DOCUMENTACIÓN EXTERNA: RECETICAS

Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

Administración de Tecnologías de Información

Diseño de Software

Prof. Andrei Fuentes Leiva

Integrantes:

Benjamín Calvo de León

Esteban Mora Soto

Diego Vásquez Valerio

Primer semestre 2014

Tabla de contenido

Introducción.

El conocimiento culinario es cada vez más escaso en el mundo. Más y más personas prefieren la comida chatarra a una buena comida en casa, haciendo cada vez menos la popularidad de los platos de tradición familiar y cultural, impulsando todo tipo de problemas; por lo que es un tema crítico encontrar una forma de promulgar estas artes por un bien cultural y personal.

Resumen ejecutivo.

El proyecto Receticas (www.receticas.biz/login) tiene como objetivo presentar una solución a nivel de aplicación web la cual se centra en la presentación de información sobre recetas de distintas categorías y tiempos de comida, donde los visitantes del sitio pueden ver las recetas que están publicadas en la misma. Otra característica para los lectores es tener la posibilidad de suscribirse a los autores de las recetas y obtener una línea del tiempo personalizada con las recetas de los autores a los que están suscritos.

Como último elemento para los visitantes, los mismos pueden dar referencias de la receta, mediante comentarios o señales de aprobación.

Otra posibilidad para el usuario es el tener la facilidad de acceder al sitio y redactar sus propias recetas, dichas recetas pueden recolectarse para crear uno o varios libros que pertenezcan al autor de las recetas, el o los libros que se crean pueden estar organizados por categorías o por tiempos de comida para facilitar la organización de las recetas.

Las recetas al ser escritas tendrán las características generales de una receta en libro impreso: nombre de la receta, categoría, tiempo de comida, porciones que permite la receta, ingredientes, medidas de cada ingrediente, marca del ingrediente, pasos e indicaciones para preparar la receta (los mismos pueden contener imágenes). Las recetas además de tener lo anterior tendrán el precio de los ingredientes, esto para generar un precio general de la receta. Además, los editores pueden alterar la receta luego de creada.

Justificación.

El proyecto se justifica en base a la idea de realizar un servicio el cual permita integrar elementos de información en base al tema culinario; con el fin de conocer y compartir sobre diferentes tipos de cocina y variedad de recetas, así como las variaciones de una misma receta aportadas por otros usuarios; siempre tomando en cuenta que nuestro público meta son diferentes tipos de personas, desde cocineros expertos hasta aficionados, amantes de la cocina o personas que buscan una receta para cocinar algo diferente.

Especificación de requerimientos.

Requerimientos de lectores.

RQL-01	Visualización de información de una receta.
Funcional	Toda receta debe mostrar la información general de la misma: Nombre, categoría, tiempo de comida, porciones, ingredientes, medidas, pasos e indicaciones para preparar la receta.

RQL-02	Agregación de comentarios a una receta.
Funcional	Toda receta debe permitir comentarios, los usuarios visitantes ingresaran comentarios respecto a una receta.

RQL-03	Evaluar una receta
Funcional	Toda receta puede tener cero o varias aprobaciones de los visitantes por medio de un “me gusta”.

RQL-04	Ilustración de las indicaciones de elaboración
Funcional	Todas las indicaciones de elaboración de una receta deben de estar ilustradas con una imagen.

RQL-05	Aparición de las cantidades de medida de los ingredientes.
Funcional	Para toda receta debe aparecer una tabla o lista con la lista de los ingredientes y las medidas de los mismos a un lado.

RQL-06	Aparición de los precios de los ingredientes.
Funcional	Para toda receta debe aparecer al lado lista de los ingredientes los precios respectivos a cada ingrediente

Requerimientos de escritores.

RQE-01	Ingreso de información de la receta
Funcional	Se debe permitir ingresar toda la información base de una receta: Nombre, categoría, tiempo de comida, porciones, ingredientes, medidas, pasos e indicaciones para preparar la receta.

RQE-02	Cambiar información de la receta
Funcional	Se debe permitir al autor de la receta hacer cambios a la misma

RQE-03	Hacer colección o libro de recetas
Funcional	Un autor puede hacer colecciones de recetas en base a categorías o tiempos.

Requerimientos de sistema.

RQS-01	Control de usuarios
Funcional	La aplicación contará con un sistema de manejo de usuarios.

RQS-02	Obtención de precios mediante api externo
Funcional	La aplicación usará el api Junar para obtener información de los precios de los ingredientes.

RQS-04	Aplicación responsive
Funcional	La aplicación debe funcionar en web para computador, tableta o Smartphone.

RQS-05	Notificación por suscripciones.
Funcional	La aplicación debe notificar a los usuarios suscritos a un escritor, cuando éste ingresó una nueva receta.

Prioridades.

