DOCUMENTACIÓN

Manual de operaciones

Descripción

Este documento contiene toda la información necesaria para la correcta mantención y ampliación del sistema hacia nuevos requerimientos.

Rodrigo Ignacio Zenteno Smith
i.zentenosmith@gmail.com

Tabla de contenido

Introducción

Este documento está orientado a la correcta mantención y extensión de la aplicación para agregar y/o modificar funcionalidades nuevas o ya existentes.

Se detallará cada uno de los componentes que componen la aplicación, desde su inicio hasta su final.

Requerimientos

Para esta aplicación se generaron los siguientes requerimientos:

- Registro y login del sistema
- Sistema de roles de usuario
- Permisos por roles
- Generar un menú por fecha
- Generar almuerzos por menú
- Enviar recordatorios por mensaje directo a través de Slack
- Seleccionar menú del día hasta las 11:00 CLT
- Especificar comentarios del menú
- Recordatorio debe ser una URL con UUID
- Recordatorio no debe requerir autenticación

Tecnologías

Para satisfacer todos estos requerimientos, se utilizaron las siguientes librerías y tecnologías.

Librerías:

- asgiref==3.2.7
- Django==3.0.4
- django-model-utils==4.1.1
- djangorestframework==3.12.2
- linecache2==1.0.0
- Pillow==7.0.0
- pip==19.2.3
- psycopg2==2.8.4
- pytz==2019.3

- setuptools==41.2.0
- six==1.15.0
- sqlparse==0.3.1
- traceback2==1.4.0
- Unipath==1.1
- unittest2==1.1.0

Lenguaje:

