21-6-2019

ISMAEL PRADAS GÓMEZ

I.E.S Vega de Mijas – 2DAW

Proyecto Appetito

Gestión de cartas y búsqueda de comida.

Índice

[1. Appetito 3](#_Toc11774667)

[1.1 ¿Qué es Appetito? 3](#_Toc11774668)

[1.2 Requisitos funcionales específicos de Appetito 3](#_Toc11774669)

[1.2.1 Funcionalidades para restauradores 4](#_Toc11774670)

[1.2.2 Funcionalidades para clientes 5](#_Toc11774671)

[2. Requisitos para el desarrollo 6](#_Toc11774672)

[2.1 Requisitos de hardware 6](#_Toc11774673)

[2.2 Requisitos de software 6](#_Toc11774674)

[3. Estructuras de datos 8](#_Toc11774675)

[3.1 Especificaciones de la base de datos 8](#_Toc11774676)

[3.2 Diseño E-R de la base de datos 10](#_Toc11774677)

[3.3 Diseño relacional de la base de datos 11](#_Toc11774678)

[4. Estructura del proyecto 12](#_Toc11774679)

[4.1 Modelos 14](#_Toc11774680)

[4.2 Aplicaciones 14](#_Toc11774681)

[4.3 Multimedia 16](#_Toc11774682)

[4.4 Estáticos 17](#_Toc11774683)

[5. Despliegue de la aplicación 17](#_Toc11774684)

[5.1 Requisitos previos 17](#_Toc11774685)

[5.2 Proceso de despliegue 18](#_Toc11774686)

[5.2.1 Fichero manage.py 18](#_Toc11774687)

[5.2.2 Preparación de los ficheros estáticos 18](#_Toc11774688)

[5.2.3 Migración de la base de datos 19](#_Toc11774689)

[5.2.4 Creación el superusuario 19](#_Toc11774690)

[5.2.5 Ejecución de la aplicación 19](#_Toc11774691)

[5.2.6 Configuraciones iniciales 20](#_Toc11774692)

[6. Manual de uso 20](#_Toc11774693)

[6.1 Uso por clientes 20](#_Toc11774694)

[6.1.1 Diagrama de uso por clientes 20](#_Toc11774695)

[6.1.2 Búsqueda y filtrado de resultados 21](#_Toc11774696)

[6.2 Uso por restauradores 22](#_Toc11774697)

[6.2.1 Registro 23](#_Toc11774698)

[6.2.2 Logueo 23](#_Toc11774699)

[6.2.3 Creación de empresa 24](#_Toc11774700)

[6.2.4 Panel de control 24](#_Toc11774701)

[6.2.5 Restaurante 26](#_Toc11774702)

[6.2.6 Carta 26](#_Toc11774703)

[6.2.7 Ubicaciones 30](#_Toc11774704)

[6.2.8 Plan 31](#_Toc11774705)

[6.3 Uso por administrador 32](#_Toc11774706)

[6.3.1 Administración de Django 32](#_Toc11774707)

[7. Posibles mejoras 32](#_Toc11774708)

[7.1 Mejoras a corto plazo 32](#_Toc11774709)

[7.2 Mejoras a largo plazo 33](#_Toc11774710)

[7.2.1 Especificaciones de mejora de clientes 33](#_Toc11774711)

[8. Bibliografía 34](#_Toc11774712)

[9. Dependencias 35](#_Toc11774713)

[10. Recursos 35](#_Toc11774714)

[11. Conclusiones 36](#_Toc11774715)

[11.1 Conclusiones sobre el proyecto 36](#_Toc11774716)

[11.1.1 Durante el curso 36](#_Toc11774717)

[11.1.2 Pre proyecto 36](#_Toc11774718)

[11.1.3 Durante el proyecto 37](#_Toc11774719)

[11.1.4 Finalización 37](#_Toc11774720)

[11.2 Conclusiones personales 38](#_Toc11774721)

# Appetito

## ¿Qué es Appetito?

Appetito es una aplicación web dispuesta para restauradores. Los restauradores podrán crearse una cuenta gratuita en la aplicación. La aplicación pone a su disposición un panel de control para gestionar su restaurante, las posibles ubicaciones en las que se encuentren establecimientos de su cadena de restaurantes, así como poder componer su carta online. Gracias al motor de búsqueda interno, todo este contenido se pondrá a disposición posteriormente de los clientes, quienes podrán realizar una búsqueda de cualquier plato de comida que le apetezca en ese momento y Appetito le mostrará aquellos restaurantes cercanos que le pueden ofrecer dicho servicio.

Adicional a la cuenta gratis, Appetito pone a disposición de los restauradores planes que incrementan las posibilidades de sus configuraciones. Dichas se adquieren con un pago único y sin caducidad, y que le permitirá, entre otras cosas, incrementar el total de ubicaciones en las que puede ubicar sus restaurantes, aumentar la cantidad de categorías y platos que le permite incorporar a su carta, así como, en futuras releases, crear anuncios para sus productos que se les mostrarán a sus usuarios.

Básicamente la funcionalidad de Appetito es la siguiente:

* Un usuario puede dar de alta un restaurante.
* El alta estará suscrita a un plan (FREE: Puedes añadir una sola categoría con un máximo de 10 productos con sus fichas, STANDARD: Puedes añadir un máximo de 30 productos con sus fichas y podrás añadir 3 tags a los productos para que se encuentren por #tags además de por su nombre, PRO: Sin máximo de categorías y productos, hasta 7 tags por producto y además podrás crear anuncios, y si en una búsqueda tu restaurante queda dentro de la zona de búsqueda, aunque tu producto no esté listado, tendrás una posibilidad de que se muestre tu anuncio.)
* El usuario puede modificar la información del restaurante (Nombre, ubicación, descripción, foto corporativa
* El usuario podrá crear **Categorías** (Entrante, pastas, ensaladas, carnes, etc.). Las categorías tendrán un icono representativo que será el mostrado en el mapa de resultados de búsqueda.
* El usuario podrá añadir productos a esas categorías (Espagueti, Ensalada tropical, Hamburguesa bestial) Los productos tendrán un nombre, un precio (a mostrar o no), una descripción, y una foto.
* El usuario podrá ver y modificar su producto.
* El usuario podrá añadir/eliminar X tags a esos productos para aumentar las posibilidades de que sea listado.

## Requisitos funcionales específicos de Appetito

Como se ha listado anteriormente, Appetito hace frente a la necesidad de dos grandes sectores. Por una parte y siendo este el más relevante, el sector de la restauración. Desde pequeños negocios familiares hasta grandes cadenas de comida, Appetito pone a disposición de los empresarios las herramientas necesarias y adaptadas a su volumen de negocio.

Por otra parte, Appetito hace frente a la necesidad de los clientes, ya sean locales o desplazados a la zona por algún motivo. Para ellos ofrece un buscador que, utilizando toda la información creada por los restauradores, mostrará al cliente una lista de negocios cercanos a su ubicación en los que podrá adquirir el producto que desee.

Es común, sobre todo cuando el cliente visita zonas desconocidas para él, tener dudas sobre qué se ofrece en cada restaurante o bar, así como su precio. O bien quiere encontrar una receta en concreto, pero no sabe bien si está disponible en todos los restaurantes. Appetito le ofrece a ese cliente toda la información necesaria para cubrir su necesidad.

En los siguientes puntos se describe la funcionalidad total de acuerdo a cada sector.

### Funcionalidades para restauradores

Las operaciones que pueden realizar los restauradores en Appetito son las siguientes:

* **Registro y logueo:** Appetito cuenta con un sistema de registro para empresarios. Solo es necesario para el empresario decidir un nombre de usuario, usar su correo profesional y asegurar su cuenta con una contraseña.

Una vez que el restaurador ha creado su cuenta podrá loguearse en Appetito para hacer sus gestiones necesarias.

* **Creación y edición de su negocio:** El primer paso y obligatorio para el restaurador es configurar su negocio. Appetito le ofrece al empresario un panel para decidir el nombre de su empresa, su logo, su descripción y una imagen representativa.
* **Subscripción a planes:** Como existen distintas modalidades de negocio, Appetito pone a la disposición de los restauradores distintos planes (subscripciones) que le permiten aumentar sus límites de creación.   
  Cualquier cuenta recién incorporada a Appetito tendrá automáticamente una subscripción gratuita.Las subscripciones actuales y sus límites son los siguientes:
  + **Plan Gratuito:**
    - Permite crear y editar el restaurante.
    - Permite añadir una ubicación el restaurante.
    - Permite crear una categoría en la carta del restaurante.
    - Permite añadir 5 platos a cada categoría de la carta.
    - Permite añadir 3 tags a cada plato para aumentar su posicionamiento en las búsquedas.
  + **Plan Básico:**
    - Permite crear y editar el restaurante.
    - Permite añadir hasta 5 ubicaciones a la cadena de restaurantes.
    - Permite crear hasta 5 categorías en la carta del restaurante.
    - Permite añadir 5 platos a cada categoría de la carta.
    - Permite añadir hasta 7 tags a cada plato de la carta.
  + **Plan Pro:**
    - Permite crear y editar el restaurante.
    - Permite añadir ubicaciones sin límites al restaurante (para grandes cadenas de restauración).
    - Permite crear categorías sin límite (cartas tan amplias como se quieran)
    - Permiten añadir platos ilimitados a las categorías.
    - Permite añadir hasta 12 tags a cada plato.

El precio de cada plan es diferente. El plan **Gratuito** como dice su nombre, es gratuito, y es el plan por defecto para cada cuenta nueva. El resto de plan tienen un precio de:

* Básico: 15,99€ (Pago único).
* Pro: 44,99€ (Pago único).

Los planes no tienen periodicidad ni fecha de caducidad. Appetito cuenta con una pantalla de pago en la cual le permite al restaurador realizar el pago para ascender de plan.

* **Gestión de ubicaciones:** Todos los restaurantes deben tener como mínimo una ubicación. Sin ella, será imposible situar sus productos en el mapa. Para ello, se pone a disposición de los restauradores una sección en el panel de control para marcar tantas ubicaciones como su plan le permita.
* **Gestión de carta:** La gestión de categorías y platos se hace en una misma pantalla, en la que el restaurador puede gestionar su carta.  
    
  Appetito le permite crear, modificar y eliminar tantas categorías como su plan le permita.  
    
  Además, y de nuevo dependiendo de los límites de su plan, podrá crear tantos platos como pueda en sus categorías, así como editarlos y/o eliminarlos.  
  Con toda esa información, Appetito es capaz de situar dichos platos en las ubicaciones creadas por el restaurante, así como de montar una carta entera que será consultable por los clientes.

### Funcionalidades para clientes

En cuanto a los clientes, el servicio principal que les ofrece Appetito es el de la búsqueda personalizada y centrada de productos de consumo en los restaurantes.

Derivado de esta funcionalidad, el cliente tendrá acceso a los perfiles de las empresas para poder visualizar toda su información así de poder visualizar su carta completa.

# Requisitos para el desarrollo

## Requisitos de hardware

Ha sido necesario un equipo común con capacidad para correr Python. No se necesitan especificaciones muy exigentes por lo que no hay problema en usar cualquier equipo informático.

## Requisitos de software

* Python 3.7 o superior con los siguientes paquetes:
* asn1crypto==0.24.0
* astroid==2.2.5
* autopep8==1.4.4
* cairocffi==1.0.2
* CairoSVG==2.4.0
* certifi==2019.3.9
* cffi==1.11.5
* chardet==3.0.4
* colorama==0.4.1
* cryptography==2.4.2
* cssselect2==0.2.1
* defusedxml==0.6.0
* Django==2.2
* django-crispy-forms==1.7.2
* django-template-maths==0.2.0
* html5lib==1.0.1
* idna==2.8
* isort==4.3.17
* lazy-object-proxy==1.3.1
* mccabe==0.6.1
* menuinst==1.4.14
* Pillow==6.0.0
* Pmw==2.0.0
* preppy==2.7.0
* pycodestyle==2.5.0
* pycosat==0.6.3
* pycparser==2.19
* pylint==2.3.1
* pylint-django==2.0.8
* pylint-plugin-utils==0.5
* pyOpenSSL==18.0.0
* pyPdf==1.13
* PyPDF2==1.26.0
* Pyphen==0.9.5
* pyRXP==2.2.0
* PySocks==1.6.8
* pytz==2019.1
* pywin32==223
* reportlab==3.5.23
* requests==2.21.0
* rlextra==3.5.23
* ruamel-yaml==0.15.46
* six==1.12.0
* sqlparse==0.3.0
* tinycss2==1.0.2
* typed-ast==1.3.1
* urllib3==1.24.1
* virtualenv==16.6.0
* webencodings==0.5.1
* win-inet-pton==1.0.1
* wincertstore==0.2
* wrapt==1.11.1
* xhtml2pdf==0.2.3
* Visual Studio Code
* DB Browser SQLite
* Google Maps

**Librerías y helpers:**

* Bootstrap
* JQuery
* Google Maps JS
* Slick Carousel

# Estructuras de datos

## Especificaciones de la base de datos

|  |  |
| --- | --- |
| Entidades |  |
| Atributos |  |
| Claves primarias |  |
| Relaciones |  |
| Requisitos a mostrar |  |

* Se dan de alta **usuarios.** Se identificarán por un **ID único**.
* Cada usuario tendrá un **perfil** que contendrá información adicional sobre su **restaurante**. almacenarán los siguientes datos: **Nombre del restaurante, Direcciones físicas, Tlf principal, E-mail, imagen corporativa, e imagen de banner.**

\*Permitir que los restaurantes tengan varias ubicaciones para contemplar las cadenas de restaurantes y que sean localizables con una sola entidad de restaurante.