Funcionales

1. Crear un sistema de control de usuarios que nos permita especificar que permisos tiene un usuario de nuestra página sobre los elementos que contiene; ya sea una receta, un libro de recetas, un comentario, entre otros…

2. Asegurar el correcto almacenamiento de la información personal de los usuarios, sus aportes culinarios y demás publicaciones; esto con el fin de que no haya ninguna pérdida ni incongruencia entre los contenidos.

3. Conectar nuestra página, de manera segura y estable, con los API's de Junar y Facebook con el fin de extraer información relevante de los ingredientes y otros métodos de divulgación del contenido de la página respectivamente.

4. Notificar a los usuarios suscritos de las nuevas recetas que otro usuario al que se está suscrito actualmente publique en la página de manera reciente.

5. Implementar un sistema de evaluación, basado en aprobaciones (me gusta) de los usuarios, de las recetas publicadas en la página; además de poder dejar algún comentario en la receta para que los demás usuarios puedan visualizar.

De Calidad

1. Diseñar la página para que esta pueda ser desplegada de manera correcta tanto en computadoras personales como en dispositivos móviles como tabletas y celulares.

2. Crear una vista apropiada para que, las recetas creadas por los escritores, se visualicen de una manera clara y correcta para los demás lectores de la página.

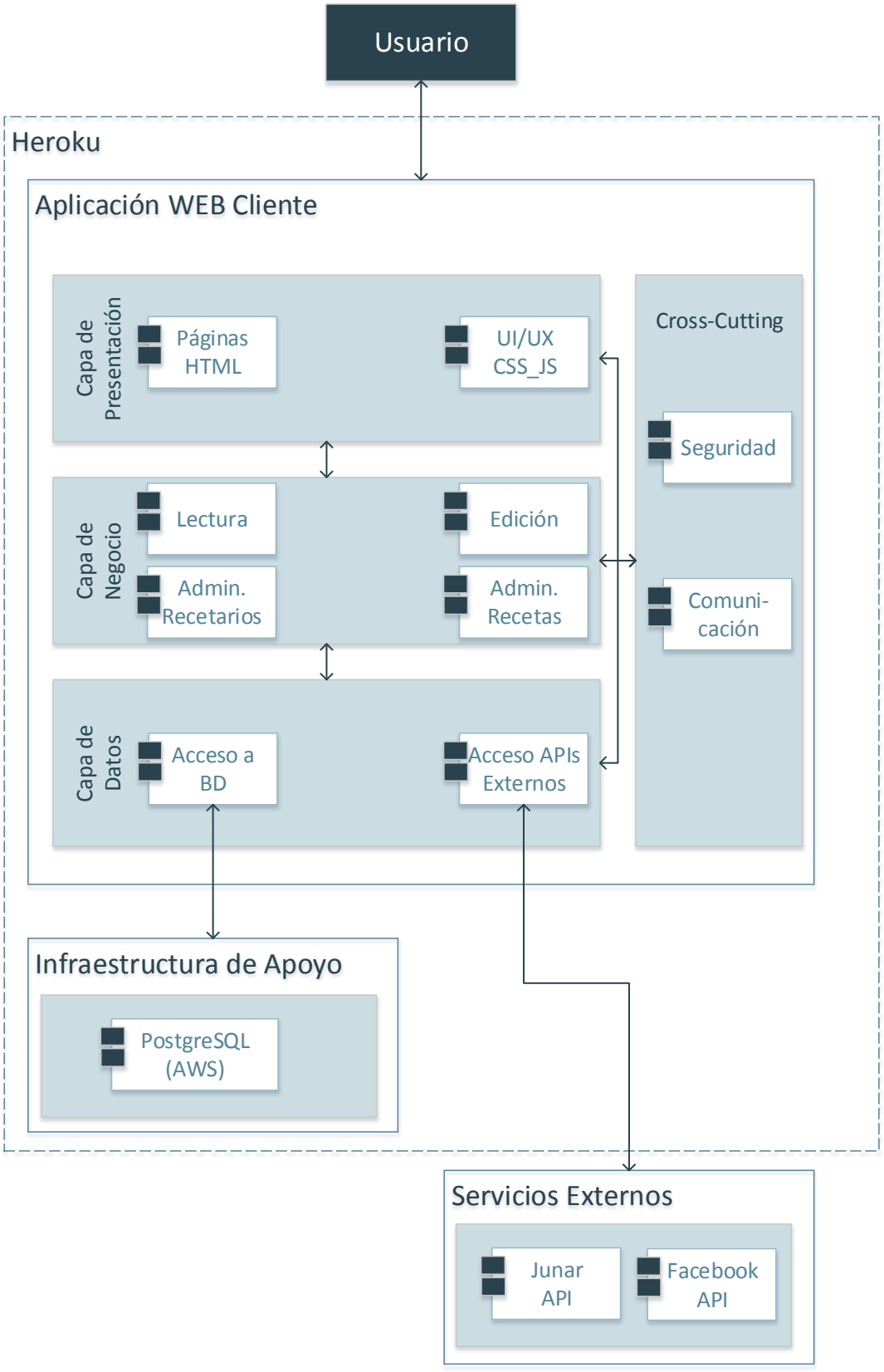
3. Obtener información clara y concisa de los datos extraídos del api de Junar para la actualización dinámica de los datos de los ingredientes específicos.