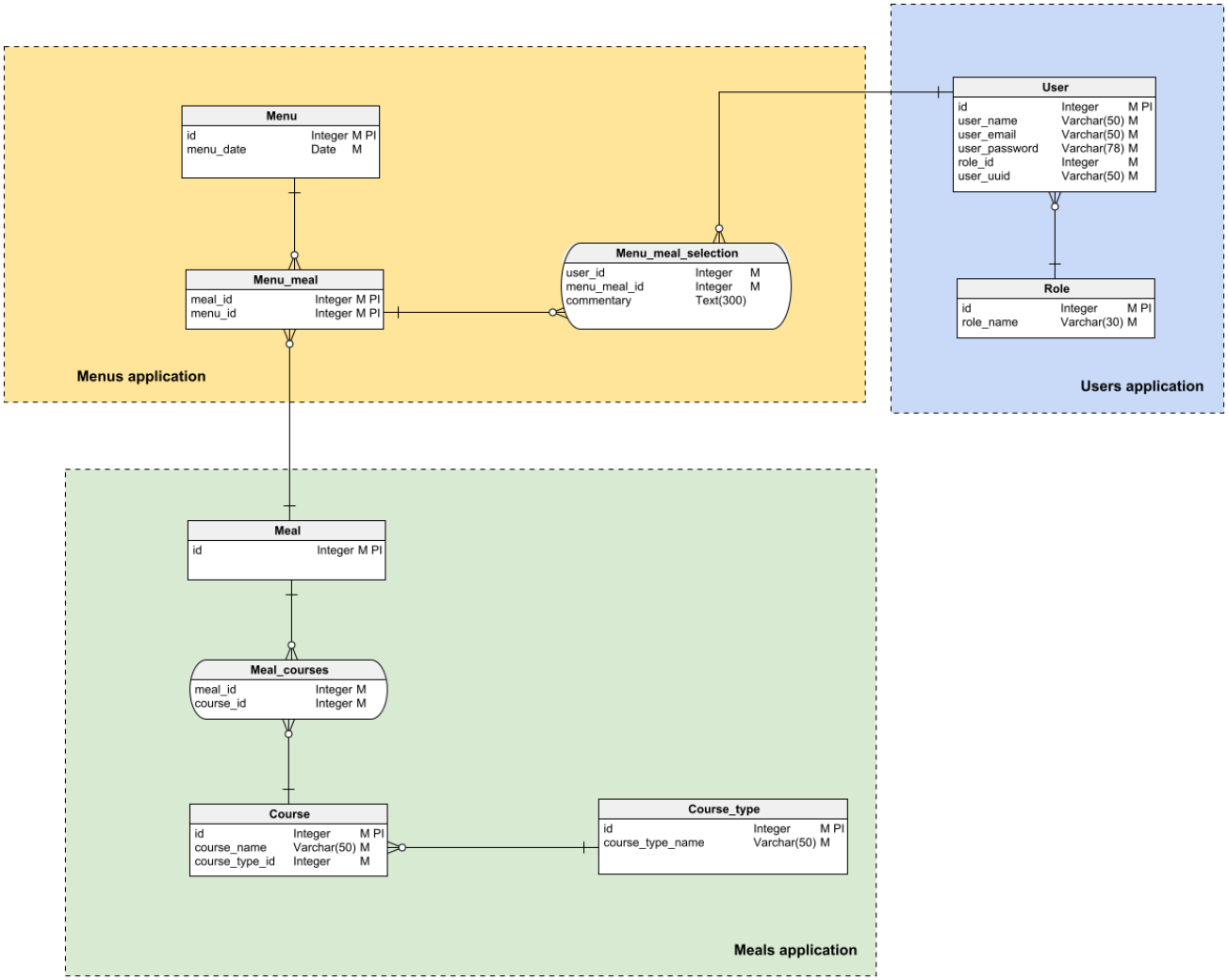
Python 3.7.6

Base de datos:

Postgres 12.5

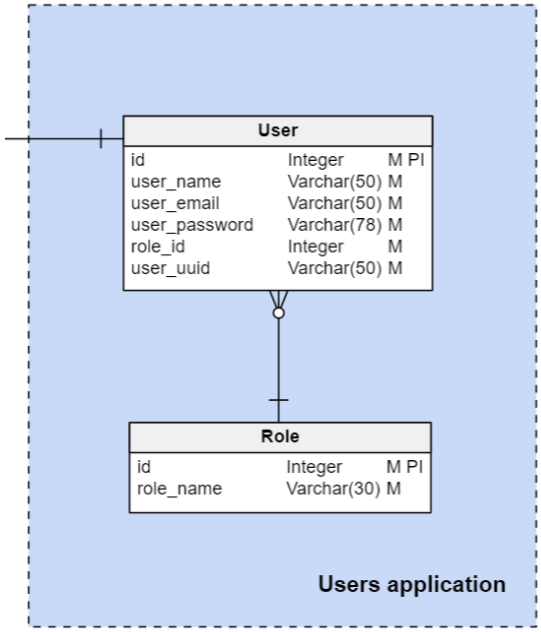
Base de datos

Para hacer frente a los requerimientos, se diseñó el siguiente MER (Modelo Entidad-Relacion). Para su diseño se utilizó Vertabelo.

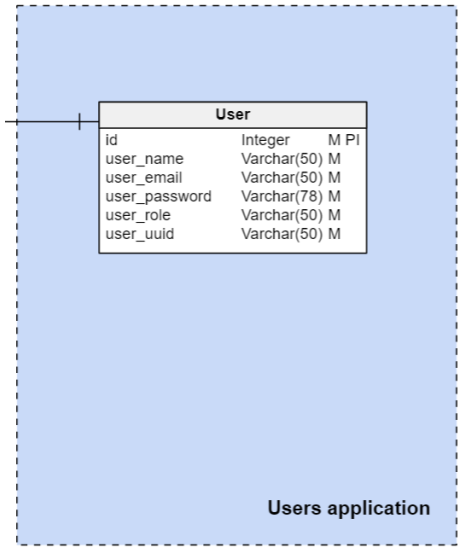


La aplicación se dividió en 3 aplicaciones principales, a continuación, se detallará el modelo para cada una de ellas.