* Los perfiles **restaurante estarán sujetos a una** **subscripción**. Además de su **ID**, las suscripciones tienen un **nombre, una descripción corta, precio**, y contienen información sobre sus límites: **cantidad de ubicaciones permitidas, cantidad de categorías** permitidas, **cantidad de productos** permitidos, **cuántos tags** permite por producto.

\*Posible mejora: Estos campos se replicarán en la empresa para poder personalizarlos.

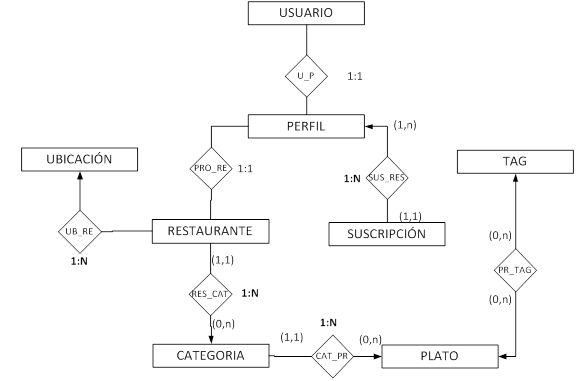
\*Posible mejora: En un principio la subscripción es única y sin caducidad, aunque puedes “ascender” pagando la diferencia + algo.

\*Posible mejora: Los ascensos son: Natural (La empresa hereda las propiedades de la suscripción); Incremental (La empresa “suma” las propiedades de la suscripción, por si tuviera un “dedazo” en sus propiedades, para que no se reseteen.); De clave de referencia (La empresa cambia de subscripción, pero mantiene sus propiedades manuales)

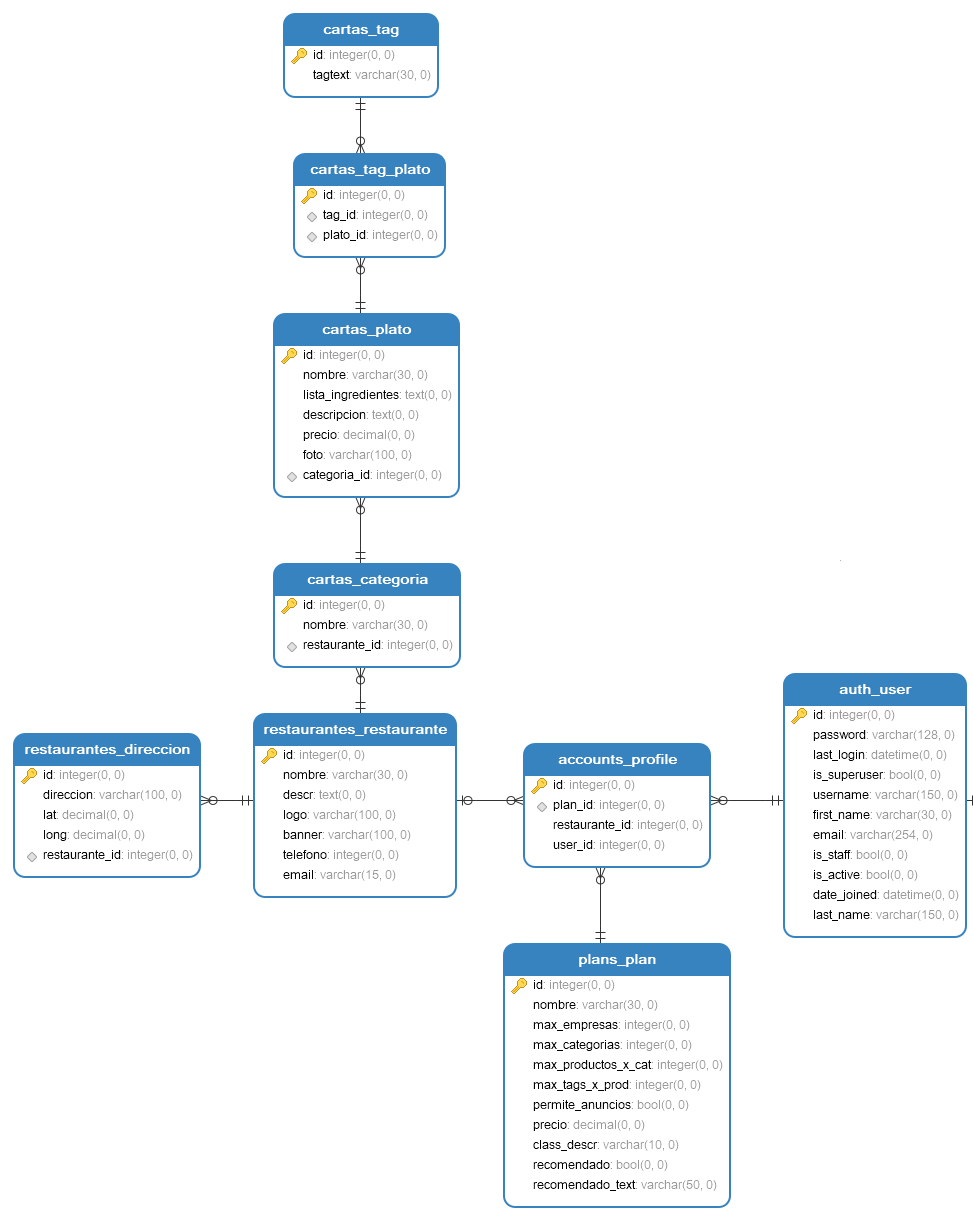
* Los planes pueden tener límites ilimitados, aunque suene extraño. Quiere decir que pueden no presentar restricciones a las direcciones máximas, o platos, categorías y/o tags.
* Los restaurantes podrán componer una carta. Se organiza de la siguiente forma:
  + El **restaurante crea** **categorías** con un **nombre descriptivo**.
  + Dentro de esas **categorías el restaurante añade productos**. Esos productos tendrán un **nombre, una descripción, una foto, un precio y una lista de ingredientes.**
  + El resultado de mostrar cada categoría con sus productos será **la carta** del restaurante.

Los **productos** creados por el restaurante podrán, según su plan, **tener o no tags.** Los tags simplemente tendrán, además de su **ID** y el **texto**.

## Diseño E-R de la base de datos



## Diseño relacional de la base de datos



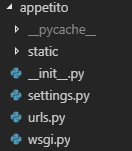
# Estructura del proyecto

En cuanto a la estructura de directorios del proyecto, este se ve sometido a las necesidades de Django Framework.

Tanto Python como Django siguen la filosofía de que cada funcionalidad debe ser una APP completa, así que si por ejemplo tienes una funcionalidad que es gestionar platos de comida, se creará una aplicación que realice todas las funciones necesarias (crearlos, editarlos, eliminarlos, etc.) y no se desvincularán entre otras aplicaciones.

Bajo ese concepto, en Appetito encontramos por lo tanto esa estructura de aplicaciones.

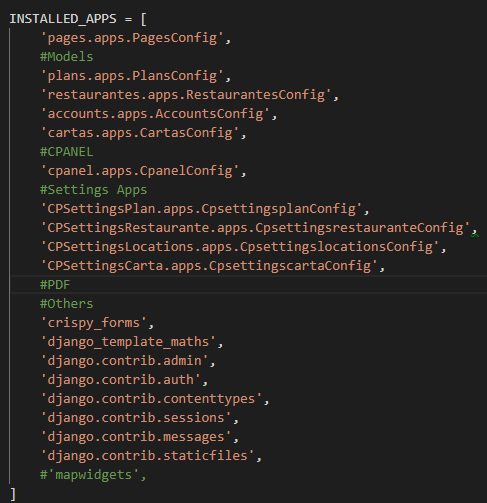
Además, y para fomentar la reutilización, los modelos en este proyecto también han sido tratado como aplicaciones. Así, si en una aplicación creamos un modelo para “Plato de comida”, este podrá ser reutilizado en cualquier proyecto que trabaje con platos de comida. Ya será trabajo del programador crear las aplicaciones para gestionarlas, etc. Cuando creamos un proyecto de Django, solamente tenemos una estructura como la de la imagen:



Dichos ficheros son los más importantes para nuestro proyecto ya que son los de arranque. Sus funcionalidades son:

* \_\_**pycache**\_\_: Almacena un caché de la aplicación para optimizar su arranque y funcionamiento.
* **Static**: Almacena ficheros estáticos de la aplicación.
* \_\_**init**\_\_.py: fichero necesario en cualquier aplicación Python para que el directorio en el que se encuentra sea reconocido como ejecutable. (Aunque el fichero esté vacío, si Python encuentra un \_\_init\_\_.py en un directorio, sabrá que dentro se encuentra un programa Python. Si no lo encuentra, lo tratará como un directorio cualquiera y no podrá realizar sus funciones.
* **Settings**.py: este es el fichero más importante. En él se encuentran las principales configuraciones de nuestro proyecto, como pueden ser:
  + Directorio raíz de la aplicación
  + Origen de los ficheros multimedia
  + Configuraciones de servicios externos (e-mail, exportar pdf)
  + Configuraciones de las bases de datos
  + Configuraciones de entornos
  + Lista de hosts permitidos
  + Middlewares de nuestra aplicación
  + Configuración raíz de las urls
  + Configuración de templates instalados.
  + Validadores de seguridad para las contraseñas
  + Idioma y localización
  + Origen de los ficheros estáticos
  + Y el más importante… Lista de aplicaciones instaladas.

Esta lista de aplicaciones instaladas es la más importante porque aquí definimos qué aplicaciones es capaz de correr nuestra aplicación. Aunque nosotros creemos una aplicación para gestionar cartas, si no la registramos aquí, no podremos trabajar con ella. Aquí una muestra de la configuración de aplicaciones del proyecto Appetito:



Entre ellas se encuentran los modelos de nuestra aplicación, las aplicaciones de gestión de restaurantes, y varias aplicaciones comunes para todos los proyectos de Django. A continuación, enumeraremos las aplicaciones de Appetito.

## Modelos

En Appetito existen estos modelos: **Perfil** (Usado para extender al usuario nativo de Django, que es un poco complicado de editar), **Plan, Dirección, Restaurante, Categoría, Plato y Tag.** Se encuentran gestionados en aplicaciones separadas de la siguiente manera:

* App Accounts: Modela los perfiles de los usuarios.
* App Carta: Modela categorías, platos y tags.
* App Plans: Modela las subscripciones.
* App Restaurante: Modela restaurantes y direcciones.

Además, y en virtud de las funcionalidades de Django, aparte de contener modelos, estas aplicaciones contienen un fichero llamado **forms.py.** En este fichero, y basándose en los modelos, se pueden crear formularios “prefabricados” que solamente deben mostrarse en las vistas y que estarán automáticamente linkeados a una instancia de un modelo por lo que el modelo solo servirá tanto de validador, como de fuente de datos, así como de receptor de datos, ahorrando muchísimo trabajo en creación de formularios.

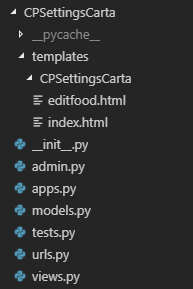
Como añadido, los ficheros de modelos no solo contienen la definición del modelo, sino que también pueden contener métodos o funciones que el modelo usará. Por ejemplo, el modelo de Dirección, tiene definida una función que recibe una distancia, latitud y longitud y comprueba si la latitud /longitud de ese modelo está a menos de esos KM de distancia.

## Aplicaciones

Para Appetito se han creado varias aplicaciones:

* **Pages**: Aplicación para gestionar las distintas páginas que se muestran.
* **Accounts**: Para gestionar la creación de cuentas y perfiles.
* **Cpanel**: Es el contenedor de las opciones para los restaurantes.
* **CPSettingsPlan**: Pantallas para gestionar las subscripciones.
* **CPSettingsRestaurante**: Pantallas para gestionar los restaurantes.
* **CPSettingsLocations**: Pantallas para gestionar las ubicaciones.
* **CPSettingsCarta**: Pantallas para gestionar las cartas.

Cada aplicación tiene una estructura similar a la siguiente:



Esta captura corresponde a la sección de Appetito que se encarga de gestionar las cartas.

Por orden aparecen:

* **\_\_pycache\_\_:** Directorio de caché de Python para agilizar la ejecución.
* **Templates**: directorio que almacena los distintos templates de la vista. Obligatoriamente en su interior debe existir un directorio con el mismo nombre de la app, y en el interior de este todos los templates, ya sean completos o parciales de cada uno.
* **\_\_init\_\_.py**: Archivo necesario para que python reconozca el directorio como una app ejecutable. Cualquier programa que hagamos con Python tendrá su \_\_init\_\_.py.
* **Admin.py**: Directorio de configuración de administración de los modelos. O para que nos entendamos. Si esta aplicación necesita un modelo (Carta, por ejemplo), en este fichero habilitaremos que el administrador de Django nos auto-genere un panel de edición para este modelo, así como también podemos editar las propiedades de este panel de edición (campos que muestra, tipos de formularios, límites, validadores, filtros, etc.)
* **Apps**.py: fichero que sirve para configuraciones internas de las apps (nombres, comportamientos, etc.) Este fichero de configuración es el que se carga en settings.py.
* **Models**.py: fichero que gestiona los modelos de la aplicación. En ellos se definen los campos de cada modelo, así como sus relaciones con otros. De este modelo parten las posteriores migraciones.
* **Tests**.py: fichero para test unitarios, no usados en este proyecto por falta de tiempo.
* **Urls**.py: Como he dicho, cada aplicación es independiente y se instala dentro del proyecto global. Esta naturaleza hace que cada aplicación pueda tener sus distintas urls para gestionar (carta, por ejemplo, tiene urls para crear platos, editarlos, crear y editar categorías, tags, etc.). Todas esas configuraciones de urls se realizan dentro de este fichero.
* **Views**.py: Aunque su nombre engaña un poco, este es el que hace las veces de controlador. Las urls de urls.py van asociadas a funciones definidas dentro de views.py, quien se encarga de hacer las operaciones y comprobaciones necesarias y al final, renderizar una vista, redireccionar, etc.