Descripción de diseño de alto nivel

Diagrama de arquitectura conceptual

La aplicación se va a encontrar hospedada en Heroku, la cual se autodenomina como "una plataforma como servicio de computación en la nube que soporta distintos lenguajes de programación", y va implementar el patrón de arquitectura conocido como model-view-controller, mejor conocido como Patrón MVC.

Los usuarios de la plataforma anteriormente mencionada les son facilitados el hospedaje de la aplicación y la base de datos en los servicios proporcionados por Amazon.com en sus herramientas de desarrollo en la nube: AmazonWS.



Uso de Django

A continuación se presenta una pequeña introducción a Django, marco de desarrollo web basado en Python, el cual utiliza Heroku para la gestión de aplicaciones web.

Manejo de controladores (views)

El manejo de los “controladores” en Django se utiliza mediante los archivos llamados views. En el caso de la aplicación existe views.py. Dentro lo que poseen son los métodos de llamada para las vistas de la página, cada request permite mostrar una página con la información que contenga el archivo estático y la información dinámica que se deba presentar.

```
def home(request):
    if request.user.is_authenticated():
        return HttpResponseRedirect()
    else:
        return login(request)

def index(request):
    return render_to_response("rece/index.html",
                              RequestContext(request))
```

Manejo de vistas (HTML)

Las vistas y los elementos HTML son elementos estáticos los cuales deben de colocarse en una sección llamada “templates” la cual es una carpeta que contiene las vistas estáticas que se presentan al usuario.

También está la carpeta “static” la cual es la carpeta de elementos estáticos que no son vistas como los CSS, las imágenes y archivos JavaScript, en los archivos de template y las views se deben pasar estos elementos para definir y conectar los elementos estáticos.

Manejo de rutas (URL)

Dentro del archivo url.py se utilizaron las siguientes rutas para hacer uso del API de Facebook y navegar dentro de la aplicación.

```
url(r'^$', 'receticas.views.index'),
# url(r'^blog/', include('blog.urls')),
url(r'^accounts/logout/$', 'django.contrib.auth.views.logout', {'next_page': '/rece/'}),
url(r'^admin/', include(admin.site.urls)),
url(r'^login/$', 'auth.views.login_user'),
url(r'^home/$', 'receticas.views.home'),
url(r'^accounts/', include('allauth.urls')),
url(r'^rece/', 'receticas.views.index'),
```

Manejo de modelos (models)

Respecto a los modelos, estos son lo que realizan el manejo propio de usuarios dentro de la aplicación.

```
class AuthGroup(models.Model):
    id = models.IntegerField(primary_key=True)
    name = models.CharField(max_length=80)
    class Meta:
        managed = False
        db_table = 'auth_group'

class AuthGroupPermissions(models.Model):
    id = models.IntegerField(primary_key=True)
    group = models.ForeignKey(AuthGroup)
    permission = models.ForeignKey('AuthPermission')
    class Meta:
        managed = False
        db_table = 'auth_group_permissions'
```

Diagrama de componentes

Este diagrama describe en alto nivel los componentes que tendrá nuestro sistema en sus diferentes niveles, al mismo tiempo que nos permite ver cómo se interrelacionan dichos elementos entre sí. Como se puede observar, el primer componente (componentes de UI/UX), trata los temas de visualización para los usuarios de la página, tanto para lectores y escritores como para administradores. Seguida de este componente general encontramos el componente que involucra la lógica de negocio (Componentes de Manejo de información) donde se presentan los componentes de interacción entre el front-end y las conexiones con sistemas externos y almacenamiento de la información (Componentes de Conexión).

