MP0391
Módulo de prácticas profesionales no laborales

PROYECTO: GUIA DE RESTAURANTES

Fernando López López

DESCRIPCIÓN

Debemos crear una aplicación web una aplicación web con una guía de restaurantes, con buscador y reseñas, donde los usuarios podrán comentar y valorar los restaurantes.
La aplicación se debe desarrollar en ***HTML***, ***CSS***, ***framework Bootstrap*** para facilitar el diseño adaptativo (*responsive*), lenguaje ***PHP*** en el entorno servidor, y servidor de base de datos ***MySQL***.
Al *frontend*, además de poder buscar, comentar y valorar los restaurantes, se mostrará una ficha individual de cada restaurante con información detallada.
Además, se ha de mostrar un mapa de ***Google Maps*** o similar, con la ubicación de los restaurantes.
Al *backend*, los usuarios con perfil de administrador, podrán realizar las tareas de mantenimiento básicas ***CRUD*** (*Create*, *Read*, *Update*, *Delete*) de los restaurantes.
La aplicación se desplegará dentro contenedores ***Docker*** para mejorar su modularidad.

Durante el desarrollo del proyecto se utilizará las metodologías ágiles **SCRUM**, para emular de la forma más fiable un desarrollo profesional con ésta filosofía de desarrollo.

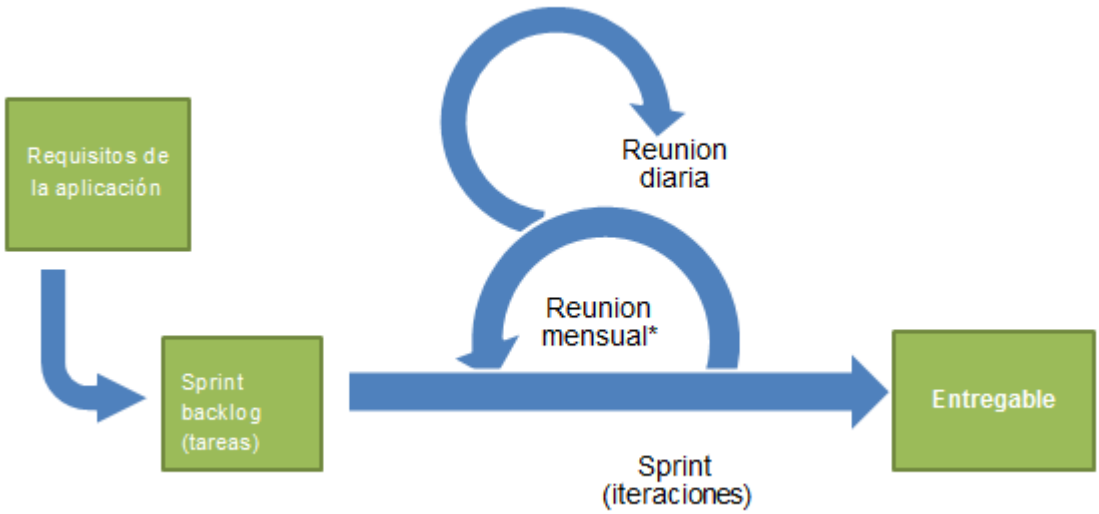
CRITERIOS DE EVALUACIÓN

Los criterios de evaluación que se han tenido en cuenta en este proyecto son los siguientes:

- Participar en el análisis y diseño del desarrollo de los sistemas realizados en sus correspondientes tecnologías web.
- Colaborar en el análisis de todas las especificaciones del desarrollo, tanto en el entorno cliente como servidor.
- Evaluar la conformidad de los sistemas web respecto a los requisitos preestablecidos por la empresa o cliente.
- Validar la utilización de tecnologías y estándares en la realización del sistema web
- Elaboración de diferentes tipos de documentación.
- Análisis de nuevas herramientas que puedan ser útiles para el desarrollo y despliegue de aplicaciones web
- Colaborar en la definición y mejora de los entornos organizativos que se han creado para facilitar la coordinación entre los grupos de trabajo: Diseño, desarrollo, administración de aplicaciones web, etcétera
- Análisis de usuabilidad
- Participación en los procesos de trabajo de empresa, siguiendo las normas y horarios estabelcidos por el centro de trabajo.

SCRUM COMO METODOLOGÍA DE TRABAJO

Para el desarrollo de este proyecto se ha seleccionado el marco de trabajo para el desarrollo ágil de software llamado **SCRUM** (en castellano, melé).



¿Qué es una metodología ágil?

Es una forma de trabajo **que abraza el cambio**. Las viejas formas de trabajo planifican a priori todo el proyecto, con cronogramas de entrega, por ejemplo, consumiendo mucho tiempo en esta fase sin haber avanzado nada. Si por algún motivo existe algún cambio en el camino de desarrollo toda esa especificación anterior del proyecto puede quedar obsoleta, necesitando una nueva replanificación. Como se entenderá con facilidad, esto generará un gran retraso en la entrega del proyecto, y por tanto, un encarecimiento considerable del mismo, saliéndose del presupuesto con gran facilidad.

Una metodología ágil en esencia**, busca reaccionar rápidamente ante los cambios en el desarrollo** sin que por ello se vuelva inestable el proyecto.
En este sentido, se busca un equilibrio en la producción de un proyecto siempre estable y presentable, con un enorme poder de adaptación ante todas las circunstancias y cambios futuros.

¿Qué es SCRUM?

Es complicado definir de forma concreta que es el **SCRUM**. La forma más sencilla que se me ocurre es que es un marco de trabajo que determina unas pautas, **que no necesitan ser seguidas a pie de la letra**. Hasta tal punto es flexible, que cada cual elije las pautas que considere necesario adaptándolas a sus necesidades y gustos. Por ello muchos desarrolladores lo consideran un *Framework*, ya que permite amoldarse.

Me gustaría destacar en éste sentido que los proyectos no se han de adaptar a SCRUM sino que es SCRUM quien se adapta de la forma más conveniente al proyecto.

Roles en SCRUM

A continuación, relatamos los roles que forman la metodología **SCRUM** para posteriormente comentar como se ha desarrollado en éste proyecto en concreto.

PRODUCT OWNER

El *Product Owner* es el enlace entre quien toma las decisiones finales del proyecto, generalmente el cliente, y el equipo de desarrollo técnico. Estos clientes, que ya tienen muy claro que especificaciones debe de tener el proyecto transmiten los detalles al *Product Owner* para **que éste las filtre** y se sepa siempre que se debe de hacer en cada momento. En cierta manera es el director de orquesta, que decide que es primordial ahora, que se aplaza,

que es importante desplegar, y en general, controla el equilibrio entre una versión presentable del proyecto, y su verdadero desarrollo interno. Prioriza las tareas pendiente de **que siempre exista una versión funcional**. Despliega todas **las historias de usuario** que conformarán del *Backlog*.

SCRUM MASTER

El *Scrum Master* suele ser, o debería ser, un experto en **Scrum**. Se encarga de que el equipo trabaje fielmente bajo ésta metodología, **ayudándoles y facilitándoles el trabajo.** No se trata de un jefe o gestor del proyecto. Es un líder que facilita el trabajo a los miembros del equipo de desarrollo en la medida en que estos dispongan de dificultades o inseguridades. Es el encargado de dirigir las reuniones diarias que comentaremos más adelante.