MER Users

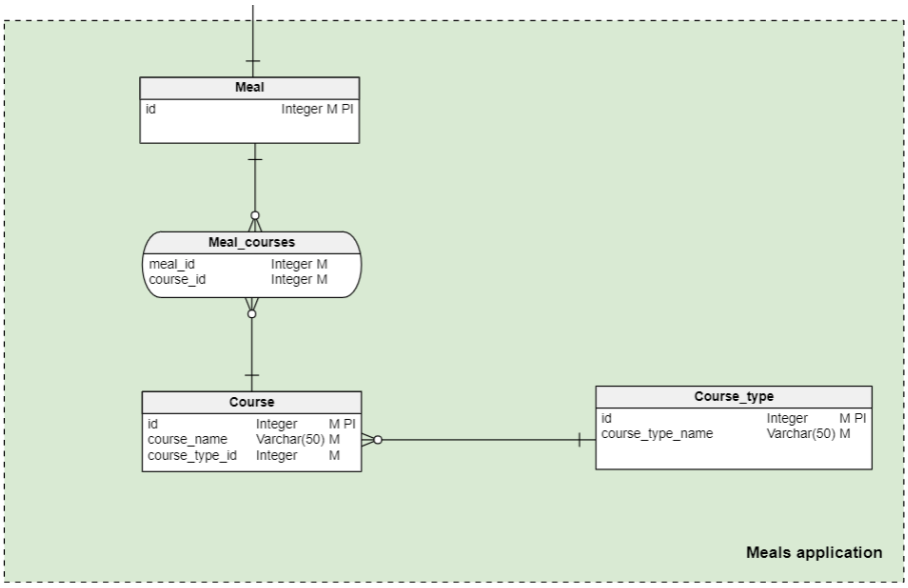


La entidad de usuarios contiene todo lo relacionado con los usuarios. Esta entidad tiene una relación uno a muchos con respecto a los roles (Un rol puede pertenecer a más de un usuario).



El diseño original es el de la primera imagen, mientras que el de la segunda imagen se utilizó una de las ventajas de Django para simplificar el modelo, el cual es el CHOICES. Más adelante en el documento se detallará sobre esa sección.

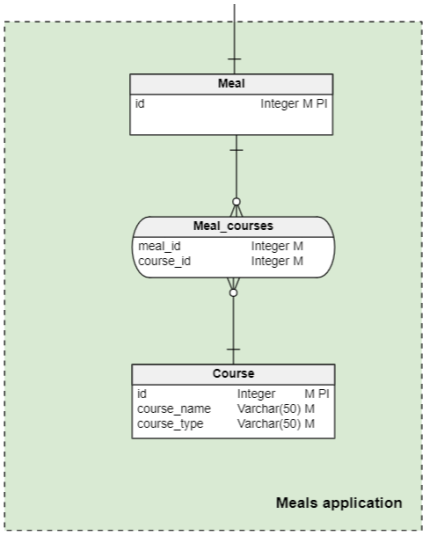
MER Meals



La entidad de Course hace referencia a los platos de un almuerzo, conocido en inglés como “Three course meal”. Course type hace relación a los tipos de plato (Entrada, plato principal y postre) y tiene una relación de uno a muchos respecto a los Courses (Un tipo puede pertenecer a más de un plato).