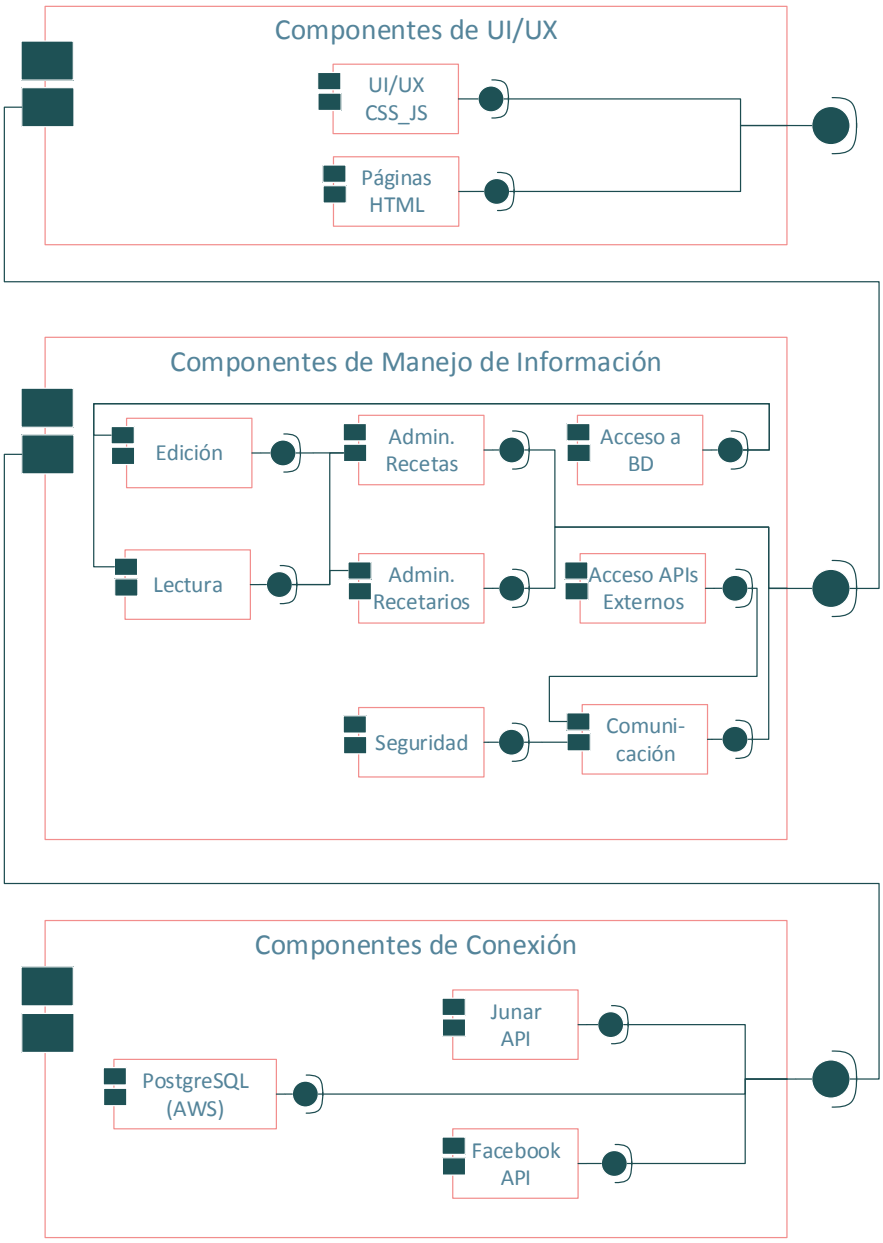


Diagrama de paquetes

En este diagrama se expresan las decisiones y representaciones lógicas del sistema de alto nivel, de manera que se muestren las agrupaciones y conexiones de cada paquete del sistema entre sí. Como lo representa la figura, existe un macro paquete, en el cual se contemplan todos los detalles de visualización, conexiones, usuarios y contenidos. Este paquete (Sistema) se comunica a su vez con los paquetes de bases de datos, el API de la red social Facebook y el API de Junar, del que se extraen los valores de los ingredientes por medio de la base de datos del MEIC contenida en este sitio.