EQUIPO DE DESARROLLO

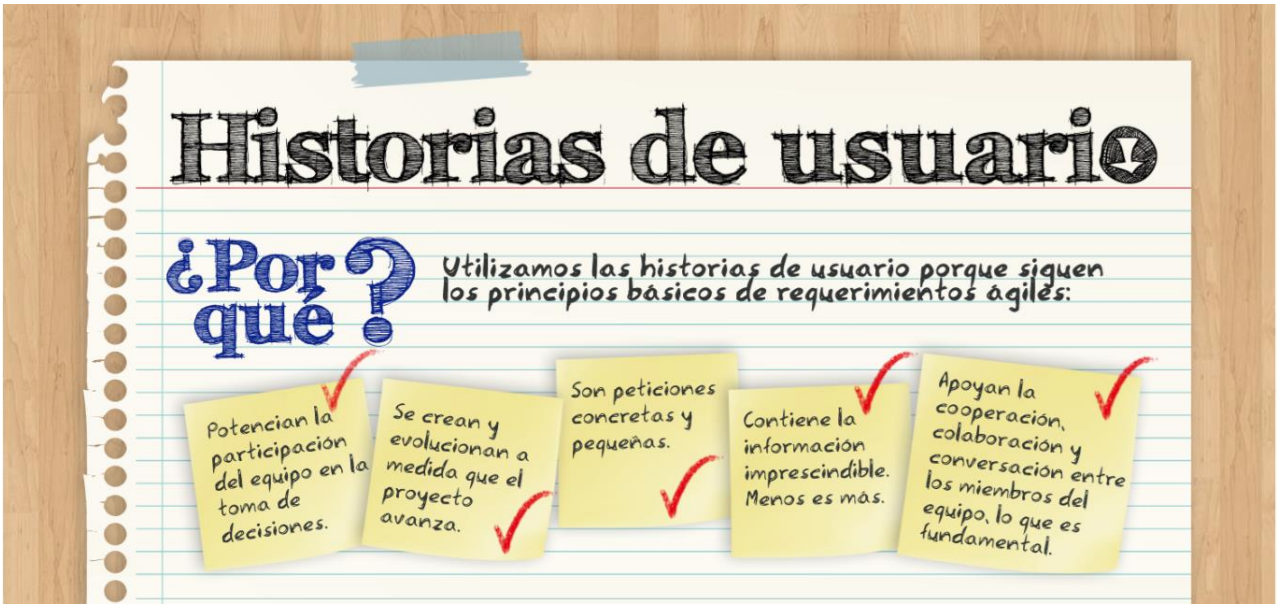
Se trata del equipo de profesionales que se encargan de crear el producto. Diseñadores, *Front-end, Back-end, Devops*, etcétera. Es interesante destacar que estas clasificaciones o especializaciones del equipo **no se tienen en cuenta en la metodología SCRUM aplicada al desarrollo informático**. Por tanto, estas categorías no pertenecen a la metodología. Ya que se supone que el equipo también tiene que tener adaptabilidad al cambio y capacidad para realizar tareas en todos estos perfiles clásicos. Esto, al menos, es lo deseable. Otro punto muy interesante **es que el equipo mismo elije las tareas preparadas en el backlog que van a desarrollar. En este sentido, el equipo de desarrollo se autogestiona la elección y desarrollo de los tickets.**

CUSTOMERS

Los tomadores internos de decisiones en una organización. Serían miembros de otros departamentos en la empresa, por ejemplo; Contabilidad, marketing, RRHH, etcétera, que determinan las especificaciones de un proyecto. También podría ser un representante **de un cliente externo o *USER*.**

USER

El *user* sería el cliente externo que evaluaría las versiones funcionales que se les va presentando. A través de sus indicaciones el *Product Owner* generará **las historias de usuario** que se convertirán en el *Backlog* de trabajo.



SPRINTS

Todas las tareas que forman el *Backlog* se desplegarán divididas en ***Sprints***. Generalmente suelen durar unas dos semanas. El objetivo es que siempre exista al final del Srpint una versión funcional del proyecto. De ahí la importancia de que el *Scrum Master* sepa evaluar las dificultades en el desarrollo del proyecto para conseguir este objetivo y gestionar la problemática diaría para facilitar el trabajo al equipo de desarrollo. Las tareas que se van a realizar se definen al inicio del Sprint teniendo en cuenta el tiempo de trabajo. Al finalizar el Srpint se realiza una review y una retrospectiva para comentar como ha sido el flujo de trabajo del Sprint.

ADAPTACIÓN DE LAS PRÁCTICAS A LA METODOLOGÍA SCRUM

El Módulo de prácticas profesionales no laborales, **MP0391,** del Certificado Profesional **IFCD0210 (Desarrollo de aplicaciones con tecnología web),** tenía una duración de 80 horas (dos semanas). Como se comprenderá no es tiempo suficiente para realizar varios Sprints dentro de la metodología ágil **SCRUM**. Por tanto, se ha tenido que adaptar la metodología SCRUM al limitado tiempo disponible. Esto nos lleva a comprobar la **elevada versatilidad de la metodología,** ya que a pesar de éstas circunstancias se ha podido realizar con éxito. Aun así, considero que se ha realizado un esfuerzo por todo el equipo digno de mención. Aprovecho la ocasión para agradecer a todos esta implicación.

Comprobemos las adaptaciones particulares que se han realizado durante estas prácticas para la realización de varios **Sprints en tan poco tiempo**, y así poder emular de la forma más cercana posible a la realidad de un clima laboral en desarrollo web con metodologías ágiles

---

TIEMPO DE EJECUCIÓN y PRODUCT OWNER

El profesor responsable de la supervisión de nuestras prácticas no laborales fue **Miquel Àngel Cabot[1].** Será quien adoptase el importante papel de *Product Owner*. Él sería el encargado de traer las historias de usuario para el *backlog* de todo el proyecto. Nos adaptaría todas las tareas al breve tiempo disponible según las prioridades que él estimase, para llegar a realizar **a tiempo** en cada Sprint una presentación funcional del proyecto.

---

[1] https://github.com/miquelcabot

Temporalización

Semana	Lunes (8h)	Martes (8h)	Miércoles (8h)	Jueves (8h)	Viernes (8h)
1 (29/6/2020)	09:00 -Sprint Planning 09:30 - Scrum Daily	09:00 - Scrum Daily	09:00 - Scrum Daily **12:00 - Tutoria presencial**	09:00 - Scrum Daily 18:00 - Review/Retrospective	09:00 -Sprint Planning 09:30 - Scrum Daily **11:00 - Tutoria presencial**
	Sprint 1	Sprint 1	Sprint 1	Sprint 1	Sprint 2
2 (6/7/2020)	09:00 - Scrum Daily	09:00 - Scrum Daily **12:00 - Tutoria presencial** 18:00 - Review/Retrospective	09:00 - Sprint Planning 09:30 - Scrum Daily	09:00 - Scrum Daily	09:00 - Scrum Daily **11:00 - Tutoria presencial** 18:00 - Review/Retrospective
	Sprint 2	Sprint 2	Sprint 3	Sprint 3	Sprint 3

Se dividen las dos semanas **en tres *Sprints***. El primero, de cuatro días y dos de tres días. Se realizan los *Scrum Dailys* al igual que en cualquier otra empresa. Al comienzo de cada *Sprint* se realizan también **las planificaciones pertinentes**. Al acabar el *Sprint* se realiza una valoración general en la retrospectiva, no antes de haber realizado **la presentación funcional** en el correspondiente *review*.