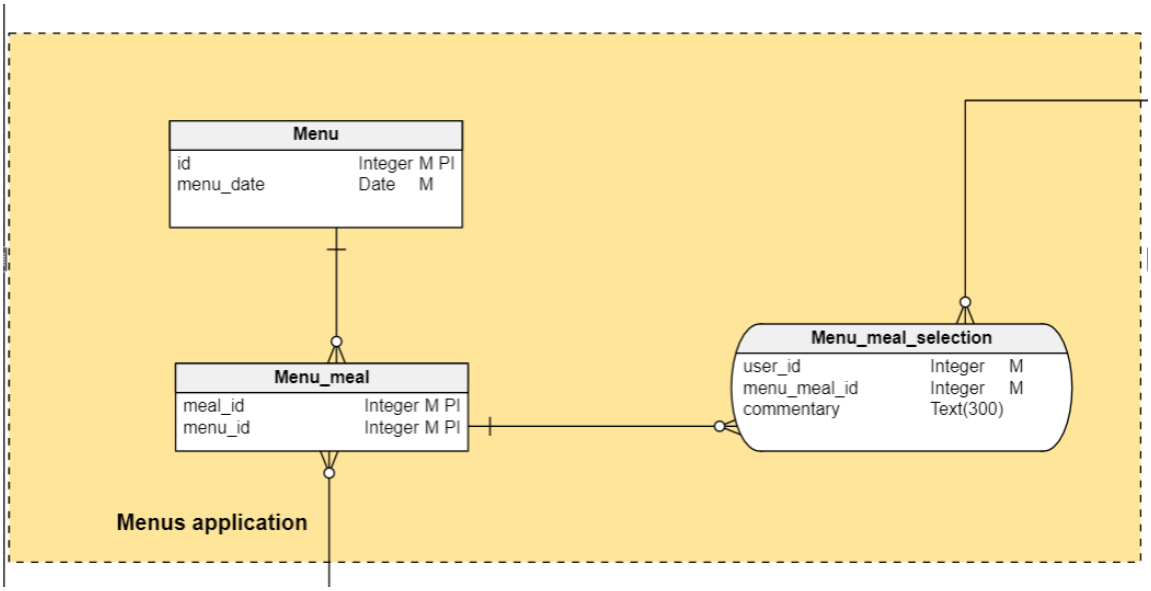
Luego, está la entidad de Meal, la cual hace referencia al almuerzo en si. Este almuerzo se diseñó como una compilación de platos.

La relación que une a estas dos entidades es la entidad asociativa Meal_courses, que hace la compilación entre almuerzos y sus platos respectivos.



El diseño original es el de la primera imagen, mientras que el de la segunda imagen se utilizó una de las ventajas de Django para simplificar el modelo, el cual es el CHOICES. Más adelante en el documento se detallará sobre esa sección.

MER Menu



La entidad menú es la entidad que relaciona la fecha con los almuerzos, es decir, un menú es una compilación de almuerzos para una fecha dada.

Esta relación está dada por la entidad asociativa entre Menu_meal y Meal.

Finalmente, la relación entre el menú de la fecha y el almuerzo escogido por el usuario está dada por la entidad asociativa compleja Menu_meal_selection, en donde también se especifica el comentario del usuario para su selección. Esta entidad asociativa tiene el apellido de compleja debido a que se relaciona a otra entidad asociativa.

Si bien se podría haber diseñado respecto a las entidades Menu y Meal directamente, esto hubiese traído problemas de consistencia de los datos.

La asociación de esta manera cumple también con la cuarta forma normal de las bases de datos.

Settings del proyecto

A diferencia de un proyecto normal de Django, se dividió el archivo settings.py en base.py, local.py y prod.py dentro de la carpeta settings en el directorio del proyecto.

Base.py

Este archivo contiene todos los settings que son comunes al ambiente local como de producción.

Local.py

Este archivo contiene todos los settings que son exclusivos del ambiente local.

Prod.py

Este archivo contiene todos los settings que son exclusivos del ambiente de producción. Dado que no se hizo un levantamiento a producción de la aplicación, este archivo se encuentra vacío.

Aplicación Users

Esta aplicación está directamente relacionada con el modelo Users, y se encarga de otorgar todas las funcionalidades relacionadas a los usuarios del sistema.

Esto incluye:

- Login
- Registro
- Editar perfil

Cuando un usuario se registra en el sistema, automáticamente se genera un token de usuario y un uuid del usuario. Estos datos serán necesarios para la autenticación sin login cuando vayan a seleccionar el menú del día.