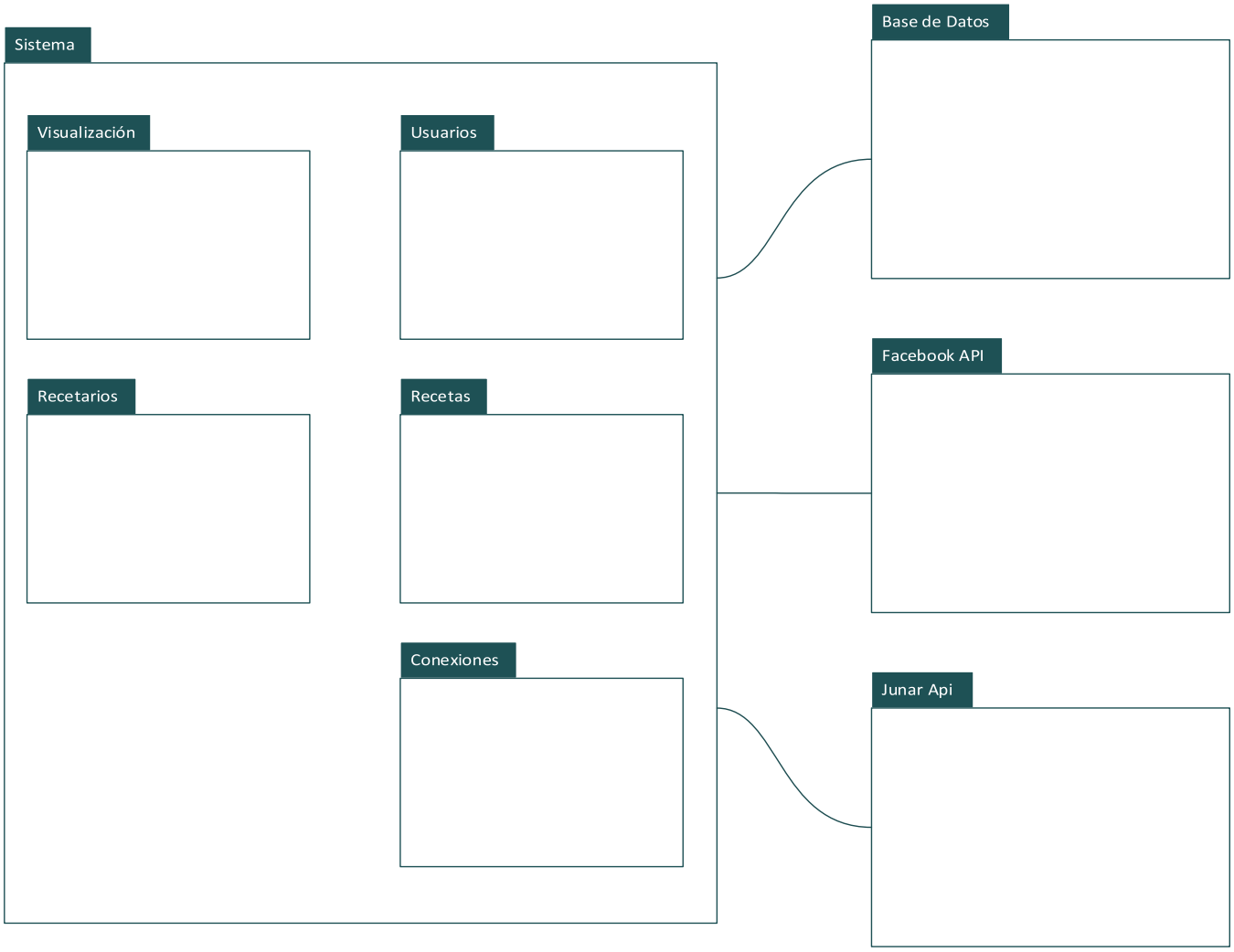
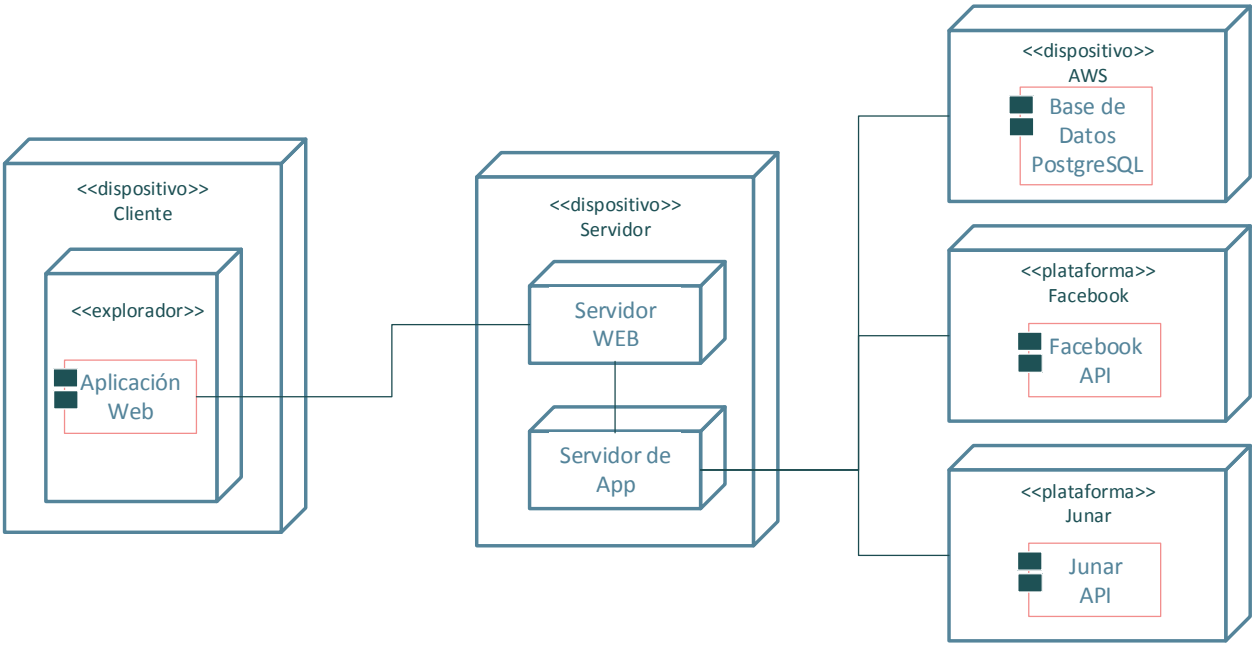