ROLES

- ***Product Owner***: el Profesor.
- ***Scrum Master***: un estudiante en rotación diaría. Será seleccionado por orden alfabético.

- ***Development Team***: todos los estudiantes son un ***Development Team***. Cada cual realizará **su propio proyecto de forma individual para una mejor evaluación del mismo.**

NORMAS Y RESPONSABILIDADES

- **Reglas del proceso**
  - Sinceridad absoluta
  - Un nuevo *Scrum Master* cada día
  - *Scrum Daily* cada día, al principio de la clase
  - El miembro de un *Development Team* que se retrase pagará una penalización simbólica (por ejemplo, un café).
  -
- **Responsabilidades del *Scrum Master***
  - Saber quién es el siguiente *Scrum Master* de la lista
  - Si l'*Scrum Daily* se realiza telemáticamentea través de **Discord**, asegurarse que el día siguiente los roles estarán asignados
  - Iniciar *Scrum Daily* y asegurarse de que sea ágil.
  - Moderar el'*Scrum Daily*
  - Dar el *token* al miembro del *Development Team* que inicia el S*crum Daily,* por ejemplo una pelota de beisbol.
  - Dar visibilidad a los problemas del día a día.
  - Controlar el tiempo global de quince minutos.
  - Dar el espacio suficiente a todos los problemas y pretender resolverlos en el mismo día aportando soluciones.
  - Moderar a los miembros del *Scrum Daily* que no cumplan las normas.

- **Responsabilidades del *Development Team***
  - Entrega continua a los repositorios de *GitHub* de las tareas asumidas.
  - Realizar *commit*s continuos .
  - Cada projecto de cada equipo (Cada alumno es un equipo) dispondrá de un repositorio diferente de *GitHub*
  - Participar sinceramente en el proceso
  - Evaluar al *Scrum Master*
  - Avisar l'*Scrum Master* que no cumple con su cometido.

- **Responsabilidades del *Product Owner***
  - Expresar claramente las taeras del *Product Backlog*
  - Optimizar el valor del trabajo de cada *Development Team*
  - Asegurarse de que el *Product Backlog* sea visible, transparente y claro para todos
  - Asegurarse que el *Development Team* entienda las tareas del *Product Backlog*
  - Evaluar el *Development Team* según el cumplimiento de:
    - Despliegue continuo en *Github*.
    - Participación en el'*Scrum Daily*
  - Evaluar al *Scrum Master*
  - Avisar al *Scrum Master* si no cumple con las reglas del proceso.

  Avisar a los miembros del'*Scrum Team* que no cumplan con el proceso.

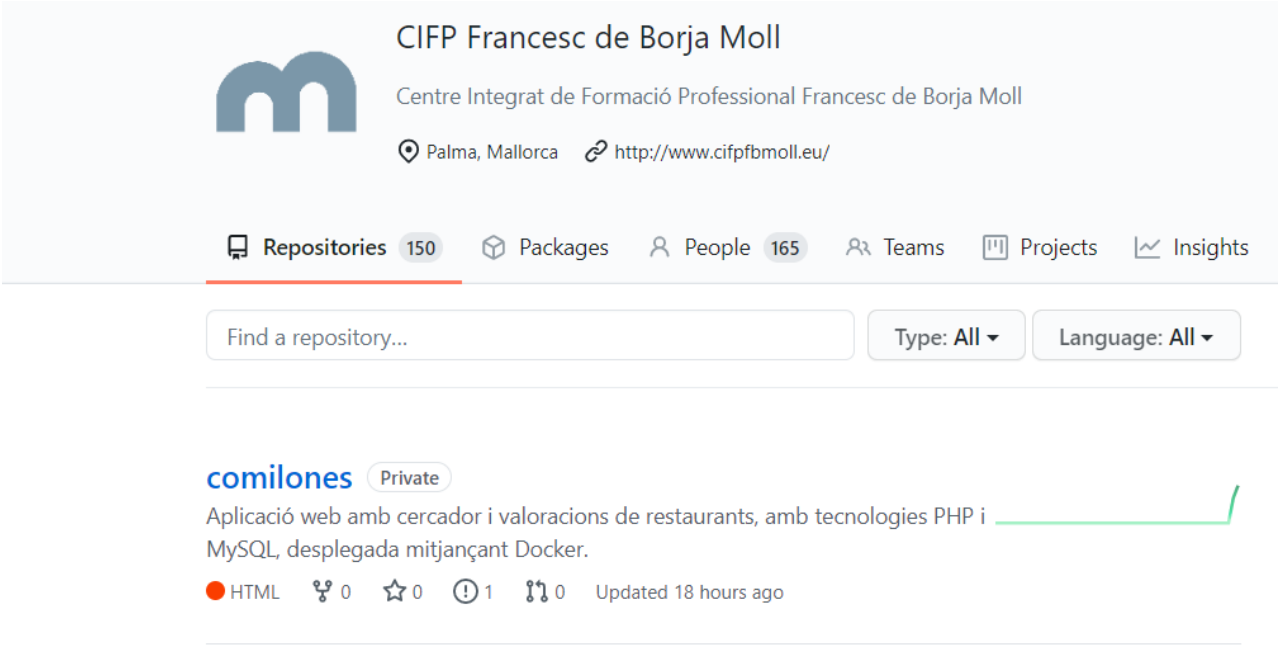
PLANIFICACIÓN INDIVIDUAL DEL PROYECTO

Como propio Development Team he tenido que planificar todas las tareas según los criterios de nuestro *Product Owner*. A continuación, relato la historia de toda ésta planificación.

CREACIÓN DE UN REPOSITORIO PRIVADO EN GITHUB

El centro que nos ha formado es el **CIFP de Francesc de Borja Moll**[2]. Éste dispone de un grupo de trabajo para sus alumnos en *Github*. Ese será el lugar en donde desplegaremos nuestros proyectos para una adecuada supervisión del responsable.