Templates

Todos los templates que hacen relación a la aplicación están dentro de la carpeta /templates/users/…

Aquí se encuentran los templates para el login, el registro y editar perfil de usuario.

Views

Aquí se encuentra las vistas de la aplicación, en donde están las funcionalidades de registro (UserRegisterView), login (LoginUser), logout (LogoutView) y perfil de usuario (UserUpdateView).

Solo la vista de perfil requiere autenticación.

Urls

Aquí se encuentra la relación entre template y vistas, se siguió la siguiente regla:

Path: /user/funcionalidad

Vista: FuncionalidadView

Name: user-funcionalidad

Model

En este archivo se definió el modelo de Users respecto al MER generado en Vertabelo, en este archivo se aprovechó la ventaja de USER_ROLE_CHOICE para generar de manera fácil la relación de roles con los usuarios.

Además se utilizó la etiqueta USERNAME_FIELD para asignar el nombre de usuario al email del usuario, de esta forma el usuario debe hacer login a través de su email.

Cada vez que se registra un usuario, automáticamente se registra su UUID, el cual utiliza uuid4.

Además se aprovecha de generar su token de autenticación.

Manager

Aquí se definieron funciones comunes del modelo de usuarios, los cuales son la creación de usuario y la obtención de manera rápida de los emails de todos los usuarios. Esta función será necesaria para la API de Slack.

Tests

Para el testing de la aplicación de Users se siguió la siguiente regla:

Test_forms.py: Hace todo el testing relacionado a los formularios de la aplicación.

Test_models.py: Hace todo el testing relacionado a los modelos de la aplicación.

Test_urls.py: Hace todo el testing relacionado a las urls de la aplicación.

Test_views.py: Hace todo el testing relacionado a las vistas de la aplicación.

Aplicación Meals

Esta aplicación está directamente relacionada con los modelos Meal, Courses y su tabla asociativa respectiva.

Se encarga de las siguientes funcionalidades:

- Crear platos
- Crear almuerzos respecto a los platos
- Restricción de un almuerzo con un máximo de 3 platos y un mínimo de 1.
- Esta aplicación solo es accesible por el Lunch Manager.

Templates

Todos los templates que hacen relación a la aplicación están dentro de la carpeta /templates/meals/…

Como regla general, se utilizó la siguiente nomenclatura:

Entidad_crud.html

De esta forma es fácil e intuitivo ver que template está relacionado con que funcionalidad.

Además, se agregó un template general llamado meals_base.html que es común a todos los templates.

Views

Aquí se encuentra las vistas de la aplicación, cada una de estas vistas requiere estar con una sesión activa para poder acceder.

Como regla general, se utilizó la siguiente nomenclatura:

EntidadCrudView()

Urls

Aquí se encuentra la relación entre template y vistas, se siguió la siguiente regla:

Path: /meals/entidad_crud

Vista: EntidadCrudView

Name: entidad-crud

Model

En este archivo se definió el modelo de Course, Meal y Meal_course respecto al MER generado en Vertabelo, en este archivo se aprovechó la ventaja de COURSE_TYPE_CHOICE para generar de manera fácil la relación de courses (platos) con course_type (tipo de plato)

Se estableció unicidad conjunta entre el nombre de plato y su tipo, de manera de no repetir el mismo plato en el modelo Course.

Se aprovechó la ventaja de la etiqueta ManyToManyField en Meal para no definir de manera explicita la tabla asociativa y así aprovechar la funcionalidad de Django para su creación y mantención.

Manager

Aquí se definieron funciones comunes de los modelos. Se establecieron filtros de acceso rápido para la tabla de Courses (platos), los cuales fueron por nombre y por tipo.

Tests

Para el testing de la aplicación de Meals se siguió la siguiente regla:

Test_forms.py: Hace todo el testing relacionado a los formularios de la aplicación.