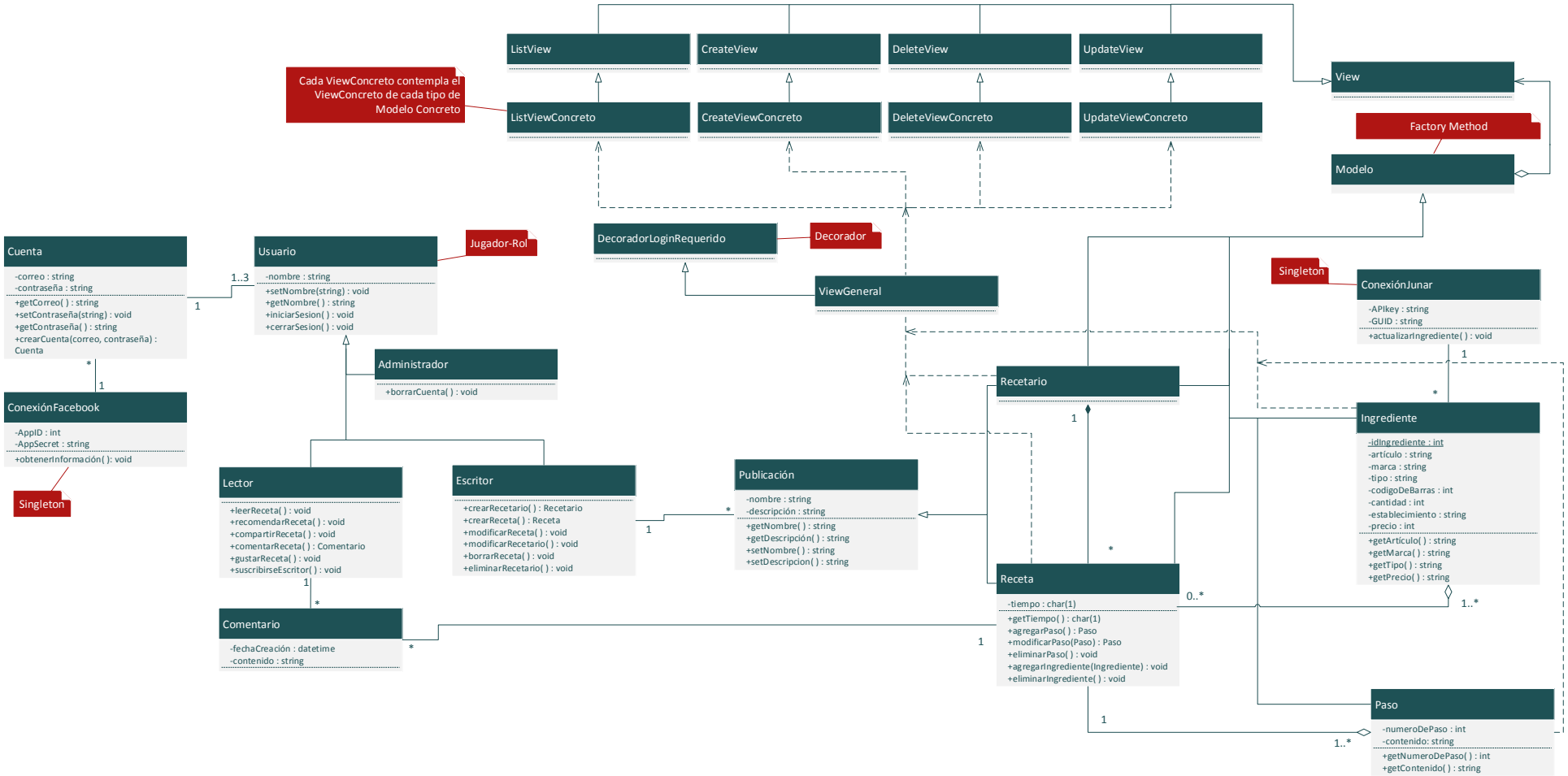
Diagrama de despliegue

Este diagrama lo utilizamos para representar los distintos dispositivos involucrados en el funcionamiento del sistema, además de los ambientes de ejecución contenidos en estos equipos. El diagrama describe, desde su inicio, el equipo, ya sea un dispositivo móvil o una computadora personal, interactuando con el servidor del sistema por medio de su explorador de internet. Este servidor, a su vez, interactúa con la base de datos hospedada en AmazonWS (con el fin de gestionar la información de manera segura y accesible), y el API de Facebook, del que se extrae la información necesaria de un usuario que desee afiliarse a la página y disfrutar de su contenido sin procesos tediosos ni olvidadizos. Además de las conexiones anteriormente descritas, se realiza una conexión eventual con el API de Junar, del que se extraen los precios de los distintos ingredientes disponibles en el país.



Descripción detallada

Diagrama de clases



Justificación de patrones usados

Patrón MVC (Model-View-Controller)

Django, como se ha descrito con anterioridad, utiliza el patrón Model-View-Controller, visto con más detalle en la sección “ Uso de Django “, pues, la vista de los usuarios viene de los elementos de vistas, aunque la interacción del usuario va directo al controlador; mientras que ambos componentes son gestionados por el modelo, donde toda la lógica de negocios toma lugar.