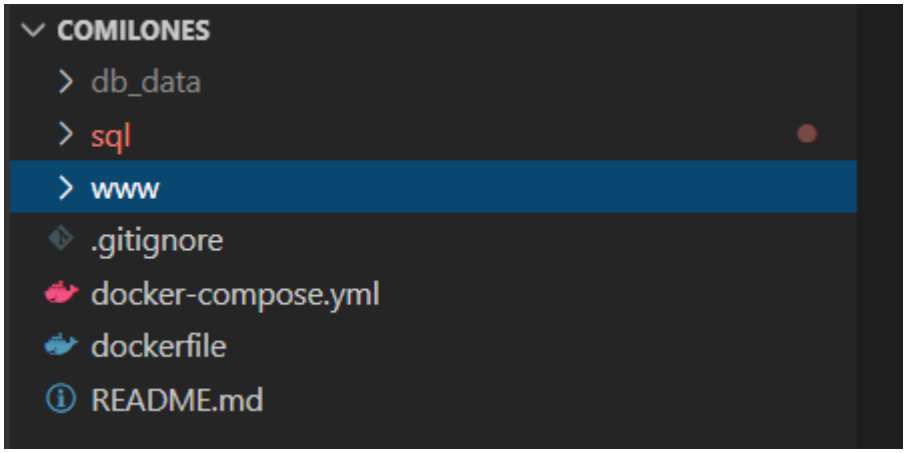
Mi proyecto se llamará **COMILONES**. Y con ese nombre cree mi proyecto remoto en ***Github.***



ESTRUCTURA INICIAL

He decidido comenzar creando los contenedores **Docker** en donde se desplegará mi aplicación. Por tanto la estructura inicial del proyecto es la siguiente;

[2] http://www.cifpfbmoll.eu/



- db_data:
  Carpeta autogenerada por **Docker cuando se inicia por primera vez el proyecto**.
  Más adelante comprobaremos donde.

- sql:
  Estructura de la base de datos

- www:
  Archivos del proyecto.

- .gitignore:
  Carpeta de Git para ignorar el seguimiento de los archivos que se consideren necesarios.

- Docker-compose.yml:
  Archivo Docker para desplegar el proyecto.

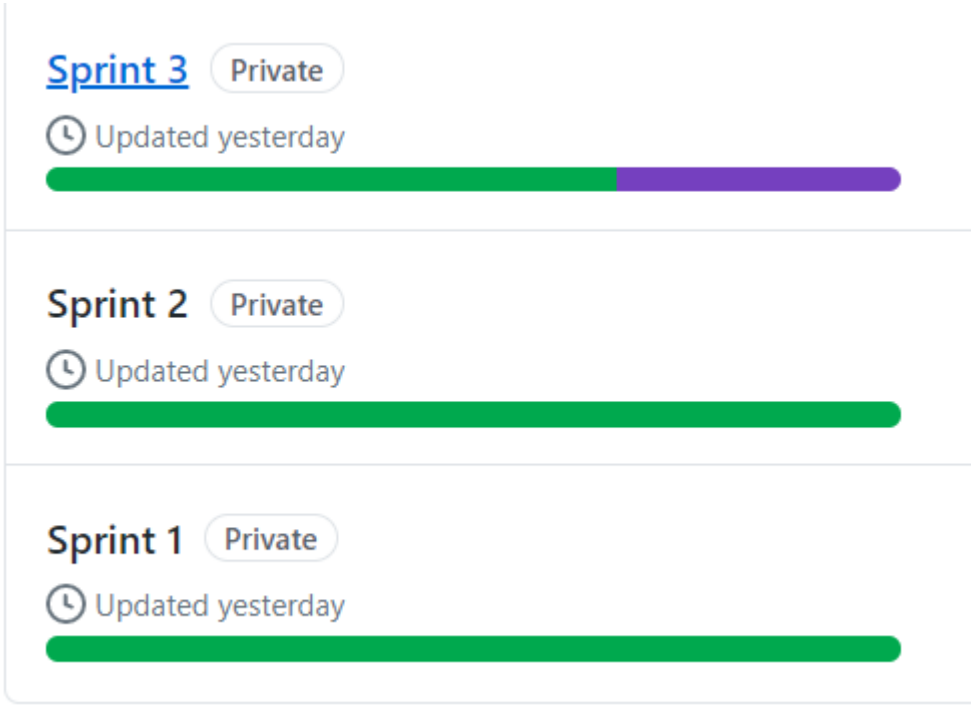
- Dockerfile:
  Es un documento de texto que contiene todos los comandos que un usuario podría llamar en la línea de comandos para ensamblar una imagen.

- README.md:
  El archivo de lectura en donde se explica la instalación de la aplicación y notas importantes a tener en cuenta.

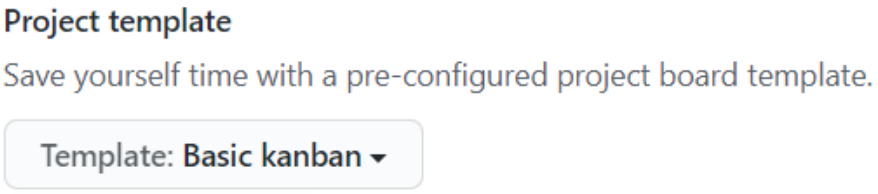
DESARROLLO EN GITHUB

Utilizando las herramientas que dispone *Github* para las metodologías ágiles, he realizado varios proyectos en donde se estructurarán las tareas con un *Kanban* básico. Veamos como ha sido este despliegue.

CREACIÓN DE LOS PROYECTOS:



Se crearon los proyectos que determinarán el desarrollo de cada *Sprint*. A la hora de realizar la creación de cada proyecto se seleccionó el tipo *Kanban Basic* que está disponible:



Las tareas diarias que serán transmitidas por nuestro *Product Owner* en el correspondiente *Backlog* serán detalladas en los *Issues* del proyecto en *Github*. Después, estos issues podrán estar vinculadas al proyecto correspondiente. Este *Kanban* básico disponía de tres estados:

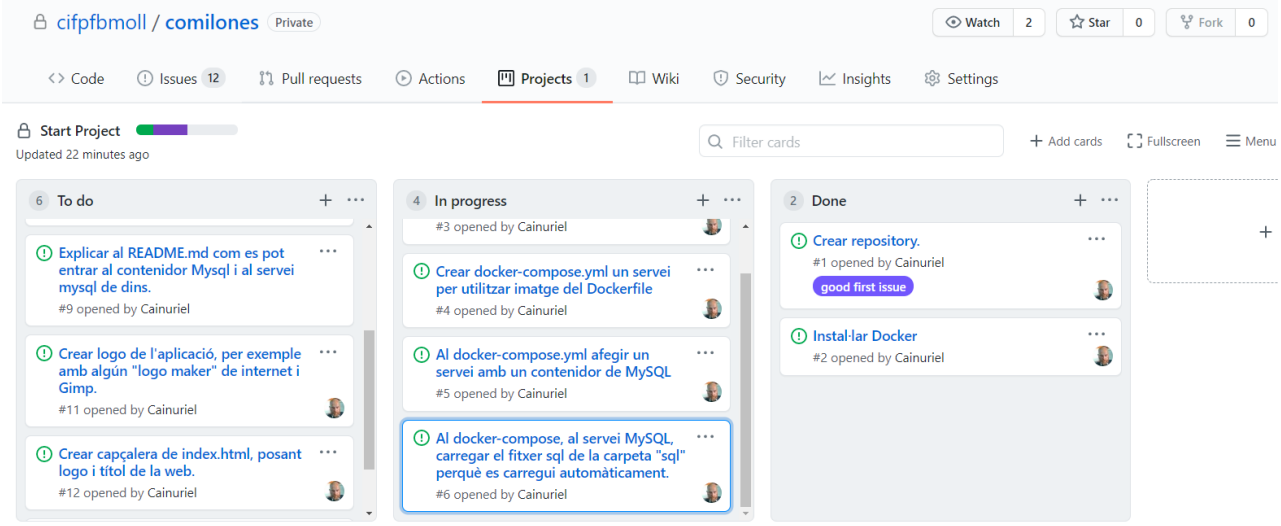
1. *To do*: Tareas no comenzadas.

