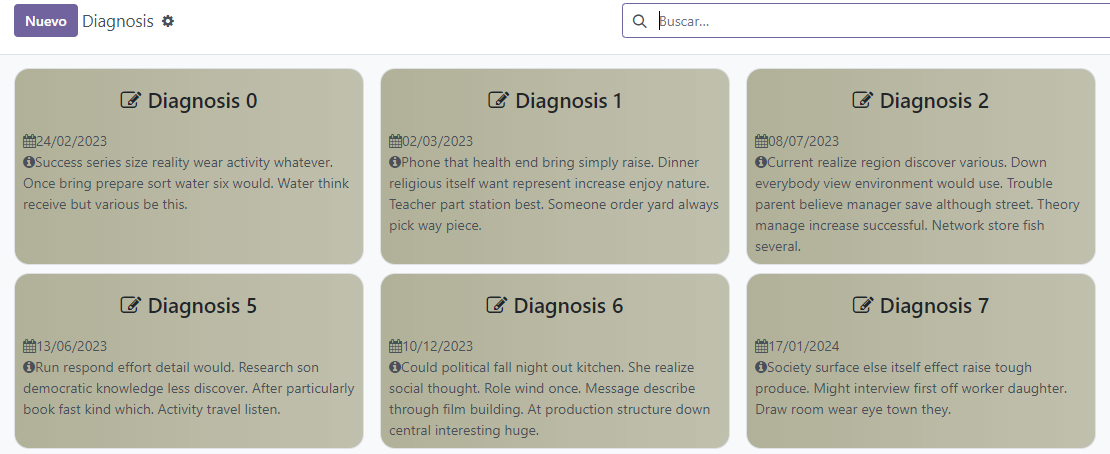
Doctor



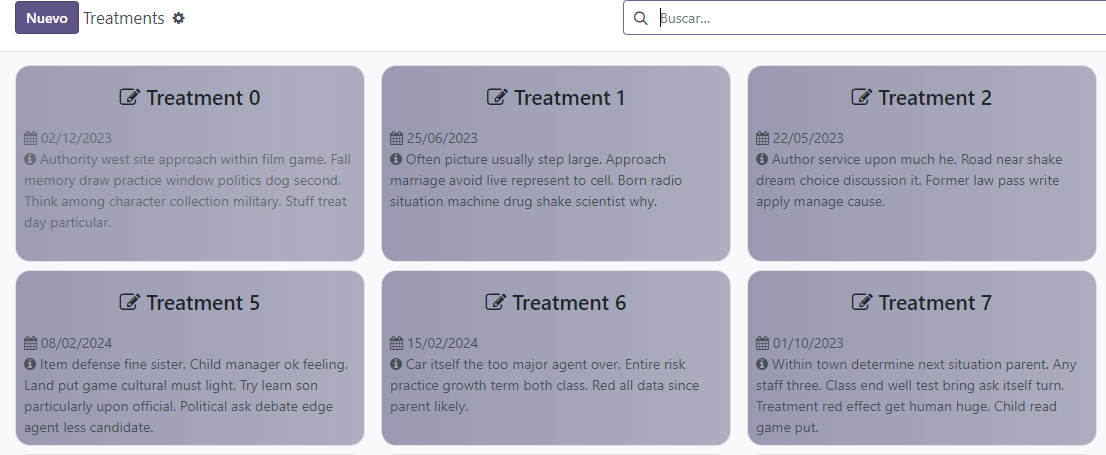
Admission



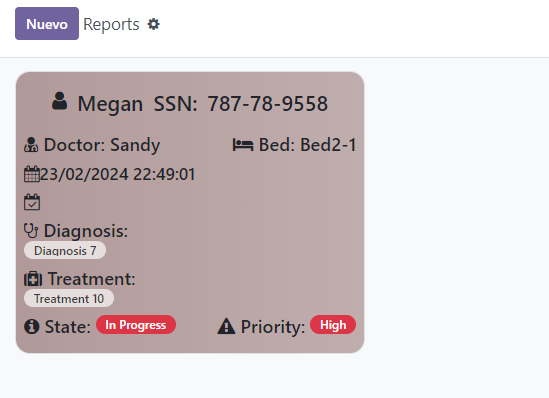
Diagnosis

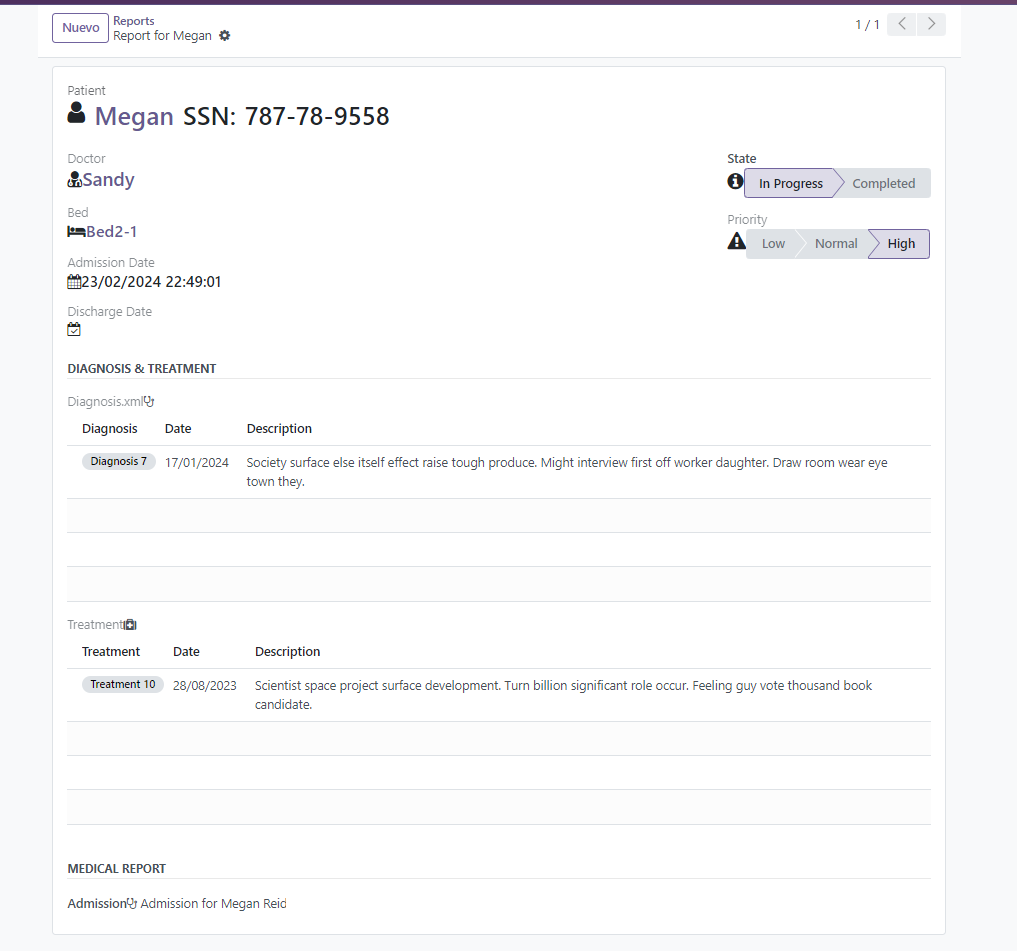


Treatments



Report





## Proceso de desarrollo

* Metodología de desarrollo: Freestyle
* Herramientas utilizadas para el desarrollo:
  + Github
  + GitKraken
  + Pycharm
  + Docker Desktop
  + Slacker

## Conclusiones

**Resumen de los logros del proyecto:**

* El proyecto ha logrado desarrollar un módulo integral para la gestión hospitalaria en la plataforma Odoo, que aborda de manera efectiva las necesidades de administración y atención médica en un entorno hospitalario.

**Durante el desarrollo del proyecto, se han identificado varias lecciones importantes que pueden ser útiles para futuros proyectos similares:**

* **La importancia de una planificación detallada:** La planificación adecuada de los requisitos del proyecto y la definición clara de los objetivos desde el principio son fundamentales para el éxito del proyecto.
* **La comunicación efectiva es clave:** Mantener una comunicación clara y constante entre todos los miembros del equipo de desarrollo, así como con los usuarios finales, es esencial para asegurar que el producto final cumpla con las expectativas y necesidades.
* **Flexibilidad para adaptarse a cambios:** Los requisitos del proyecto pueden cambiar a lo largo del tiempo, por lo que es importante mantenerse flexible y estar dispuesto a adaptarse a nuevas circunstancias y requerimientos.

**A pesar de los logros alcanzados, existen áreas de mejora que podrían explorarse en futuras iteraciones del proyecto:**

* **Ampliación de funcionalidades:** Se podrían agregar nuevas funcionalidades para cubrir aspectos específicos de la gestión hospitalaria que no fueron abordados en la versión inicial del módulo.