Jugador-Rol

Como se cuenta con un control de usuarios, es necesario mantener una jerarquía de permisos de edición y vista de datos almacenados por los demás usuarios, por lo que cada cuenta tiene asignado uno o más roles que le otorgan permisos distintos al resto de usuarios. Esto se visualiza más claramente al momento de ver las recetas o recetarios, los cuales muestran íconos de edición al lado de los elementos pertenecientes al usuario; mientras que para los demás usuarios no les son mostrados.

Singleton

La información referente a un ingrediente específico es extraída de un api externo, el cual nos permite saber la situación actual de dicho ingrediente, como lo es el precio en nuestra región. La información requerida para conectarse a dicha aplicación no debe duplicarse de ninguna forma a lo largo de la consulta de estos datos. Un escenario muy similar es el que nos encontramos con el inicio de sesión a través de Facebook, el cual, al utilizar una configuración única y específica, obtiene una conexión estable con la plataforma para extraer la información necesaria de inicio de sesión de los usuarios.

Factory Method

Este patrón se ve claramente en la estructura de vistas y modelos, donde cada modelo tiene vistas de actualización, edición, borrado y visualización distintas, asignadas específicamente a cada uno de los modelos (Ingrediente, Paso, Receta y Recetario); asociando de manera directa una vista concreta, con su respectivo modelo concreto.

Decorador

Para cada una de las vistas generadas, se utiliza un patrón decorador, el cual modifica la vista específica dependiendo del estado del usuario, específicamente si ha iniciado sesión o no, en el sitio. Esto le permite al usuario iniciar o cerrar sesión en las distintas páginas en las que se aventure el usuario a lo largo de su visita.

Problemas de diseño.

A nivel de diseño se tienen problemas con la conexión del api de Junar, este api a pesar de la buena información que puede entregar tiene problemas a nivel de documentación respecto a su forma de uso y de despliegue de información.

Los problemas de documentación son más agudos ya que no indican las fuentes de información abierta para hacer consultas, además de que dicha información no aparece en las entidades que utilizan dicho api, estas solo mencionan la forma general de efectuar la consulta, pero las consultas están pre construidas lo cual hace que las consultas aunque hagan fácil la consulta de datos, las hace difíciles de generar ya que son consultas que no pueden crearse deben buscarse dentro de las páginas de las entidades.

Interacción con sistemas externos.

Api junar.

El api de junar es un componente externo de manejo de información para elementos de información abiertos, como las bases de datos abiertas de gobiernos, empresas y organizaciones las cuales permiten el acceso a esos datos para información pública, dicho api permite conocer la información y exportarla en diferentes formatos los cuales pueden ser utilizados para mostrar la información consultada.

Mediante este api, se pretende obtener información de las bases de datos abiertas del Ministerio de Economía Industria y Comercio de Costa Rica, para obtener la información sobre los productos base, dicha información es respecto a los precios y las marcas de los productos para definir los precios de las recetas.

Api Facebook:

El api de Facebook es un componente que permite a aplicaciones ejecutar acciones desde las cuentas Facebook del propietario, como login, hacer comentarios que aparecen en Facebook, o dar me gusta a publicaciones.

El api en este caso será utilizado para dichos propósitos principalmente en el caso de login, el api se conectará con los servicios de Facebook para brindar el servicio de sesión para los usuarios y para los elementos de me gusta o de comentarios.

Bootstrap:

Librería externa, desarrollada por Twitter, para el correcto desarrollo de páginas web de tipo responsive y de estilos estandarizados (normalize) para que, en los distintos exploradores WEB, la página se vea como fue diseñada; sin variaciones de tamaño ni color. La biblioteca se utilizará para que las páginas tengan la capacidad de visualizarse de manera correcta en los distintos dispositivos con los que desee ingresar el usuario.

Conclusiones

1. A nivel de manejo con Djando, el framework obliga a que los elementos como páginas de HTML tengan que ser controladas por view's lo que hace que algunos aspectos de presentación no puedan ser manejados correctamente.

2. Los patrones de diseño, aunque no en todos los casos, se reflejan en código de forma explícita, ya que algunos patrones se pueden implementar en elementos funcionales y, a la vez, es una decisión de negocio o de implementación.

3. Además de los patrones vistos en clase, se requirió de la búsqueda de patrones externos; de los cuales algunos ya son implementados por Django en su desempeño.

Recomendaciones

1. Se debe tener un diseño preliminar de los HTML y este debe mejorarse en paralelo en base a los view que se crean para aspectos funcionales y de presentación.

2. Representar los patrones que no se colocan de forma explícita en código en la parte de conexión externa que lo requiere.

3. Se deben presentar patrones que sean usados de forma más abierta y común en proyectos de desarrollo de software.

Referencias

Adam-bray.com, (2014). HTML 5 Vertical Navigation Menu - Adam Bray. [online] Available at: http://www.adam-bray.com/blog/103/html-5-vertical-navigation-menu/ [Accessed 24 Jun. 2014].
Amorousconsultants.com, (2014). Facebook Login Step-by-Step Guide. [online] Available at: http://amorousconsultants.com/Facebook-login/Facebook-Login-Step-By-Step-Guide.asp [Accessed 24 Jun. 2014].