2. *In process*; Tareas que se están trabajando

3. *Doing*: Tareas terminadas.

Las tareas pueden moverse en diferentes estados dependiendo de las revisiones correspondientes. Por ejemplo, una tarea terminada puede necesitar una revisión posterior. En ese caso será trasladada al estado de *In process* cuando nos estemos ocupando de ella.

A continuación una captura inicial de las *issues*, o tareas del proyecto inicial, o primer *Sprint*:



Vemos que las primeras tareas realizadas son la creación del repositorio en Github y la instalación de *Docker* para comenzar con el proyecto.

REQUISITOS DEL PROYECTO

Nuestro proyecto se desplegará con ***Docker***. Y estará formada por las siguientes tecnologías:

MYSQL version:8.0

A continuación, la configuración de su imagen en el *Docker-compose.yml:*

```
db-comilones:
   container_name: db-comilones
   image: mysql:8.0
   restart: always
   volumes:
     - ./db_data:/var/lib/mysql
     - ./sql:/docker-entrypoint-initdb.d
   ports:
     - '3306:3306'
   command: --default-authentication-plugin=mysql_native_password
   environment:
     MYSQL_DATABASE: comilones
     MYSQL_USER: user
     MYSQL_PASSWORD: password
     MYSQL_ROOT_PASSWORD: password
```

Cuando se despliegue por primera vez este contenedor creará **la carpeta db_data si no existe**. Veamos esos volúmenes:

```
 volumes:
     - ./db_data:/var/lib/mysql
     - ./sql:/docker-entrypoint-initdb.d
```

El ***initdb*** captura la estructura de la base de datos contenida en la carpeta **sql** y la carpeta **db_data** guardará los datos de forma persistente...

Phpmyadmin como gestor de base de datos.
A continuación las especificaciones de este contenedor con su imagen:

```
 phpmyadmin-comilones:
   depends_on:
     - db-comilones
   container_name: phpmyadmin-comilones
   image: phpmyadmin/phpmyadmin
```

```
    restart: always
    ports:
      - '8088:80'
    links:
      - db-comilones
    environment:
      PMA_HOST: db-comilones
      MYSQL_ROOT_PASSWORD: password
```

PHP version:7.3 con apache.

Este contenedor necesitará ser construido con el *dockerfile*. Ya que deseamos incorporar el uso de las librerías ***Mysqli*** dentro de la imagen. Para ello especificamos en el *dockerfile*:

```
FROM php:7.3-apache
RUN docker-php-ext-install mysqli
```

Ahora ya podemos montar este contenedor construyendo la imagen. El *dockerfile* está localizado en el mismo nivel que el archivo *Docker-compose.yml* y por tanto solo es necesario colocar un punto.

```
  php-apache-comilones:
    container_name: php-apache-comilones
    build: .
    restart: always
    volumes: ['./www:/var/www/html']
    ports:
      - '80:80'
    links:
      - db-comilones
```

Finalmente, el archivo ***Docker-compose.yml*** ha quedado de la siguiente forma;

```
version: '3'

services:
  php-apache-comilones:
    container_name: php-apache-comilones
    build: .
    restart: always
    volumes: ['./www:/var/www/html']
```

```
    ports:
      - '80:80'
    links:
      - db-comilones


  db-comilones:
    container_name: db-comilones
    image: mysql:8.0
    restart: always
    volumes:
      - ./db_data:/var/lib/mysql
      - ./sql:/docker-entrypoint-initdb.d
    ports:
      - '3306:3306'
    command: --default-authentication-plugin=mysql_native_password
    environment:
      MYSQL_DATABASE: comilones
      MYSQL_USER: user
      MYSQL_PASSWORD: password
      MYSQL_ROOT_PASSWORD: password


  phpmyadmin-comilones:
    depends_on:
      - db-comilones
    container_name: phpmyadmin-comilones
    image: phpmyadmin/phpmyadmin
    restart: always
    ports:
      - '8088:80'
    links:
      - db-comilones
    environment:
      PMA_HOST: db-comilones
      MYSQL_ROOT_PASSWORD: password
```

ANALISIS DE UN *SPRINT*

Vamos a detallar la metodología realizada durante los *Sprints*. A modo de ejemplo, analizaremos por encima las tareas detalladas del Backlog para el primer *Sprint*.

Descripción del esquema:
La primera columna es el numero de Historia de Usuario.
La segunda columna son las tareas por historia de usuario.
La tercera columna una descripción de las tareas.
La cuarta columna el tiempo estimado de realización de las tareas.

US0	T0.1	Crear repositorio a GitHub @cifpfbmoll, privado. Ir poniendo código, Issues (una por tarea), y proyectos de tipo "Basic kanban" (uno por Sprint)	1
US1	T1.1	Instalar Docker	1
	T1.2	Crear Dockerfile con imagen de PHP + Apache, con librerías de MySQL.	1
	T1.3	Crear docker-compose.yml un servicio para utilizar el Dockerfile y desplegar la carpeta del proyecto: www.	1
US2	T2.1	Al docker-compose.yml añadir un servicio con un contenedor de MySQL	1
US3	T3.1	Al docker-compose, al servicio MySQL, cargar el archivo sql de la carpeta "sql" para que se cargue automáticamente.	1
	T3.2	Crear el fichero sql con una tabla de restaurantes mínima.	0,5
US4	T4.1	Al docker-compose, añadir un servicio con un contenedor de PhpMyAdmin	0,5
US5	T5.1	Explicar al README.md como se puede entrar en el contenedor Mysql y al servicio mysql dentro.	1
US6	T6.1	Crear fichero index.html con plantilla de Bootstrap 4..	1
	T6.2	Crear logo de la aplicación.	1
	T6.3	Crear cabecera de index.html, poniendo logo y título de la web.	0,5

US7	T7.1	Añadir al pie del frontend el copyright, nombre de la empresa desarrolladora y año de la aplicación.	1
US8	T8.1	Añadir al frontend un menú superior para acceder a las diferentes secciones de la aplicación. Por ejemplo, con un navbar de Bootstrap	1
US9	T9.1	Con bootstrap, crear 5 elementos para contener datos del restaurante. Por ejemplo, tipo Card.	1
	T9.2	Añadir en el fichero sql una tabla de restaurantes con todos los campos necesarios.	0,5
	T9.3	Realizar los selects necesarios para la tabla de restaurantes desde el código PHP	1,5
	T9.4	Mostrar a los bootstrap de la página restaurante los 5 restaurantes encontrados. Por el momento, basta mostrar el nombre del restaurante.	1,5
US10	T10.1	Añadir un formulario en la página principal del frontend, que permita introducir la búsqueda por nombre de restaurante, localidad, índice de precios y tipos de cocina. ejemplos:	2
	T10.2	Capturar los parámetros devueltos por el formulario y pasarlos a la función de búsqueda de restaurantes.	0,5
	T10.3	Modificar el select en la tabla de restaurantes para que tenga en cuenta los parámetros del buscador.	1
US11	T11.1	Añadir a buscador de la página principal del frontend un desplegable para seleccionar ordenación. ejemplos:	0,5
	T11.2	Capturar los parámetros devueltos por el formulario y pasarlos a la función de búsqueda de restaurantes.	0,5
	T11.3	Modificar el select en la tabla de restaurantes para que ordene según el parámetro devuelto por el buscador.	1
US12	T12.1	Asegurarse de que el select los restaurantes devuelve de cada restaurante: nombre, localidad, índice de precios, valoración, teléfono, fotografía principal	0,5
	T12.1	A los elementos Card de la página principal del buscador, mostrar de cada restaurante: nombre, localidad, índice de precios, valoración, teléfono, fotografía principal	2