## Multimedia

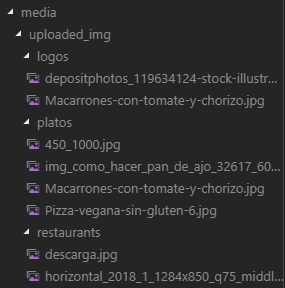
Los ficheros multimedia, en el caso de Appetito, hacen referencia a aquellas imágenes que forman parte del modelo de Plato y de Restaurante.

En la definición de esos modelos, se especifica que el directorio de carga de imágenes es uploaded\_img, y Django ya se encarga de mover y gestionar las imágenes.

En el modelo solamente se almacenará la URL relativa desde el directorio multimedia en el que se encuentra la imagen.

Volviendo al fichero settings, en el que se había configurado el origen de ficheros multimedia, ya podemos mostrar haciendo una concatenación de la URL Multimedia + URL del fichero, todas las imágenes necesarias.

En esta captura se muestra una previa de cómo quedan las imágenes organizadas en función del modelo que las cree.



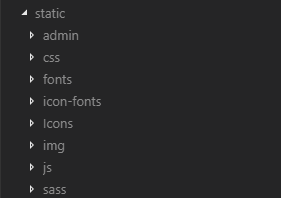
## Estáticos

Para un proyecto con Django, los ficheros estáticos son aquellos típicos de las cabeceras de las webs: hojas de estilo, scripts de JavaScript, etc.

Django presenta una funcionalidad muy peculiar. Nos permite tener en cada una de nuestras aplicaciones una serie de ficheros estáticos. Por ejemplo, para una app que trabaje con fechas, podríamos ponerle un JavaScript de datepicker sin que afecte al resto de la web.

Antes del despliegue de la aplicación, se usarán dos comandos de Python para “recolectar” todos los ficheros estáticos de las aplicaciones y situarlos todos en un mismo directorio para así agilizar su carga y edición.

Para la App de Appetito, en los estáticos se encuentran las librerías de Bootstrap, así como todas las imágenes de la web, librerías JavaScript, etc.



# Despliegue de la aplicación

## Requisitos previos

Previos al despliegue de la aplicación, necesitamos una máquina, ya sea física, virtual o rentada para poder instalar Python. Python sirve la aplicación el mismo por lo que no será necesario instalar un servidor web.

Actualmente Appetito funciona con una base de datos SQLite que se aloja en el mismo directorio, cosa que está totalmente desaconsejada.

Como me ha sido imposible encontrar una máquina gratuita que ejecute Python, lo realizo todo en local.

Antes de poder ejecutar la aplicación, será necesario tener instalado Python versión mínima 3.7. En los requisitos de software se especifican todos los paquetes necesarios para que la aplicación funcione.

Si es verdad que muchas de esas aplicaciones son de dependencia. Por ejemplo, al instalar Django, se instalan multitud de las demás de la lista.

Para instalar los paquetes, una vez teniendo Python instalado (Se incluye manual de instalación en “Recursos”, bastará con ejecutar el comando siguiente en la consola de comandos (en cualquier directorio del sistema, pero preferiblemente en el directorio de la aplicación) para instalarse.

|  |
| --- |
| pip install <nombredelpaquete> |

Como alternativa, podemos usar un entorno virtualizado.

Se adjunta con el proyecto, en el comprimido “appe.zip” un entorno virtual. Un entorno virtual no es nada más que un ejecutable de Python, con todas las dependencias instaladas que se puede descomprimir en cualquier sitio y ejecutarse sin necesidad de instalar ni Python ni los paquetes dependientes.

Una vez descomprimido, para activar el entorno virtual, habrá que entrar en la carpeta Scripts que se encuentra dentro y ejecutar el siguiente comando:

|  |
| --- |
| actívate appe |

Dicho comando ejecuta un script por lotes que automáticamente nos cambiará la consola a la consola de Python dentro de nuestro entorno virtual.

## Proceso de despliegue

Una vez que tengamos todo preparado, debemos realizar las siguientes operaciones:

### Fichero manage.py

El fichero manage.py es el intérprete de comandos de Python. Cualquier operación que queramos realizar sobre Appetito deberán apuntar a dicho script que será el encargado de por ejemplo arrancar el servidor, migrar la base de datos, etc.

En este punto, todos los comandos ejecutados se harán en el directorio principal del proyecto, que no es ni más ni menos el mismo directorio en el que se encuentra el fichero manage.py.

### Preparación de los ficheros estáticos

Debemos recolectar los ficheros estáticos ya mencionados que se pudieran encontrar repartidos entre los distintos directorios y aplicaciones que componen Appetito.

El comando para realizar dicha operación es el siguiente:

|  |
| --- |
| python manage.py collectstatic |

El comando agrupará todos los ficheros estáticos en la carpeta static allá donde nosotros la hayamos configurado en el fichero settings del proyecto (No es necesario tocarlo, ya viene configurad).

### Migración de la base de datos

El siguiente paso será preparar la base de datos. En un principio las migraciones ya van preparadas y no será tarea del equipo de despliegue crearlas, solo deberán lanzarlas. Pero en este caso realizaremos los dos pasos.

Lo primero que necesitamos hacer es crear los propios ficheros de migración. Para eso ejecutamos el siguiente comando:

|  |
| --- |
| python manage.py makemigrations |

Este comando recorrerá los distintos modelos y creará los ficheros SQL para construir la base de datos (es necesario que hayamos creado el fichero de base de datos. En ese mismo directorio, llamado db.sqlite3, que es como está configurado en el fichero settings).

Una vez creado los ficheros deberemos ejecutarlos. Para eso ejecutamos el siguiente comando:

|  |
| --- |
| python manage.py migrate |

Esto automáticamente volcará todos los SQL dentro de nuestra base de datos de sqlite.

### Creación el superusuario

Si no usamos la base de datos existente será necesario crear un superusuario para nuestra aplicación. Ese superusuario será el administrador del sitio y tendrá permisos para conectarse al administrador de Django.

Para crear el superusuario debemos ejecutar el siguiente comando desde el directorio raíz:

|  |
| --- |
| python manage.py createsuperuser |

Al ejecutarlo la consola nos pedirá un nombre de usuario, una contraseña y algunas informaciones. Las configuraremos al gusto y es importante no olvidarlas.

### Ejecución de la aplicación

Para ejecutar la aplicación, desde el fichero raíz, deberemos ejecutar el siguiente comando:

|  |
| --- |
| python manage.py runserver |

Automáticamente la aplicación arrancará y se pondrá a escuchar en el puerto 8000. Si queremos cambiar el puerto de escucha, deberemos ejecutar el siguiente comando:

|  |
| --- |
| python manage.py runserver IP:Puerto |

Donde los parámetros de IP y puerto son la IP y puerto pro el que queremos que escuche. Si no configuramos IP, podemos definir tan solo el puerto y escuchará desde todas.

### Configuraciones iniciales

La aplicación nada más empezar no tiene planes configurados, ya que queda a disposición de la persona encargada de gestionar la aplicación el crearlas a su gusto. Para ello, deberá acceder a la ruta del administrador de Django: ip:puerto/admin.

Nos conectamos con el superusuario creado anteriormente y buscamos el panel de edición de planes. Haciendo click en el botón de añadir, crearemos las que queramos al gusto.

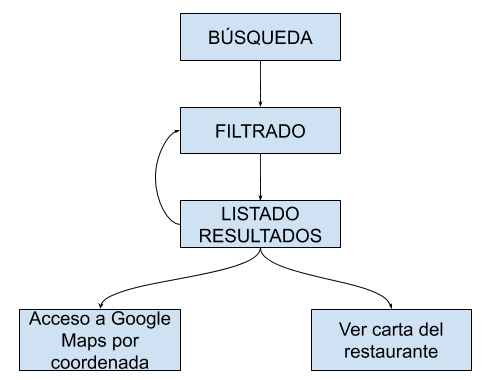
**Nota: para que un plan permita crear conceptos sin límites (ubicaciones, platos, categorías…) debemos configurar su valor a 0.**

# Manual de uso

## Uso por clientes

### Diagrama de uso por clientes

El uso básico por clientes es realizar búsquedas en la web con la posibilidad de aplicar filtros. Desde la lista de resultados, el cliente podrá filtrar de nuevo su búsqueda, o acceder tanto a Google Maps como a la carta del restaurante a partir de los resultados listados.



### Búsqueda y filtrado de resultados

A través de la barra superior de búsqueda o del panel lateral, el usuario podrá realizar búsquedas de productos.

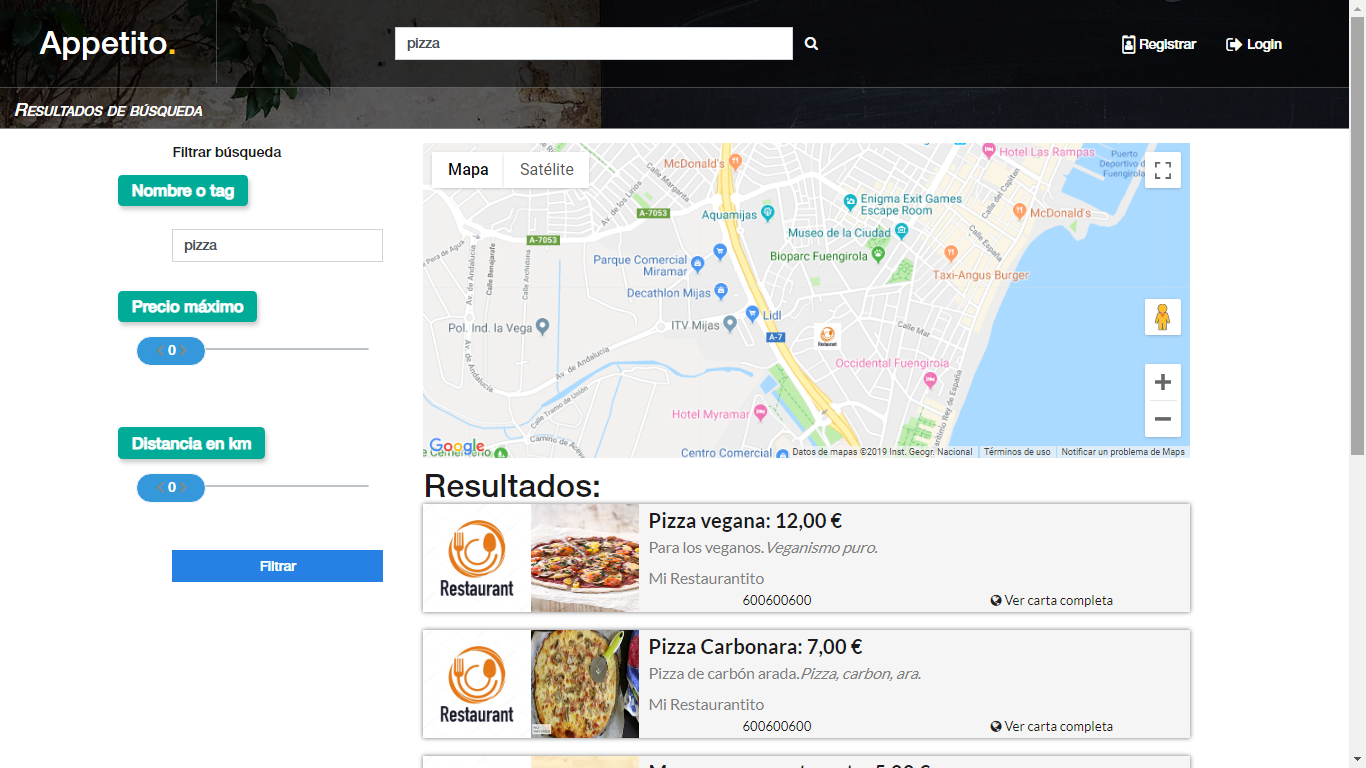
El texto escrito servirá para filtrar por nombre y tag.

Además, se puede añadir un filtro por precio máximo y distancia máxima de radio de búsqueda. En este caso, se toma como referencia el punto actual del cliente, obtenido a través de las coordenadas del navegador o del dispositivo móvil.

Si no se define distancia, se calculará con un radio por defecto de 10km.

Los resultados se mostrarán de dos formas.