Blog.xenodesystems.com, (2014). Montar entorno de Desarrollo Python-Django en Ubuntu 12.04 | Xenode Systems Blog. [online] Available at: http://blog.xenodesystems.com/2012/10/montar-entorno-de-desarrollo-python.html [Accessed 24 Jun. 2014].

Blog.xenodesystems.com, (2014). Montar entorno de Desarrollo Python-Django en Ubuntu 12.04 | Xenode Systems Blog. [online] Available at: http://blog.xenodesystems.com/2012/10/montar-entorno-de-desarrollo-python.html [Accessed 24 Jun. 2014].

Cibernatural.com, (2014). Tutorial de Django – I « Cibernatural. [online] Available at: http://www.cibernatural.com/tutorial-de-django-i [Accessed 24 Jun. 2014].

Cristalab, (2014). Python en la web con Django (IV): uso de plantillas. [online] Available at: http://www.cristalab.com/tutoriales/python-en-la-web-con-django-iv-uso-deplantillas-c103814l/ [Accessed 24 Jun. 2014].

Devcenter.heroku.com, (2014). Getting Started with Django on Heroku | Heroku Dev Center. [online] Available at: https://devcenter.heroku.com/articles/getting-startedwith-django [Accessed 24 Jun. 2014].

Django.es, (2014). Django | Tu primera aplicación Django, parte 1. [online] Available at: http://django.es/docs/intro/tutorial01/ [Accessed 24 Jun. 2014].

*Django.wikispaces.asu.edu, (2014). Django - *NEW* Django Design Patterns. [online] Available at: http://django.wikispaces.asu.edu/*NEW*+Django+Design+Patterns [Accessed 24 Jun. 2014].*

Django.wikispaces.asu.edu, (2014). Django - ModelViewControl. [online] Available at: http://django.wikispaces.asu.edu/ModelViewControl [Accessed 24 Jun. 2014].

Django-book.blogspot.com, (2014). Django, Todo lo que quieres saber y algo mas. [online] Available at: http://django-book.blogspot.com/2012/11/metodos-de-losmodelos-definir-metodos.html [Accessed 24 Jun. 2014].

Docs.djangoproject.com, (2014). Creating forms from models | Django documentation | Django. [online] Available at: https://docs.djangoproject.com/en/1.6/topics/forms/modelforms/#modelform [Accessed 24 Jun. 2014].

Docs.djangoproject.com, (2014). Form fields | Django documentation | Django. [online] Available at: https://docs.djangoproject.com/en/dev/ref/forms/fields/ [Accessed 24 Jun. 2014].

Docs.djangoproject.com, (2014). Model field reference | Django documentation | Django. [online] Available at: https://docs.djangoproject.com/en/dev/ref/models/fields/ [Accessed 24 Jun. 2014].
Docs.djangoproject.com, (2014). QuerySet API reference | Django documentation | Django. [online] Available at: https://docs.djangoproject.com/en/dev/ref/models/querysets/ [Accessed 24 Jun. 2014].

Docs.djangoproject.com, (2014). User authentication in Django | Django documentation | Django. [online] Available at: https://docs.djangoproject.com/en/1.6/topics/auth/ [Accessed 24 Jun. 2014].

Docs.djangoproject.com, (2014). Using the Django authentication system | Django documentation | Django. [online] Available at: https://docs.djangoproject.com/en/1.5/topics/auth/default/ [Accessed 24 Jun. 2014].

Docs.djangoproject.com, (2014). Using the Django authentication system | Django documentation | Django. [online] Available at: https://docs.djangoproject.com/en/1.6/topics/auth/default/ [Accessed 24 Jun. 2014].

Hellmann, D. (2014). abc – Abstract Base Classes - Python Module of the Week. [online] Pymotw.com. Available at: http://pymotw.com/2/abc/ [Accessed 24 Jun. 2014].

Iliev, I. and Villegas, C. (2014). Django forms ChoiceField and custom HTML output…»Between engineering and real life. [online] Ilian.i-n-i.org. Available at: http://ilian.i-n-i.org/django-forms-choicefield-and-custom-html-output/ [Accessed 24 Jun. 2014].

model?, H. (2014). python - How would I use django.forms to prepopulate a choice field with rows from a model? - Stack Overflow. [online] Stackoverflow.com. Available at: http://stackoverflow.com/questions/4593292/how-would-i-use-django-forms-toprepopulate-

a-choice-field-with-rows-from-a-mode [Accessed 24 Jun. 2014].
Tutorialdjango.com.ar, (2014). Empezando con Django — Django 1.5 documentation. [online] Available at: http://tutorialdjango.com.ar/ [Accessed 24 Jun. 2014].
views?, H. (2014). python - How to require login for django generic views? - Stack Overflow. [online] Stackoverflow.com. Available at: http://stackoverflow.com/questions/2140550/how-to-require-login-for-djangogeneric-views [Accessed 24 Jun. 2014].