US13	T13.1	Añadir un select por ID del restaurante que devuelva tota la información del restaurante.	2
	T13.2	Cuando hagamos clic sobre el nombre del restaurante o botón con enlace de la página principal, mostrar una página restaurant.php que muestre el nombre del restaurante.	0,5
US14	T14.1	En la ficha completa del restaurante del frontend, para cada restaurante debemos mostrar, además de su nombre: localidad, dirección, código postal, teléfono, índice de precios (de 1 a 5 euros), valoración (de 1 a 5 estrellas), email, página web, horario y tipo de cocina (puede tener más de un tipo de cocina)	3
US15	T15.1	En la ficha completa del restaurante del frontend, para cada restaurante debemos mostrar un mapa de Google Maps o similar con su localización	2
US16	T16.1	En la ficha completa del restaurante del frontend, para cada restaurante debemos mostrar fotografías del mismo. Una de ellas será la principal	2
		TOTAL HORES	34,5

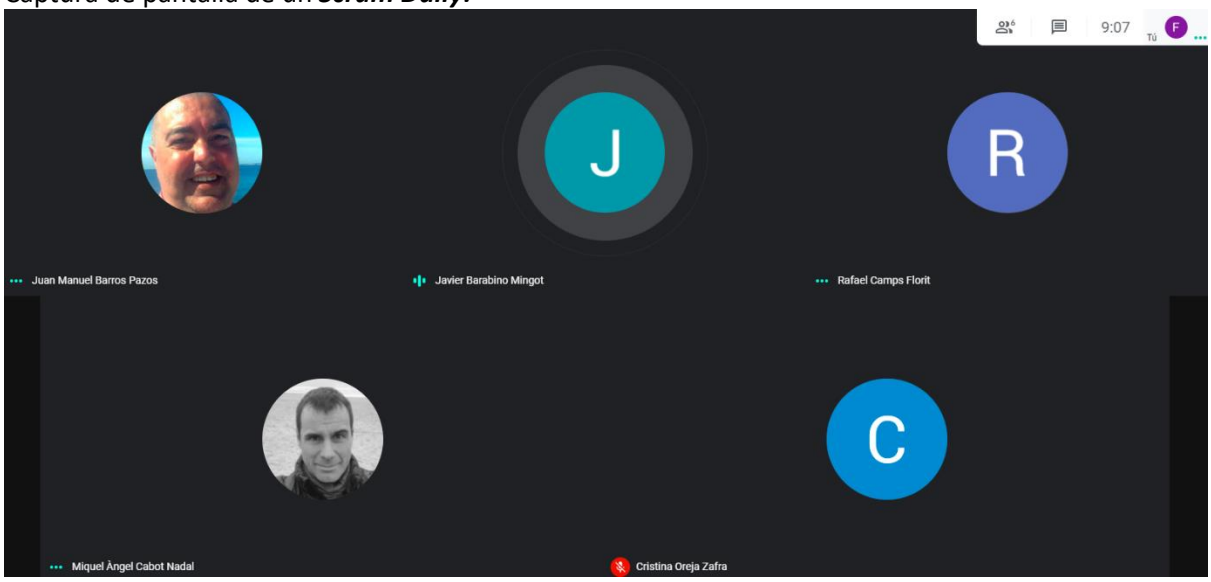
Análisis:
Cada ***Development Team*** tiene la libertad de elegir las tareas que le convengan según las sugerencias del ***Product Owner*** para llegar a tiempo con una versión funcional al final del Sprint, generalmente transmitidas en cada ***Scrum Daily.*** Así, las historias no se han de elegir necesariamente en su orden, sino en grado de prioridad. Mientras va avanzando el Sprint uno comprueba las necesidades básicas para el proyecto llegue funcional, los problemas que van surgiendo, y adapta el tiempo dispuesto para cada tarea según las incidencias diarias.

SPRINT DAILY

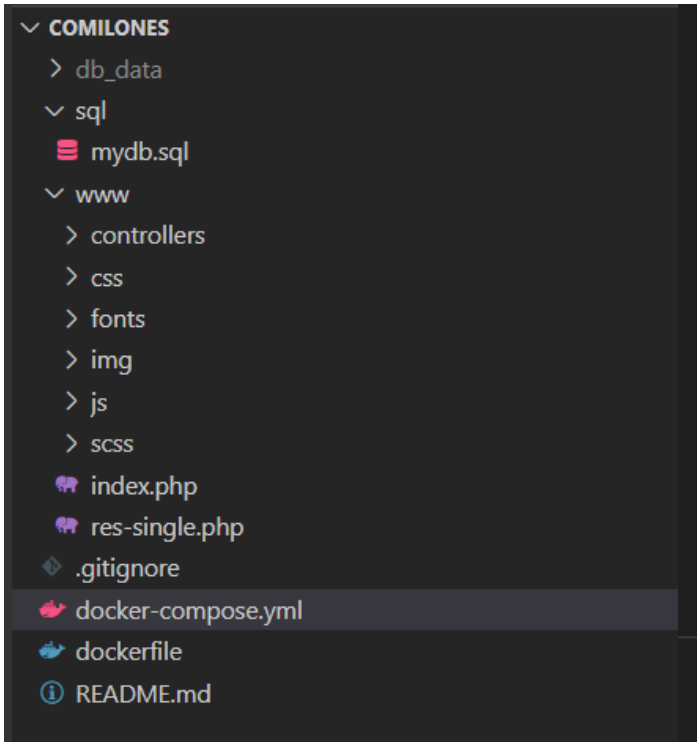
El Sprint *daily* se realizaba cada día a las nueve de la mañana. El Scrum master rotativo controlaba la sesión y disponía de las soluciones que pudiesen estar en su mano ante los problemas del *Development Team*. En este caso al ser el *Product Owner* nuestro profesor no teníamos inconvenientes de recibir rápido asesoramiento ante cualquier duda o inconveniente.

Ha sido una experiencia muy interesante porque te ayuda a sincretizar bien tus problemas. Se tiene que expresar en muy poco tiempo lo que se ha realizado, los problemas que nos hayan surgido, y lo que se pretende hacer en ese día. Este ejercicio te ayuda a concretar y expresar mejor tus ideas. Y de forma análoga, a ser más práctico y resolutivo en tu trabajo.