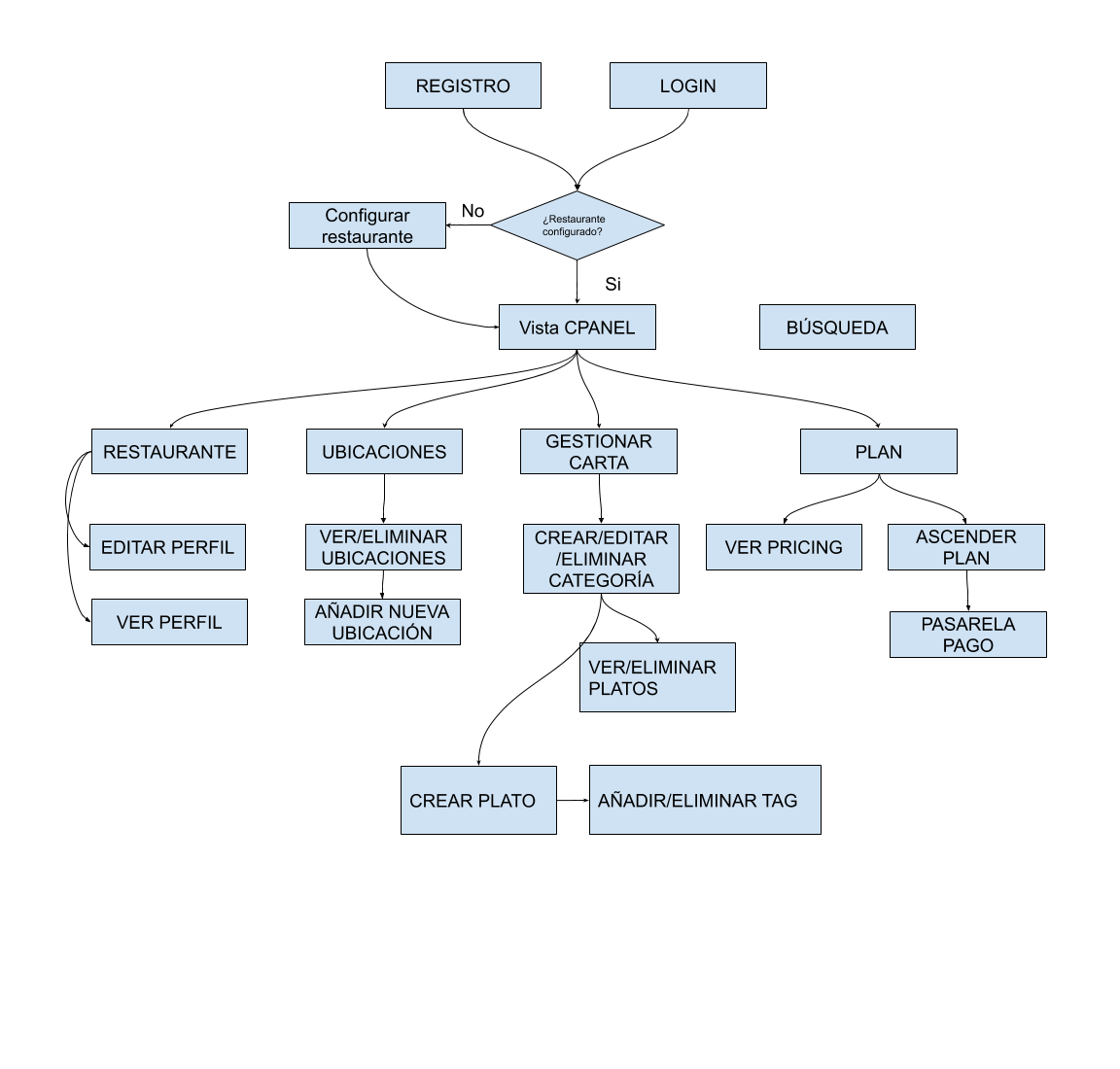
* Una lista compuesta por el icono del restaurante, una foto del plato, su información, y un botón de acceso a la carta del restaurante.
* Para cada restaurante distinto listado, en el Mapa se mostrará un icono al cual al darle click mandará a Google Maps a la ubicación del restaurante (Como mejora planteada, se contempla añadir un infowindow al marcador con información previa y un botón de Ir a la ubicación”, o “Cómo llegar”.



## Uso por restauradores

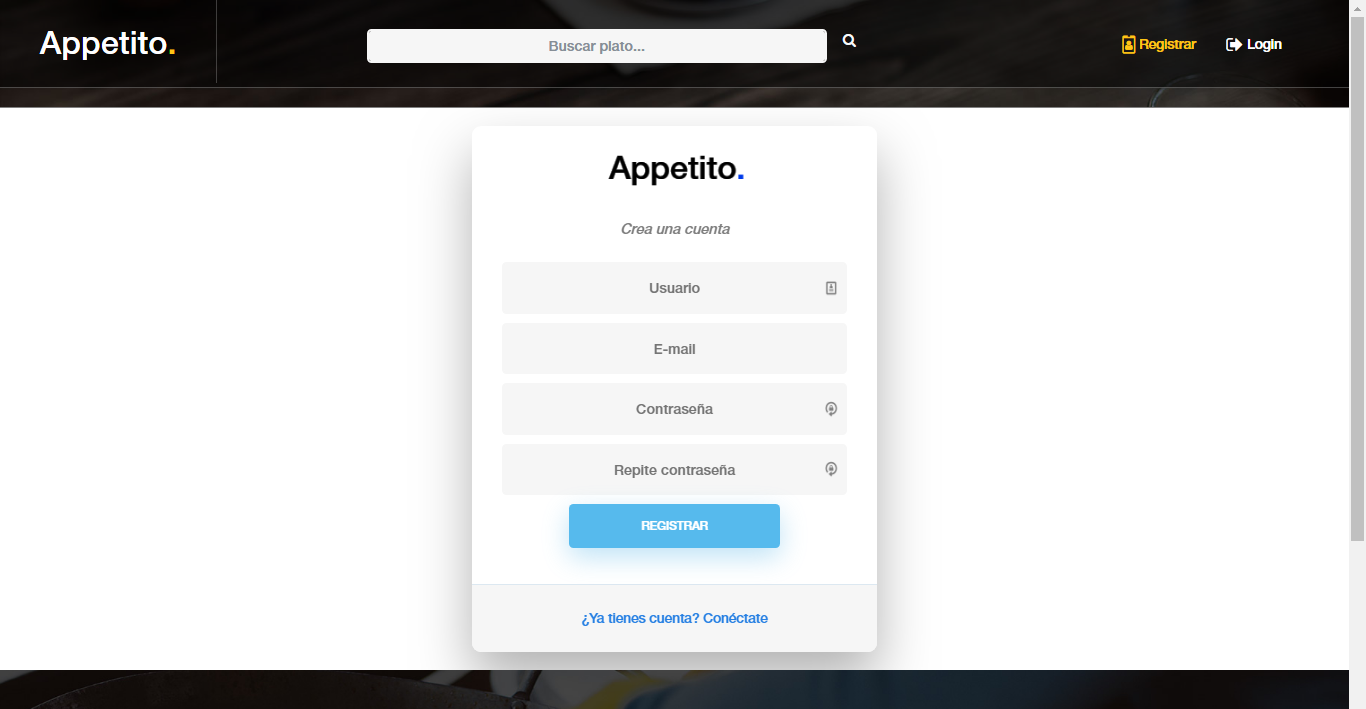
El restaurador, además de contar con la búsqueda y filtrado de platos, cuenta también con un panel de control para gestionar su cuenta.

Una vez que se registra y crea el perfil de su restaurante, tendrá acceso a paneles para editar todos sus parámetros: perfil, carta (categorías, platos, tags), ascender en su subscripción…

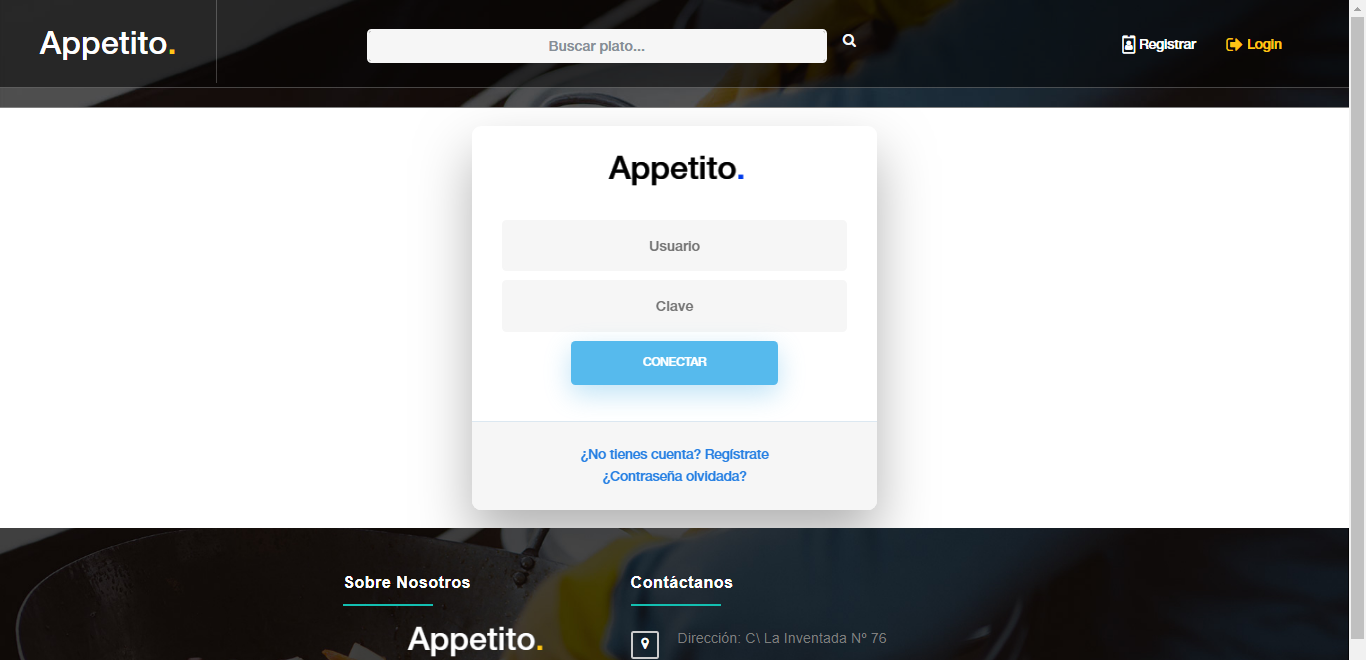
Se detalla en el diagrama de flujo todos los procesos que puede seguir el restaurador:  
  


### Registro

La primera pantalla funcional para el restaurador es la de registro. Se le pedirá que escriba las credenciales para su cuenta, para poder acceder al sistema. Basta con usuario, e-mail y contraseña repetida que debe coincidir.

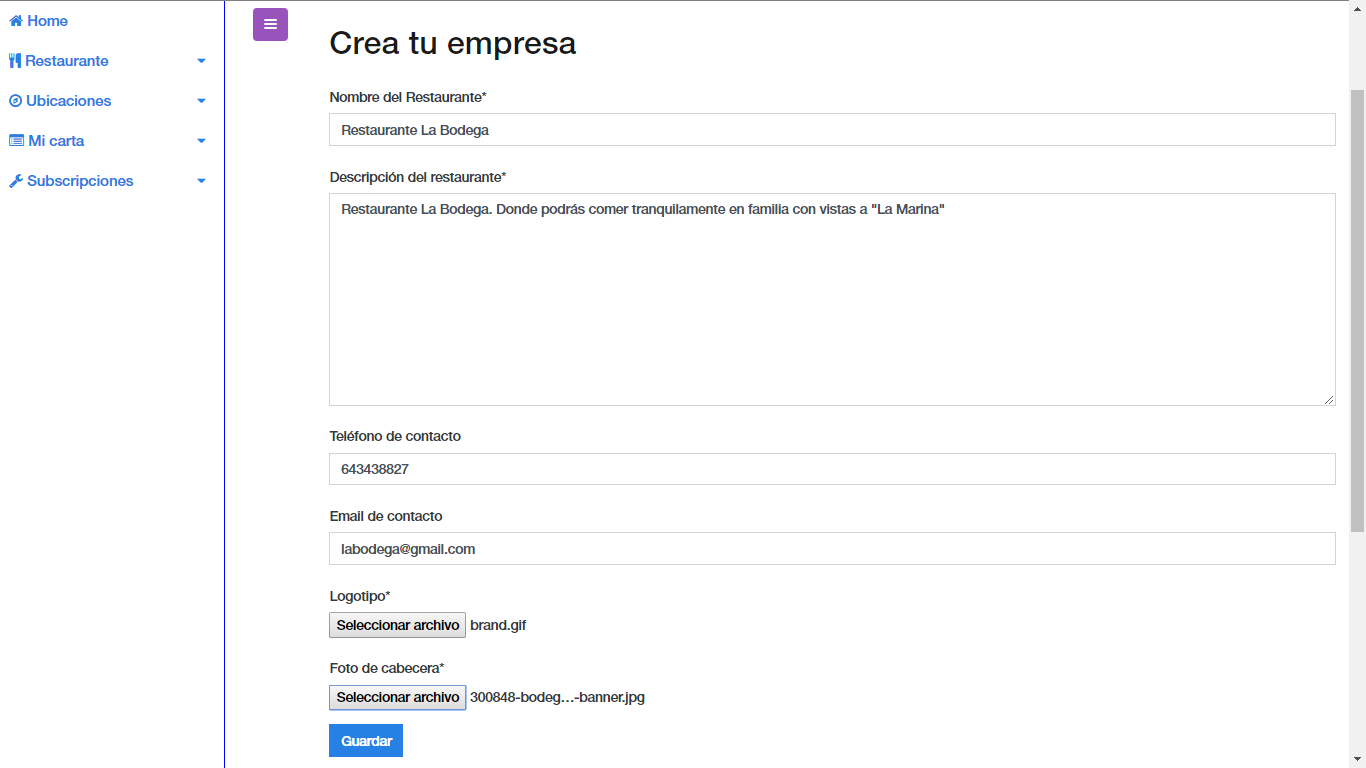
Una vez registrado, el sistema te logueará automáticamente.