Test_models.py: Hace todo el testing relacionado a los modelos de la aplicación.

Test_urls.py: Hace todo el testing relacionado a las urls de la aplicación.

Test_views.py: Hace todo el testing relacionado a las vistas de la aplicación.

Aplicación Menus

Esta aplicación está directamente relacionada con los modelos Menu, la tablas asociativa Menu_meal y la tabla asociativa compleja Menu_meal_user

Se encarga de las siguientes funcionalidades:

- Generar un menú para una fecha, restricción de no editar ni crear menús para fechas pasadas.
- CRUD de menú
- Ver “mis selecciones de almuerzos”
- Seleccionar un almuerzo del menú del día

Templates

Todos los templates que hacen relación a la aplicación están dentro de la carpeta /templates/menus/…

Como regla general, se utilizó la siguiente nomenclatura:

menu_crud.html

De esta forma es fácil e intuitivo ver que template está relacionado con que funcionalidad.

Existen 2 excepciones a esta regla, los cuales son los templates que permiten ver los almuerzos seleccionados del usuario con sesión activa (menu_my_meals.html) y la selección del menú del día (Select_menu.html).

Además, se agregó un template general llamado menus_base.html que es común a todos los templates.

Views

Aquí se encuentra las vistas de la aplicación, cada una de estas vistas requiere estar con una sesión activa para poder acceder a excepción de la vista de SelectMenuView.

Como regla general, se utilizó la siguiente nomenclatura:

MenuCrudView()

Y se utilizaron 2 vistas adicionales para el manejo de los templates excepcionales:

MyMenuMealsView y SelectMenuView.

Urls

Aquí se encuentra la relación entre template y vistas, se siguió la siguiente regla:

Path: /menus/menu_crud

Vista: MenuCrudView

Name: menu-crud

Para la URL de seleccionar un menú, se utilizó el UUID del usuario para poder obtener de esta manera quien es el usuario.

Model

En este archivo se definió el modelo de Menu, Menu_meal y Menu_meal_user, a diferencia de los modelos de Meal, aquí se definieron explícitamente los modelos asociativos, ya que de esta manera se obtiene un control más riguroso sobre los datos.

Manager

Aquí se definieron 3 funciones para el modelo Menu, los cuales son: Saber si una fecha existe, editar si su fecha es distinta y chequear si el almuerzo es valido para el menú de hoy.

Tests

Para el testing de la aplicación de Menus se siguió la siguiente regla:

Test_forms.py: Hace todo el testing relacionado a los formularios de la aplicación.

Test_models.py: Hace todo el testing relacionado a los modelos de la aplicación.

Test_urls.py: Hace todo el testing relacionado a las urls de la aplicación.

Test_views.py: Hace todo el testing relacionado a las vistas de la aplicación.

Aplicación Apirest

Esta aplicación está directamente relacionada con la conexión entre la aplicación y Slack.

Slackapi

En este archivo se define la clase con todos los métodos necesarios de la conexión entre Slack y el sistema.

En el método constructor de la clase se define la instancia de la librería Slacker con el token del bot de Slack.

En el método get_slack_users se obtienen todos los usuarios de slack y se arma un arreglo de diccionarios con los datos relevantes de cada uno de los usuarios.

En el método get_empleados_from_slack_users se obtienen todos usuarios que son empleados dentro de la aplicación a partir del arreglo de diccionarios obtenido en get_slack_users. En caso de existir este usuario, se asigna el token y el uuid al diccionario del usuario.

En el método send_empleados_direct_message se envían las notificaciones de los almuerzos con el link y su UUID respectivo.

En el método send_slack_message se establece una forma default de enviar mensajes a la aplicación, en donde si no se especifica el canal, automáticamente se envía al canal General.

Aplicación Home

Esta aplicación permite el acceso a la página de inicio de la aplicación.

Sobre las notificaciones

Debido a falta de tiempo, se implementó el envío de notificaciones de manera síncrona a través del módulo de menús.