Captura de pantalla de un ***Scrum Daily:***



ESTRUCTURA DE CARPETA DEL PROYECTO

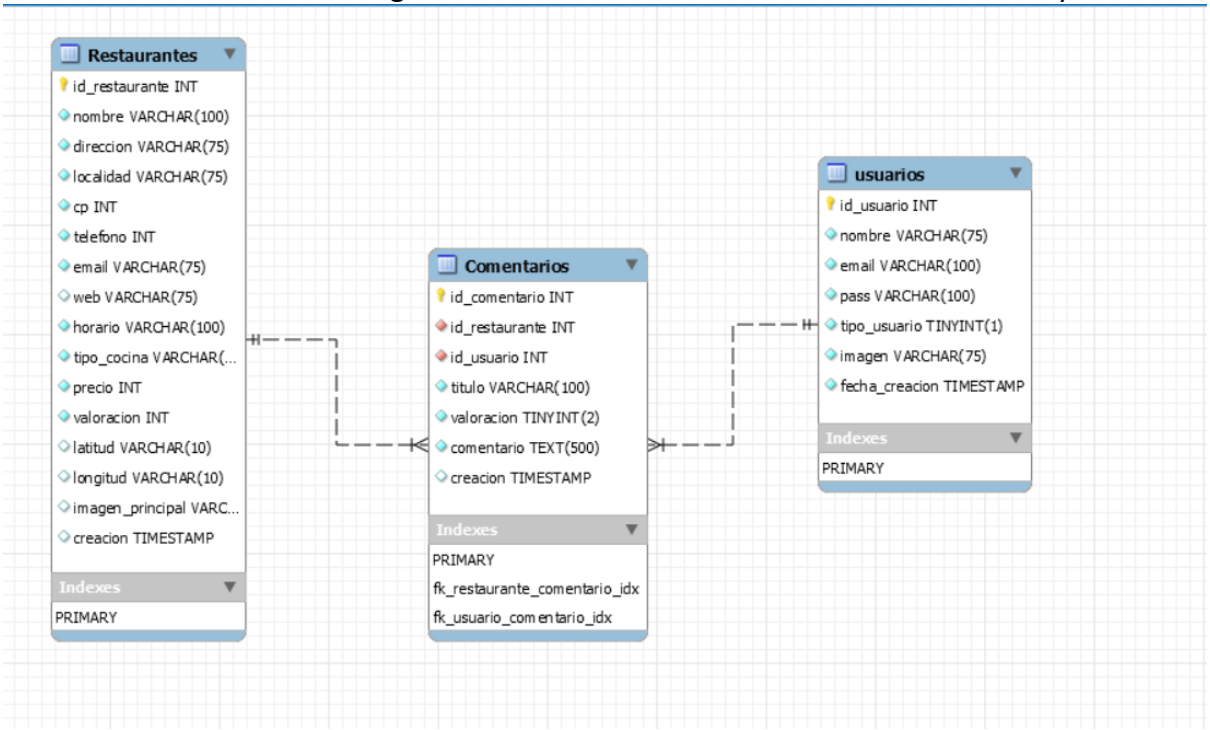


- **db_data**: carpeta para la persistencia de los datos.
- **Sql**: Modelo de la base de datos

- **www**:
    - **controllers**: archivos de control entre las vistas y el modelo de datos
    - **css**: archivos de hojas de estilo del proyecto
    - **fonts**: fuentes del proyecto
    - **img**: imágenes del proyecto
    - **js**: Archivos Javascript del proyecto
    - **scss**: Scrips *Sass* del proyecto
- index.php: Página principal del proyecto
- res-single.php: Pagina que genera las vistas individuales de los restaurantes
- .gitignore: Archivos ignorados en el control de versiones: db_data.
- Docker-compose.yml: Archivo para definir y ejecutar las aplicaciones Docker de los contenedores.
- Dockerfile: documento de texto que contiene todos los comandos necesarios para ensamblar imágenes Docker.
- Descripción de la instalación del proyecto. También se describe como acceder al contenedor Mysql por línea de comandos.

ESQUEMA DE LA BASE DE DATOS

A continuación mostramos el gráfico entidad-relación de la base de datos del Proyecto.



COMENTARIO AL CODIGO FUENTE

El código fuente no está estructurado y cuidado. La razón por la cual no se ha tenido en cuenta las buenas prácticas de programación **es la falta de tiempo**. Sin que ello deje de ser importante se tomó la decisión de realizar **un código funcional** para poder presentar a tiempo. En el tiempo que una empresa genera un *Sprint*: 2 semanas, nosotros hemos realizado 3.

Algunos ejemplos:
Aquí comprobarán que tenemos código php embebido dentro del html en el archivo **index.php**:

```
<!-- RESTAURANTS ITEMS -->



       <?php while ($value = $result->fetch_array(MYSQLI_ASSOC)) {

         // isprecio activated?
         if (isset($_GET['precios']) && ($_GET['precios'] != "Precios")) {

           if ($_GET['precios'] == "Más de 50 euros") {

             if (intval($value['precios']) > 50) {
               $ok = true;
             } else {
               echo $value['precios'] . " ";
               $value['tipo_cocina'] = 'anulado';
               $value['valoracion'] = 'anulado';
               $ok = false;
             }
           } elseif ($_GET['precios'] == "Hasta 20 euros") {

             if (intval($value['precios']) <= 20) {
               $ok = true;
             } else {
               echo $value['precios'] . " ";
               $value['tipo_cocina'] = 'anulado';
               $value['valoracion'] = 'anulado';
               $ok = false;
             }
           } elseif ($_GET['precios'] == "Entre 20 y 50 euros") {
```

```
              if (intval($value['precios']) > 20 && intval($value['precios'])
<= 50) {
                $ok = true;
              } else {
                echo $value['precios'] . " ";
                $value['tipo_cocina'] = 'anulado';
                $value['valoracion'] = 'anulado';
                $ok = false;
              }
            }
          }

            // is_cocina activated?
            if (isset($_GET['tipo_cocina']) && ($_GET['tipo_cocina'] != "Coci
na")) {

              if ($value['tipo_cocina'] == $_GET['tipo_cocina']) {
                $ok = true;
              } else {
                $ok = false;
              }
            }

          // isvaloracion activated?
          if (isset($_GET['valoracion']) && ($_GET['valoracion'] != "Estrellas
")) {

            if ((intval($value['valoracion'])) == (intval($_GET['valoracion'])
)) {
              $ok = true;
            } else {


              $value['tipo_cocina'] = 'anulado';

              $ok = false;
            }
          }




          if ($ok) {
        ?>
            <!-- tarjet Restaurant -->
            <div class="item">
              <div class="block-19 ">
                <figure>
```

```
                  <img class="img-
fluid" src="<?= $value['foto'] ?>" alt="Foto de <?= $value['name'] ?>">
                </figure>
                <div class="text">
                  <h2 class="heading"><a href="res-
single.php?id=<?= $value['id'] ?>"><?= $value['name'] ?></a></h2>
                  <p class="mb-4">Localidad: <?= $value['localidad'] ?>.</p>
                  <p>Tipo de cocina: <?= $value['tipo_cocina'] ?>.</p>
                  <div class="meta d-flex align-items-center">
                    <div class="number">
                      <span> Coste por persona: <?= $value['precios'] ?> </spa
n>
                    </div>
                    <div class="price text-
right"> Valoracion <span><?= $value['valoracion'] ?></span></div>
                  </div>
                </div>

              </div>

            </div>

        <?php  }
        } // fin del bucle.
        // Close and free result
        $result->close();

        mysqli_close($conn);  ?>



        <!-- FIN RESTAURANTS ITEMS -->
```