### Logueo

Si el usuario ya tiene cuenta, puede acceder directamente a la pantalla de logueo. En la pantalla de logueo deberá escribir su usuario y su contraseña para entrar en el sistema.

### Creación de empresa

Automáticamente, ya sea el origen el registro inicial o el logueo, si aún no se ha configurado la empresa, la APP bloquea la navegación y te redirige siempre a la creación de tu perfil de empresa, ya que este es necesario para posteriormente poder editarlo.

En este panel introduciremos el nombre y descripción de la empresa, así como un logotipo representativo y una imagen banner en formato paisaje que servirá de cabecera para el perfil y la carta.

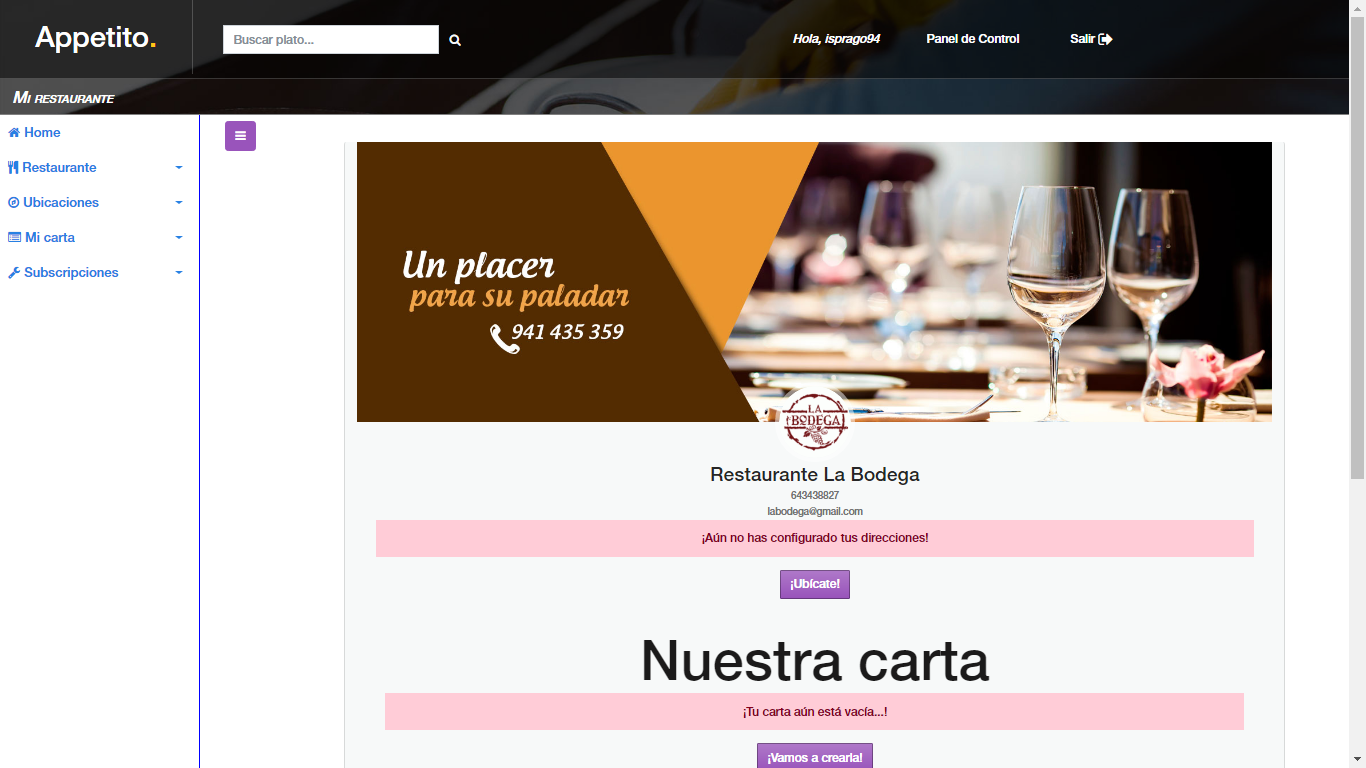
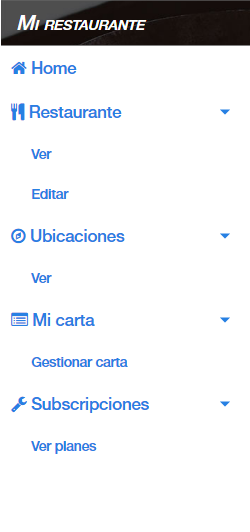
### Panel de control

Una vez creada la empresa ya podremos usar el panel de control para gestionar todas nuestras configuraciones.

En la vista central de la pantalla principal del panel del control tendremos un boceto de nuestro perfil en el que se nos informa de que podemos añadir ubicaciones, categorías, platos, etc.

Si volvemos a la vista principal del panel de control cuando ya tenemos configuradas categorías, platos y/o ubicaciones, automáticamente se nos mostrará en vez del botón que nos induce a crearlo las primeras veces.

Como se aprecia en la imagen, ya se están utilizando nuestra imagen de cabecera y de logotipo para ilustrar la vista previa de nuestra carta y de perfil de nuestro restaurante.



En el menú lateral tenemos el acceso a toda la gestión de nuestra cuenta. En orden sería:

* **Home**: Vista principal del panel de control.
* **Restaurante**: Gestión del perfil de restaurante.

**Ver**: vista previa del perfil.

**Editar**: Pantalla para editar el perfil del restaurante.

* **Ubicaciones**: gestión de ubicaciones.

**Ver**: Muestra lista de ubicaciones existentes y botón de borrar. Acceso a crear ubicaciones.

* **Mi carta**: Gestión de la carta.

**Gestionar carta**: pantalla de creación, edición y eliminación de categorías, platos, y tags.

* **Subscripciones**: gestión de la subscripción.

**Ver planes**: Permite ver el pricing de suscripciones y ascender de plan. Acceso a la pasarela de pago.

### Restaurante

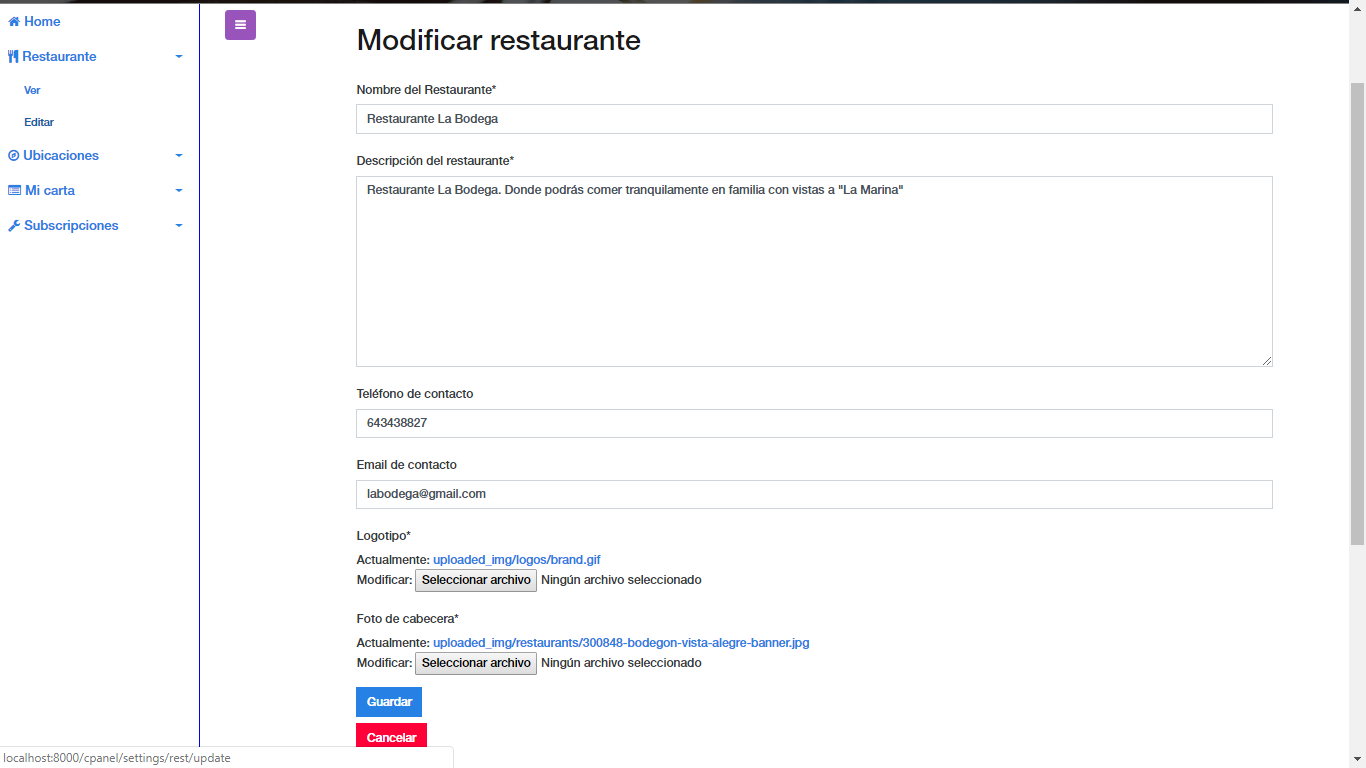
Desde el panel de control de Restaurante, tendremos acceso a dos funcionalidades.

#### Ver

Desde esta pantalla tenemos una vista previa de como verán los usuarios nuestro perfil y carta. Si no tenemos configuradas ubicaciones y/o categorías, nos mostrará botones para gestionarlas.

#### Editar

En esta pantalla se nos cargarán los datos actuales de nuestra empresa, datos que podremos editar. Si queremos ver vistas previas de las imágenes, dando click en los enlaces se nos mostrarán.

Una vez editadas, guardamos los datos. Si el formulario no pasa la validación, nos mostrará un mensaje de error. Si es correcto, habremos actualizado el restaurante.

### Carta

#### Gestionar carta

Desde gestión de carta podemos hacer las siguientes tareas.

Si nuestro plan lo permite y no hemos alcanzado el límite de categorías, se nos muestra un botón azul para crear categorías. Por cada categoría que creemos, nos aparecerá un panel con dos botones:

* Una papelera roja para eliminar la categoría y su contenido.
* Un lápiz azul para modificar el título de la categoría.

Bajo el nombre de la categoría, si nuestro plan lo permite y no hemos alcanzado el límite de platos por categorías, nos aparecerá un botón para añadir un plato a la categoría. Haciendo click podremos crear un plato. Debemos definir su nombre, descripción, lista de ingredientes, precio, y fotografía del plato.

Una vez guardemos, si el formulario es válido, habremos creado un plato en dicha categoría.

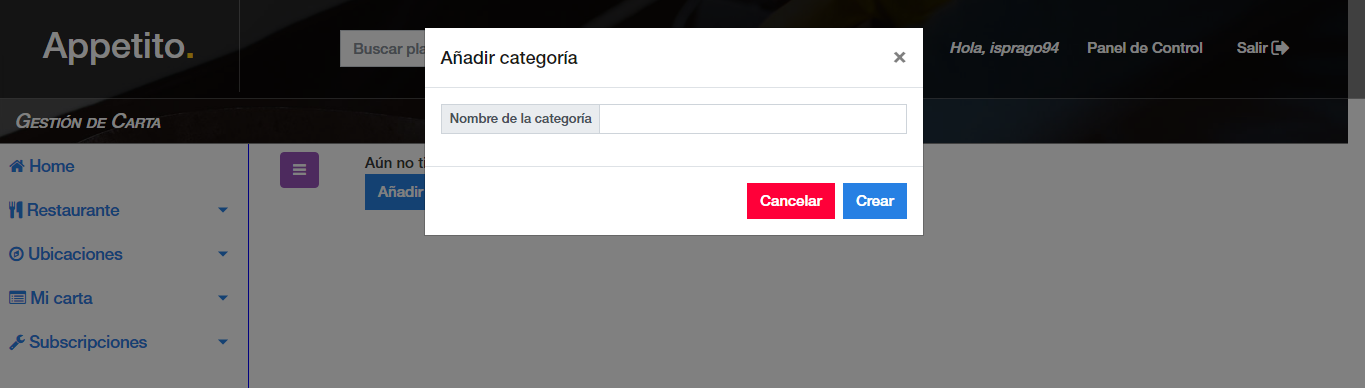
Por cada plato que tengamos en cada categoría, se nos mostrará un ítem en el bloque de la categoría. Dicho bloque tiene una previa de la foto del plato con su información. Además, aparecen dos botones.

* Una papelera roja para eliminar el plato.
* Un lápiz azul para modificar la información del plato.

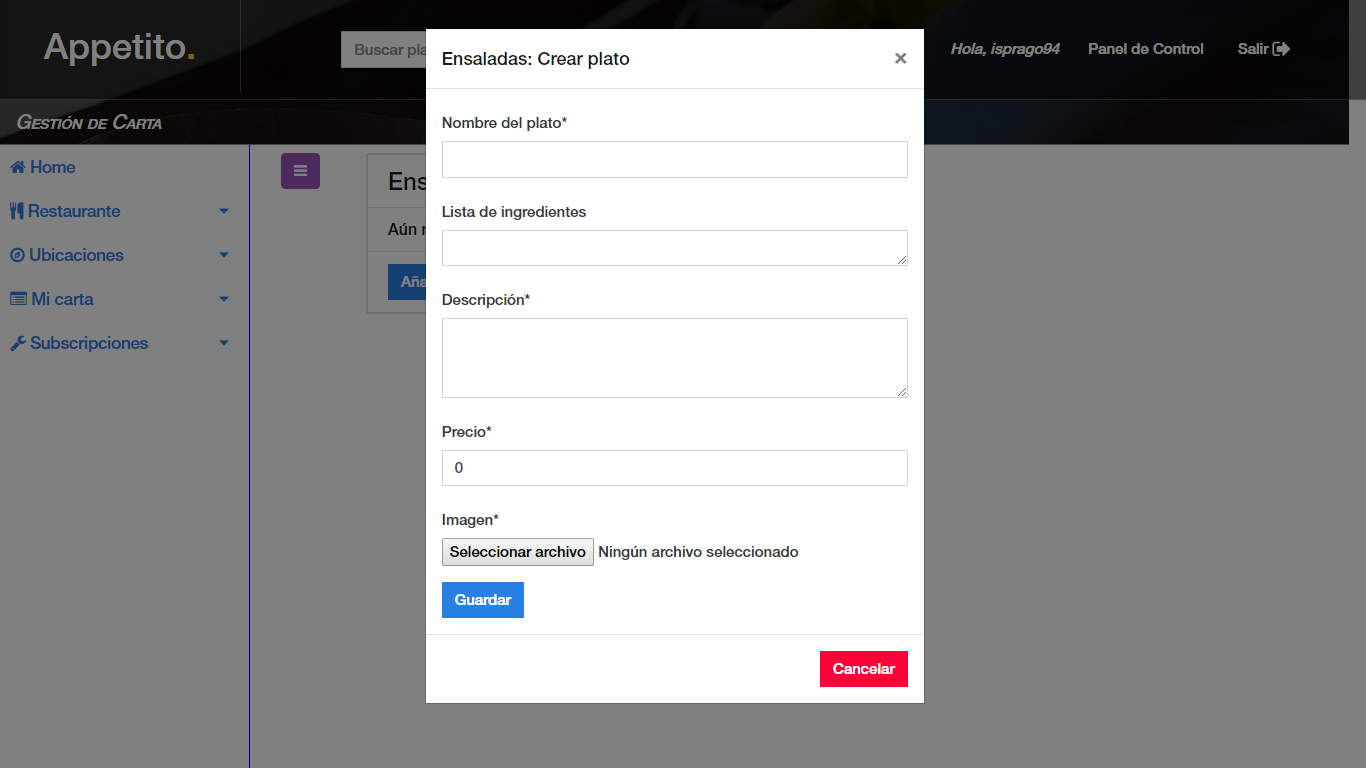
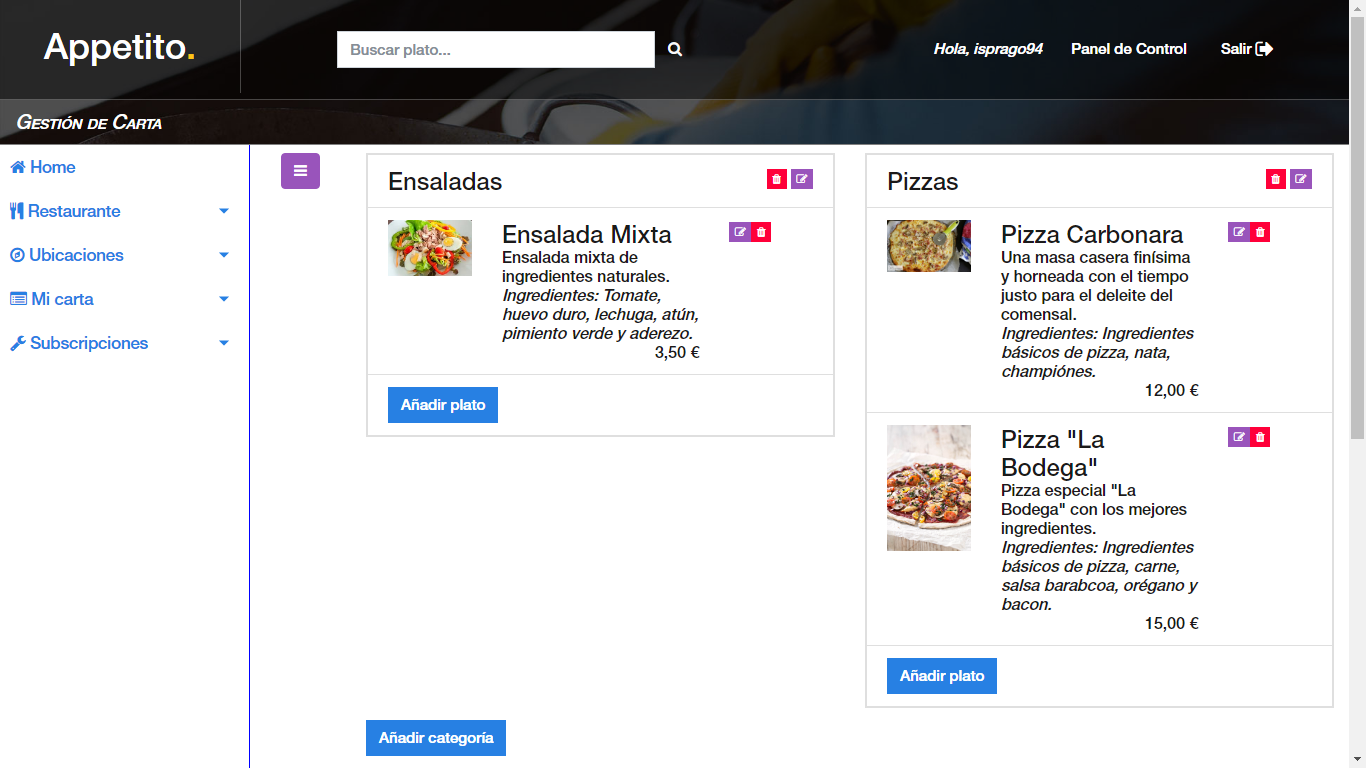
Haciendo click en el lápiz accederemos a la pantalla de edición del plato. Se nos muestra un formulario precargado con la información del plato. Al hacer click en guardar, si el formulario es válido, habremos editado el plato.

Además, a la derecha del formulario de edición se nos muestra un formulario para gestionar los tags.

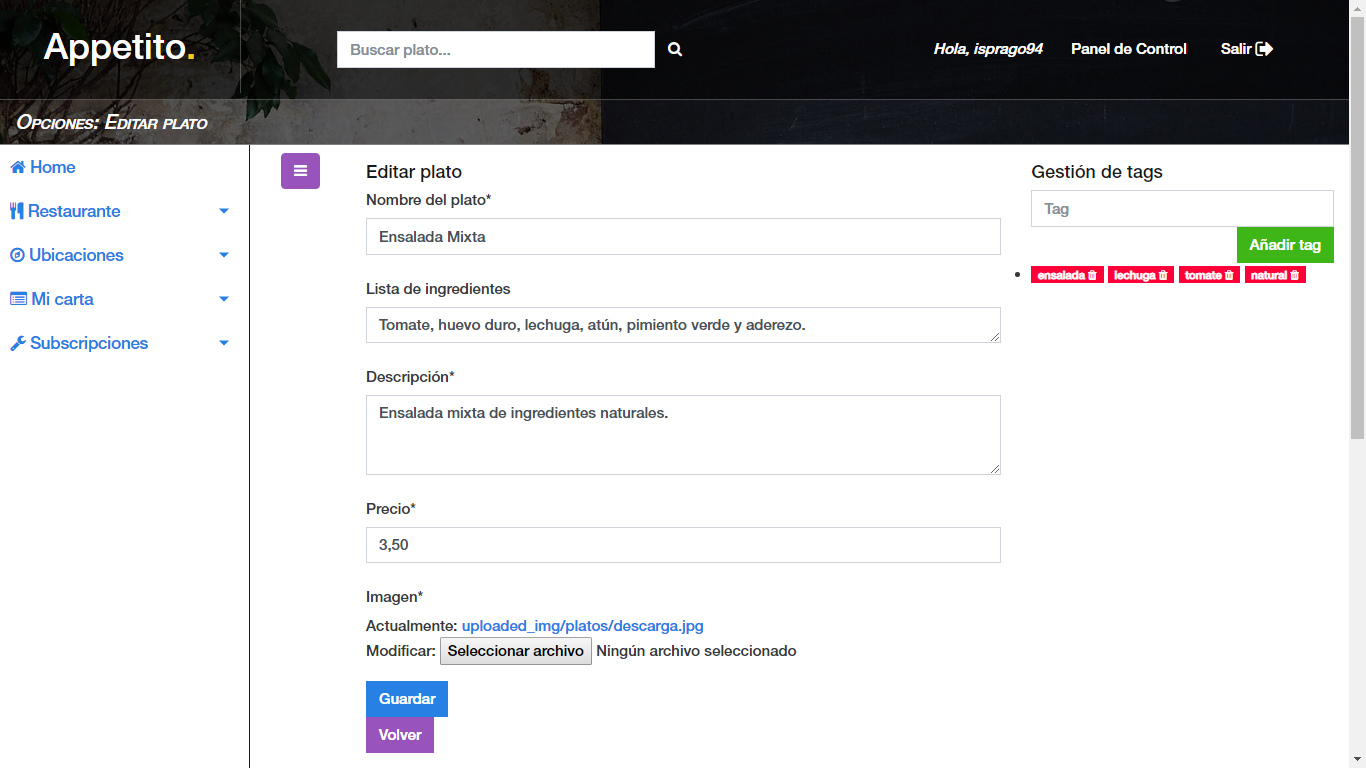
Si nuestro plan lo permite y no hemos alcanzado el límite, nos aparecerá un botón para añadir tags al plato que mejorará su aparición en las búsquedas.

Cada vez que generemos un tag, nos aparecerá en la lista inferior con un botón rojo para desasociarlo del plato.

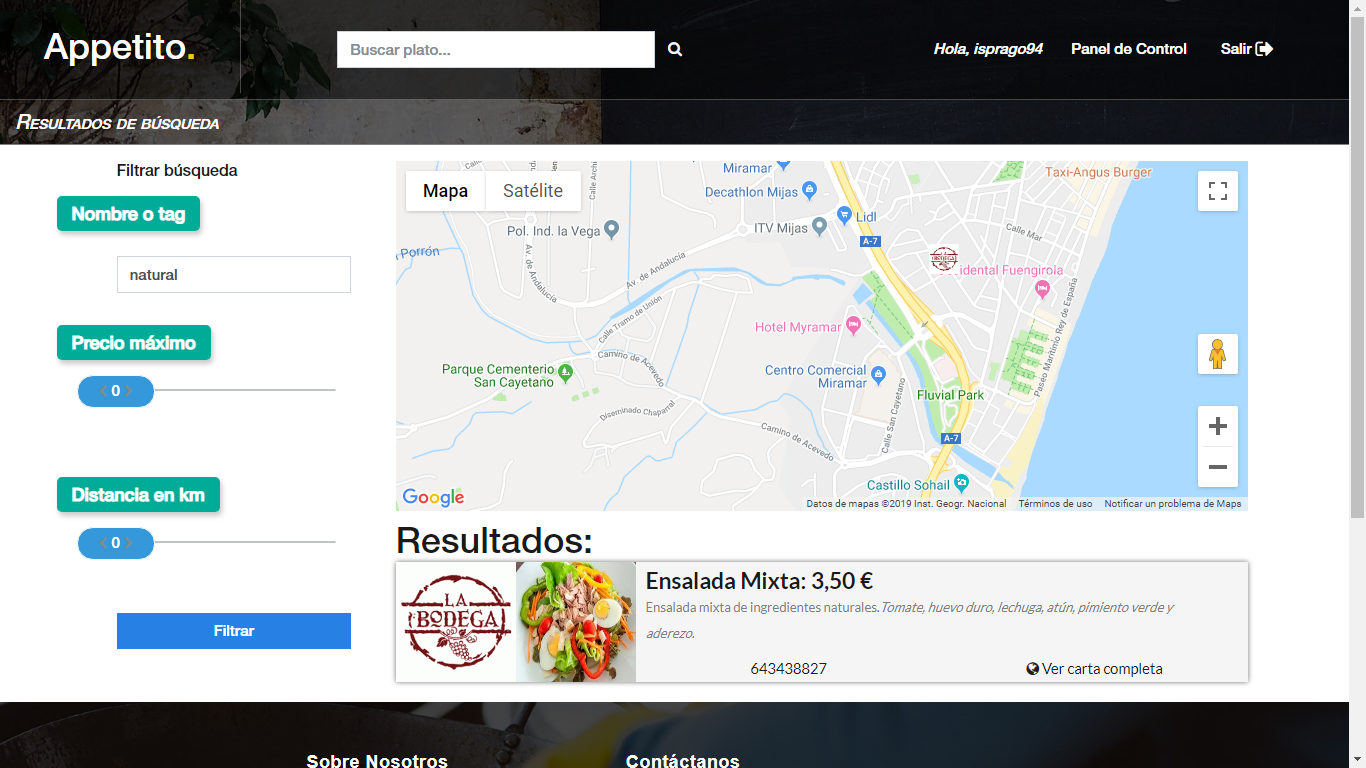
*Vista de Crear categoría.*

*Vista de Formulario de creación de plato.*

*Vista de gestión de carta completa.*

*Vista de edición de platos y gestión de tags.*

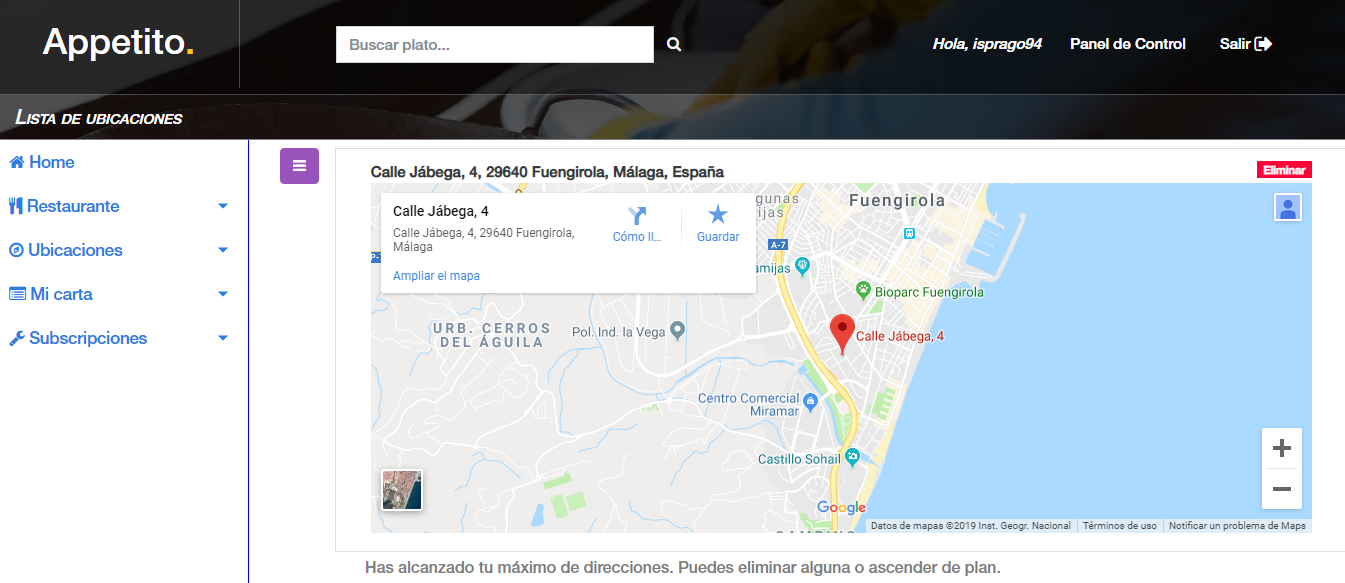
**

*Arriba:Detalle de Gestión de tags. Abajo: Muestra de resultados de búsqueda por tag “Natural”*

### Ubicaciones

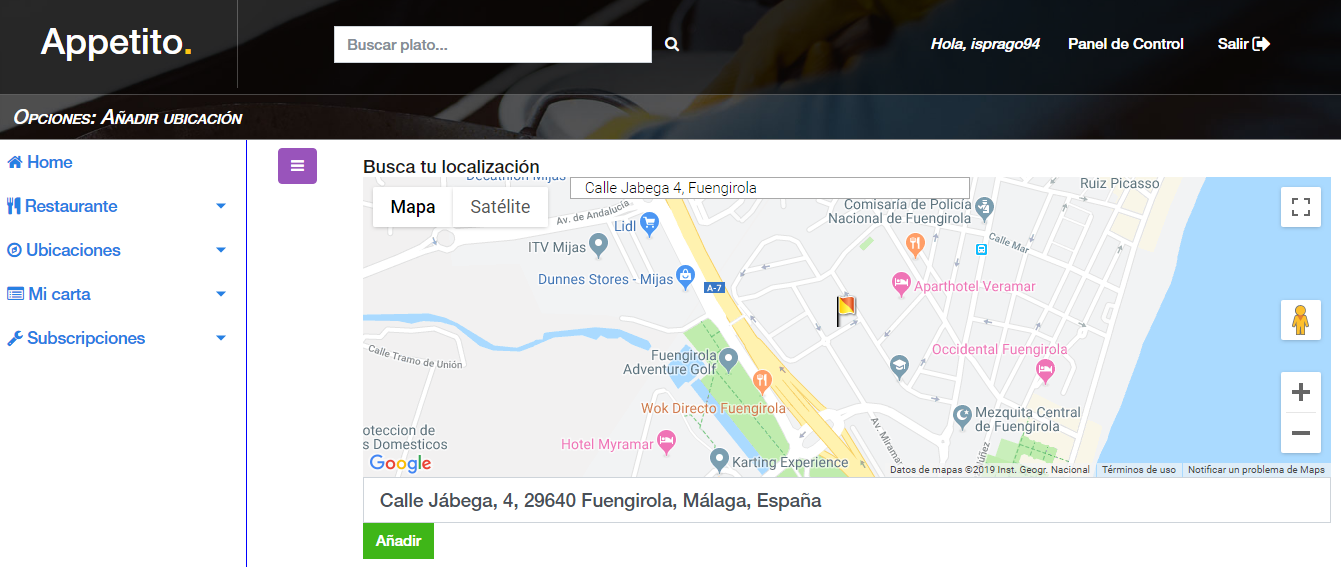
#### Ver

Desde esta vista tenemos una lista con todas las ubicaciones creadas para nuestro restaurante o cadena de restaurantes. Si el plan lo permite y no hemos alcanzado el máximo, nos aparecerá un botón para crear ubicaciones.

Además, por cada ubicación creada, nos aparecerá un bloque con la dirección física, un mapa que nos sitúa en el mapa dicha ubicación, y un botón para eliminarla.

*Vista de lista de localización.*

Si hacemos click sobre el botón de crear una ubicación, nos manda a una pantalla en la que tenemos un mapa con una barra de búsqueda en su interior.

Si hacemos click sobre el mapa automáticamente el sistema cargará la dirección deseada. SI queremos buscar una dirección concreta, debemos escribirla en la barra superior.

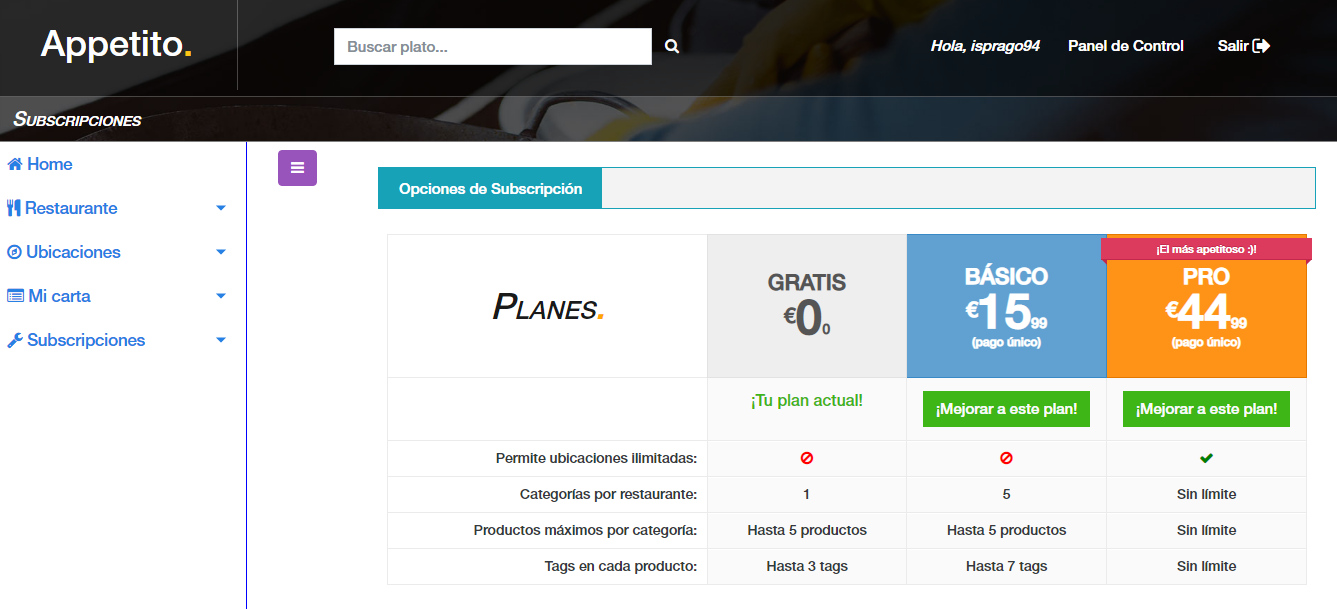
*Vista de creación de localización.*

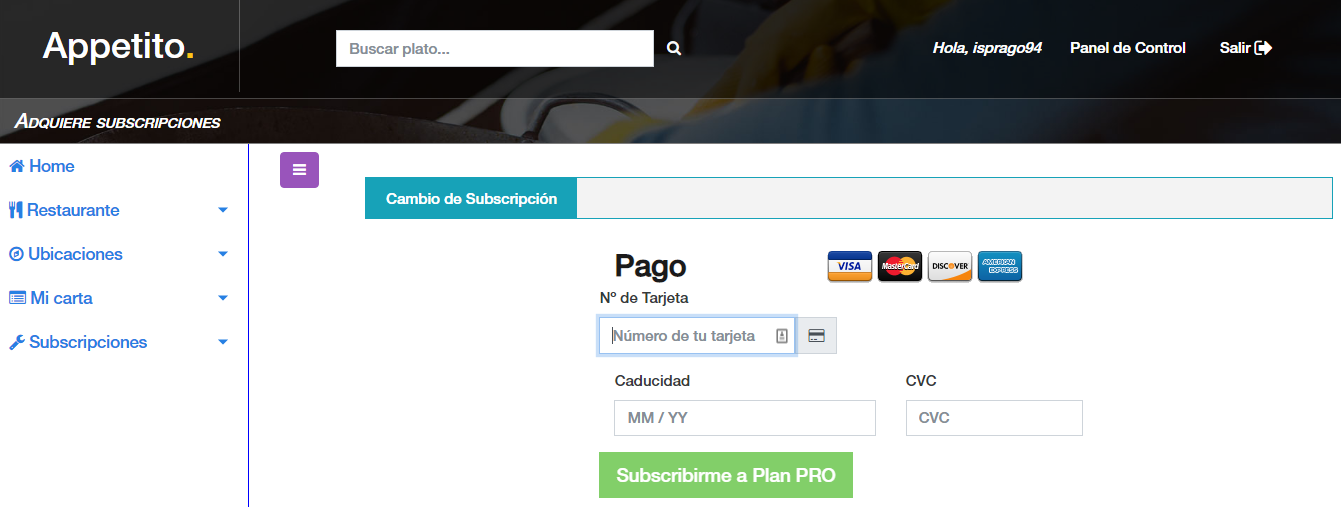
Cada vez que hagamos click, el nombre de la dirección aparece en la barra inferior. Ese será el nombre descriptivo de nuestra ubicación y se recomienda no definirlo manualmente.

### Plan

#### Gestionar subscripción

En esta vista tendremos por defecto una tabla con una vista del pricing de las subscripciones. Se nos muestran todas las que existen con sus precios y límites.

Si nuestro plan es inferior al listado, nos mostrará la tabla un botón que nos permite “Ascender” al plan. Al hacer click en él, nos manda a una pasarela de pago.

Si escribimos unos datos válidos de una tarjeta de crédito, nos cargará el coste del plan y nos ascenderá, cambiando nuestros límites de creación.

*Vista de pasarela de pago.*

## Uso por administrador

### Administración de Django

Django trae consigo un completo administrador para su sitio web. Considero que es completamente innecesario el explicar aquí su funcionamiento ya que se incluyen los enlaces a sus manuales en los recursos.

Solo destacar que sería necesario al inicio del sistema, si no se han cargado los planes por defecto que vienen en el fichero **planes.sql** del directorio de recursos, usar dicho administrador para dar de altas a los primeros usuarios.

Recuerda que para loguearte con el administrador de Django será necesario usar las credenciales del superusuario que creamos en este manual.

# Posibles mejoras

## Mejoras a corto plazo

* Completar la funcionalidad de descarga de cartas en PDF para que los restauradores puedan usar el editor de cartas como fuente para las cartas en sus restaurantes.
* Posibilidad de realizar comentarios anónimos sobre los platos/restaurantes, así como rating para mejorar los filtros y la experiencia del usuario.
* Mejorar el filtro de distancia para que te sugiera rutas sin necesidad de acudir a Google Maps.
* Integrar la pasarela de pago con el sistema STRIPE para que los cobros sean reales aunque sean con una tarjeta de demostración.
* Completar la exportación de cartas en PDF.

## Mejoras a largo plazo

* Incrementar la funcionalidad de las Subscripciones permitiendo a los restaurantes “anunciar” sus productos.
* Incrementar la funcionalidad de las Subscripciones permitiendo a los restaurantes crear cupones de descuentos y ventajas.
* Editar el widget de mapas para que al hacer click en un icono de restaurante, te muestre una ventana de información completa con sus datos de contacto, direcciones, comentarios, ratings, etc.
* Cuentas para clientes. Permitir a los clientes crear sus propias cuentas y poder gestionar sus favoritos, almacenar cupones, descuentos, etc.
* Versión móvil: lanzamiento de la versión móvil, descartada en este desarrollo por falta de tiempo.

Para la mejora de clientes se plantea la siguiente solución en la base de datos:

### Especificaciones de mejora de clientes

* Los **clientes** crearán una cuenta desde el móvil. Meterán un **nick, e-mail, clave y foto de perfil** (o se usará una por defecto). Automáticamente se guardará la **fecha de creación**.
* Los clientes realizan búsquedas y se les mostrará la lista de restaurantes/productos coincidentes.
* En los restaurantes se mostrarán algunos de los comentarios dejados en sus productos.
* Sobre los **productos hay comentarios escritos por los usuarios** en determinada **fecha**.
* La valoración del restaurante será la media de las valoraciones de sus productos.
* Cada **cliente puede dejar una única valoración sobre un producto** entre 1 y 5. Esa valoración puede ser modificable.
* En las vistas se muestra la **media del restaurante** (media de valoración sobre restaurantes) y una **media de productos** (media de la media de los productos). Además, los **productos tendrán su propia media** dependiendo de los votos de los usuarios.
* El **cliente tendrá una lista de restaurantes y productos** **favoritos** para facilitar su búsqueda.

# Bibliografía

|  |  |
| --- | --- |
| Paquetes de Django – Búsqueda de paquetes de Django útiles para Apeptito.  <https://djangopackages.org/> | Generador de CÃ³digos QR Codes |
| Sitio oficial de Django – Documentación general sobre modelos, estructuras de proyectoa y demás funcionalidades.  <https://www.djangoproject.com/> | Generador de CÃ³digos QR Codes |
| Curso Udemy Django – Realización de 3 webs que inician a la estructura de Django.  <https://www.udemy.com/curso-django-2-practico-desarrollo-web-python-3/> | Generador de CÃ³digos QR Codes |
| Sitio oficial de la API JS Google Maps – Documentación sobre como cargar mapas en JS y realizar búsquedas sobre el.  <https://developers.google.com/maps/documentation/javascript/tutorial> | Generador de CÃ³digos QR Codes |
| Cursos de Django Básico  <https://www.udemy.com/curso-basico-de-django-paginas-web-con-python/> | Generador de CÃ³digos QR Codes |
| Curso maestro Python desde 0 – Iniciación a la programación con Python. Conceptos básicos  <https://www.udemy.com/course/python-3-al-completo-desde-cero/> | Generador de CÃ³digos QR Codes |

# Dependencias

Appetito usa dos librearías para generar los PDF. Son una composición de GTK (Gimp Toolkit) y de Cairo Graphics. Para que el equipo en el que esté instalado el sistema pueda generar pdfs, será necesario instalar ambas aplicaciones.

En la sección de Recursos se encuentran los descargables.

# Recursos

|  |  |
| --- | --- |
| Descarga de Python 3.7  <https://www.python.org/downloads/> | Generador de CÃ³digos QR Codes |
| Guía de instalación de Python  <https://realpython.com/installing-python/> | Generador de CÃ³digos QR Codes |
| GTK  <https://www.gtk.org/download/> | Generador de CÃ³digos QR Codes |
| Cairo Graphics  <https://cairographics.org/download/> | Generador de CÃ³digos QR Codes |

# Conclusiones

## Conclusiones sobre el proyecto

### Durante el curso

Sin ser sorpresa, la situación previa al proyecto es algo desconcertante. Desde las primeras semanas, sin conocer las tecnologías que te van a enseñar durante el curso, los lenguajes, tu habilidad con ellas, debes decidir una idea básica de en qué va a consistir. Si que es verdad que cualquier cosa es desarrollable en cualquier lenguaje, pero en una etapa en la que, incluso teniendo conocimientos previos, somos novatos, puede que adquiramos más habilidades al desarrollar con unos lenguajes que con otros, y que haya ciertas ideas que sean más fáciles de desarrollar con esos lenguajes (No es lo mismo una aplicación que necesita API a una que no, igual que no es lo mismo un framework de front que uno fullstack).

El mensaje de “Pensad en el proyecto” se queda en el aire al principio, sin seguimiento, sin apoyo y, al final, si o si se va a convertir en una piedra angular que cambie el ritmo y finalidad del curso.

Poniéndome en la piel de otros, o incluso en la mía, que tuve que cambiar el proyecto nada más empezar, es verdad que desconocemos el ritmo de las prácticas y lo que estas llegan a influir en el mismo desarrollo del proyecto.

### Pre proyecto

El momento previo al proyecto siempre es desconcertante. Te empiezan a exigir que sigas un modelo de desarrollo cuando nunca nadie te ha explicado cómo se hace un desarrollo. Empecé por lo que para mí es obvio, la base de datos. SI no tienes un modelo de datos que sepas que funciona, al final acabaras dándote con el programa en la frente cada vez que tengas que desarrollar una funcionalidad.

Por suerte y del anterior ciclo pues considero que tenía buenas técnicas de construcción de modelos relacionales y de bases de datos, y la verdad es que no he tenido que hacer modificaciones funcionales a la base de datos, más bien, he tenido que adaptarme a problemas inesperados como que el modelo de usuario de Django es un modelo cerrado y difícil de editar, y que he tenido que eliminar funcionalidades por falta de tiempo.

Parto con el proyecto bien definido, las ideas claras, un modelo de datos funcional, y con la idea de que quiero una web con Angular6, una app con Ionic y una API con Laravel.

### Durante el proyecto

Lo que no te esperas es lo que vas a hacer en las prácticas. Cada uno distinto, pero es verdad que el horario influye mucho, incluso las situaciones personales. Tardando días hasta 2h en hacer una carretera de media hora, llegas a casa a las 7 después de haberte levantado a las 7 y estar 3 horas de carretera y 8 de trabajo y tienes que ponerte con el proyecto.

Para empezar, en la empresa nos cambiaron la tecnología del tirón. Tocaba aprender Python, y después pedimos aprender Django para saber hacer webs en Python para cuando tuviéramos que hacer tareas para la empresa.

Tienes que dejar el proyecto parado primero por falta de tiempo y después por el cambio de chip. Por muy bien que haya estado el curso es imposible acordarse de memoria de todas las funcionalidades y más cuando pretendí enfrentarme a 3 frameworks distintos.

Me era imposible estar en la empresa con dos tecnologías nuevas y llegar a casa y estar con otras 3. Y más cuando por mi situación personal (vivo de alquiler a 100km de casa) dependo de mí mismo para tener algo que cenar, que comer al día siguiente, compras papeleo… Habiendo días que literalmente si tienes algo que hacer, no puedes ni dedicarle 1h al proyecto. Pero bueno, de todo se sale. Aproveché las vacaciones de Semana Santa para darle un tirón (gracias a que pudimos decidir no ir a la empresa durante esa semana, que creo que debería ser completamente obligatorio para todos los alumnos en prácticas para que tengan un tiempo garantizado para sus proyectos.

Con respecto a los problemas con el proyecto, en cuanto al framework y lenguaje prácticamente ninguno. Los cursos que están en bibliografía son suficientes para aprender y sacar un trabajo bastante decente.

Donde más tiempo he podido perder ha sido intentando encontrar la manera de poder usar el modelo de usuario de Django para mi sistema, que al final fue tan fácil de solucionar como hacer una relación 1:1 en la base de datos entre USER y PROFILE (Creando el modelo PROFILE y definiéndole una relación 1:1 con el auth.user que es el modelo nativo que gestiona y modela a los usuarios en Django)

Otra cosa en la que sí que perdí hasta una semana fue en la gestión del mapa con Google. Google cuenta con más de 5 herramientas destinadas a mapas, y tienes que activar la key con todas para que cada una te de distinta funcionalidad: una para que te funcione el mapa en JS, otra para que puedas geolocalizar con coordenadas, otra para que puedas localizar con direcciones literales…

Después de perder una semana empezó a funcionar y por lo demás estupendo.

### Finalización

La verdad es que estoy satisfecho e insatisfecho a la vez. Por una parte he podido realizar una aplicación completa y funcional, que tiene una intención y que la cumple plenamente, en mayor o menor aspecto o con mayor o menor funcionalidad. Pero no olvidemos que estamos un proyecto evaluable sobre 40 horas. Todo lo añadido sobre las 40 horas son esas famosas “mejoras” que ya lleva incluidas, porque un proyecto de 40 horas puede ser un simple panel de registro y listado de usuarios, una herramienta de gestión de menús sin búsqueda ni tanta funcionalidad….

La parte con la que no estoy satisfecho es primero con no haber podido hacer todo lo que quería, pero la razón para no haberlo hecho es más que justa y razonable: falta de tiempo. Las horas de prácticas y demás, hubiera sido imposible. Al igual que es imposible a la hora de decidir el proyecto cual va a ser su envergadura y si esa es correcta o no.

Appetito funciona. Hace lo que se propuso. Le faltan funcionalidades, funcionalidades que no afectan al funcionamiento básico y que son suficientes, desde mi punto de vista, para que se pudiera convertir en una primera beta de una idea comercial.

## Conclusiones personales

A pesar del gesto que agradezco de haber tenido una semana dedicada al proyecto que al final fue más bien una semana de toma de aire, quizás la experiencia piloto no fue muy satisfactoria. Personalmente no creo que haya que sacrificar nada del curso para hacer hueco al proyecto, pero, y teniendo en cuenta tanto a los que tienen más capacidad de aprendizaje como a los que menos, un ritmo acelerado de clases, de contenido denso, de distintas materias, o incluso horas muertas semanales, acaban quemando mentalmente.

Por suerte, y al menos en mi proyecto, pude empezar a organizarme, ya que experiencias previas me indicaban lo que se me venía encima. No sé si todo el mundo llego a tener en serio algo antes de empezar, al menos la estructura de la base de datos revisada y funcional, pero yo lo conseguí, y aunque después la modificara, creo que es mucho más fácil deshacer o eliminar cosas innecesarias que ir construyendo al vuelo.

Después está el dilema de las 40 horas. Nunca son 40, y el problema es que ya se anuncia que se evaluará sin tenerlo en cuenta. Que hay que echarle horas al proyecto, pero que, si no alcanzas unos mínimos, te verás penado. Leyes, normas, que al final solo perjudican.

Quizás no se perciba, quizás la vista sea de un puñado de niños pequeños enfadados, pero realmente creo que se ignora que al final acabamos siendo personas con frustraciones, ya sea porque no alcanzaremos nuestra meta, o porque ni siquiera sabemos dónde debemos poner el listón de meta.

El proyecto se inicia a ciegas, sin nada, sin saber de dónde partes ni a donde llegas, por una falta muy fuerte de organización. Organización que se consigue guiando al alumno inexperto, y el alumno inexperto poniendo de su parte y mostrando al menos, interés.

Y saltando de ahí a prácticamente el final… Recuerdo palabras de algunos profesores de que durante las prácticas se perdía el feedback. Pues que conste que no es por intención nuestra ni mucho menos, y hablo en plural. Porque en ciertos momentos me he quedado a la espera de llamadas de atención. Vale que tengamos que ser independientes, y saber gestionarnos y todo lo demás, pero después llegamos a clase y vienen las ideas locas a los profesores que pretenden que metamos en el proyecto cuando ya está a medio construir. Y hasta el final de las prácticas, y sobre todo en los momentos más cruciales, cuando algunos ya estamos contratados y trabajando, se intensifican las revisiones.

Obviamente la empresa entenderá que aun somos estudiantes, pero no es algo que sea agradable para nosotros tener que estar desde el primer día pidiendo favores, y mucho menos que otras personas los pidan por nosotros.

Tanta ha sido la presión, el estrés por poder entregar algo sin tener tiempo, que sin tapujo alguno digo que he pensado en no entregar el proyecto en junio, incluso cuando me comunicaron la noticia de que estaba contratado pensé en directamente no entregar el proyecto, porque se convierte en algo innecesario el tener que estar bajo esa presión cuando ya has alcanzado la meta de este ciclo que creo que es aprender a programar y conseguir trabajo (Que no tener el titulo no te ha quitado de ninguna de las dos, y no entregar el proyecto no significa no saber programar)

La respuesta obvia es que ni se nos ocurra pensar eso, que no sabemos lo que decimos. Creo que a ninguno de nosotros como alumnos y después de dos o más años de curso, algunos yendo andando a clase y otros que (y esto son circunstancias personales) hemos tenido que invertir en un alquiler, se nos ocurriría tirar todo por la borda. Pero llega tal momento en el que al final del camino empiezan las exigencias por que durante el camino no se han hecho, no se han comunicado ni se ha seguido, que sinceramente se nos quitan las ganas.

Y a estas alturas puedo garantizar que el 90% de nosotros ya no busca el triunfar con el proyecto a buscar la satisfacción de un 5. Una nota suficiente.

Personalmente creo que algo falla si, y sobre todo con nosotros que considero que éramos un buen grupo, todos acabamos coincidiendo en que nos conformamos con un 5 para quitarnos el proyecto y el curso de encima.

Al fin y al cabo, tanto profesores como alumnos somos personas, profesionales de la enseñanza o del aprendizaje, cada uno con nuestras circunstancias y supongo que todo influye, pero no me agradaría que siempre fuera así, con ese final.

De acuerdo, todos sabemos objetar, pero no dar solución. No la puedo dar porque no la conozco y entiendo que es duro para el equipo docente recibir las críticas cuando casi todos se dedican a hacer su trabajo lo mejor posible y en beneficio de todos.

Curso superado, nivel superado. Algo ha ido bien si realmente un gran porcentaje de nosotros ha acabado con trabajo y eso es lo que hay que celebrar, pero que no quede atrás el revisar por qué un curso que empezó con ganas y muy motivado haya acabado prácticamente abatido y a disgustos, y no se conforme el asunto con un “es lo normal”.