Comprueben además que se realiza el cierre de la conexión de la base de datos dentro del mismo archivo **index.php.**

```
        <?php  }
        } // fin del bucle.
        // Close and free result
        $result->close();

        mysqli_close($conn);  ?>
```

Otro detalle es que no se ha generado la modularidad del Modelo Vista Controlador. Es decir, es el mismo archivo controlador el que realiza la conexión y control de datos:

index_controller.php:

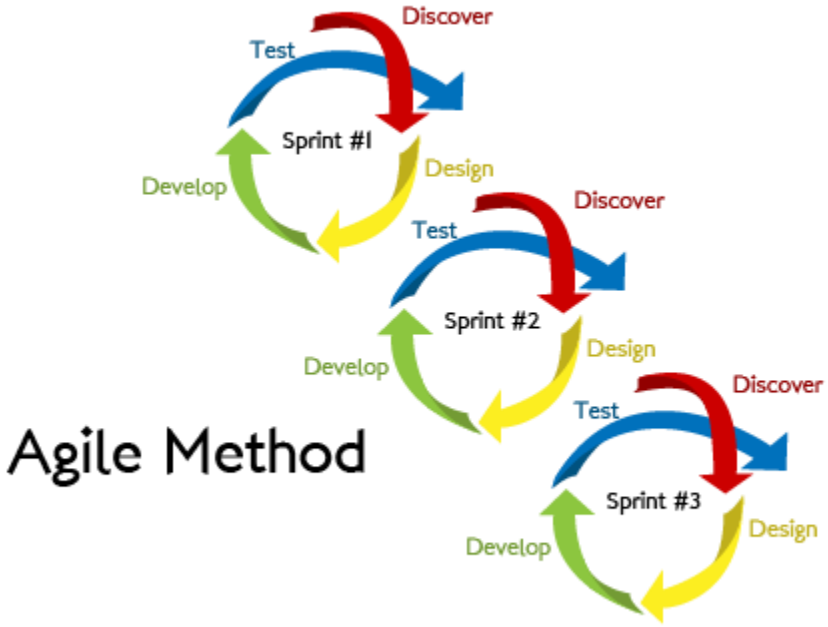
```php
<?php

$host = 'db-comilones';
$restaurante = 'user';
$password =  'password';
$db = "comilones";

// Connection to DB
$conn = mysqli_connect($host, $restaurante , $password, $db );

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}


    // funcion para el formulario de filtrado
    function filtrado_restaurantes($conn) {

        global $conn;

        $select ="SELECT * FROM restaurantes";

        // search by name
        if (isset($_GET['name']) && !empty($_GET['name'])) {

            $select = $select. " WHERE name like '%".$_GET['name']."%'";

        // search by localidad
        } elseif (isset($_GET['localidad']) && !empty($_GET['localidad'])) {


            $select = $select. " WHERE localidad like '%".$_GET['localidad']."
%'";

        }

         return  mysqli_query($conn, $select);


    }
```

```php
if(isset($_GET['submit']))  {

    $result = filtrado_restaurantes($conn);


} else {


$query = 'SELECT * From restaurantes';
$result = mysqli_query($conn, $query);

}
?>
```

res_single_controller.php:

```php
<?php

$host = 'db-comilones';
$restaurante = 'user';
$password =  'password';
$db = "comilones";

// Connection to DB
$conn = mysqli_connect($host, $restaurante , $password, $db );

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

$id = $_GET['id'] ;

$query = "SELECT * From restaurantes WHERE id='$id' LIMIT 1";
$result = mysqli_query($conn, $query);

$restaurante = mysqli_fetch_array($result);
```

Sin duda todos estos archivos han de ser factorizados convenientemente testeados.



Se podría decir que en mis tres Sprints, por motivos de urgencia, el equipo de desarrollo a deseado sacrificar el testeo y la refactorización para llegar a tiempo al cliente. En Sprints posteriores nos podríamos a trabajar sobre estos puntos esenciales en todo Proyecto.

INSTALACIÓN DE LA APLICACIÓN

La instalación de la aplicación es muy sencilla al estar "dockerizada":

1. Descargar el proyecto de su dirección en *github*: https://github.com/cifpfbmoll/comilones.git

2. Entrar en el proyecto y colocarse en la raíz. Ahí está el archivo Docker-compose.yml. Activar los contenedores con la instrucción: docker-compose up ó docker-compose up -d si deseamos que se ejecute en segundo plano

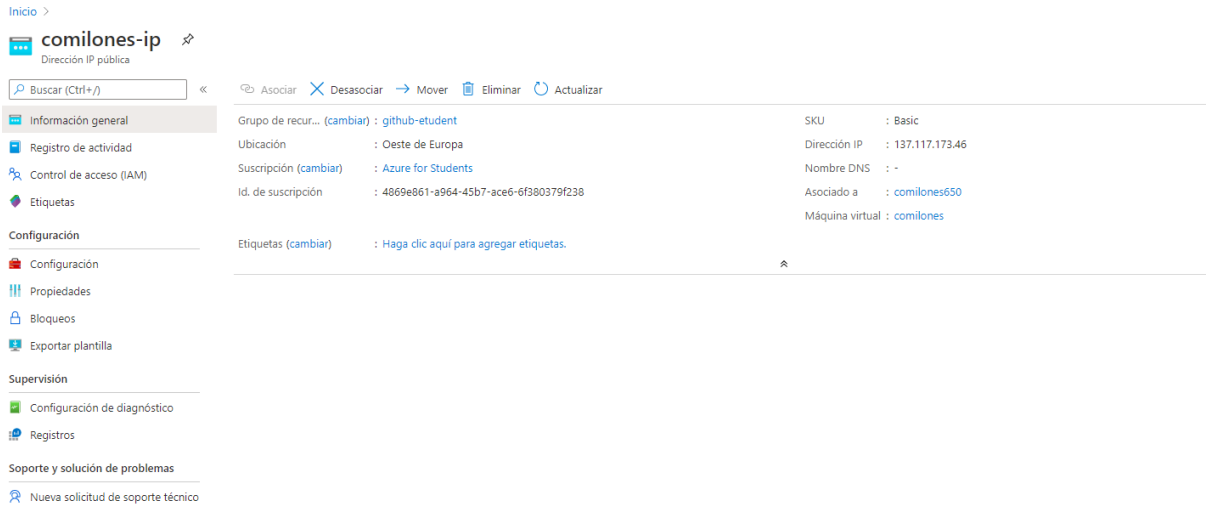
3. Entrar por el puerto 80.

4. Si da un error de conexión, esperar unos minutos para que se generen las tablas en la Base de datos.

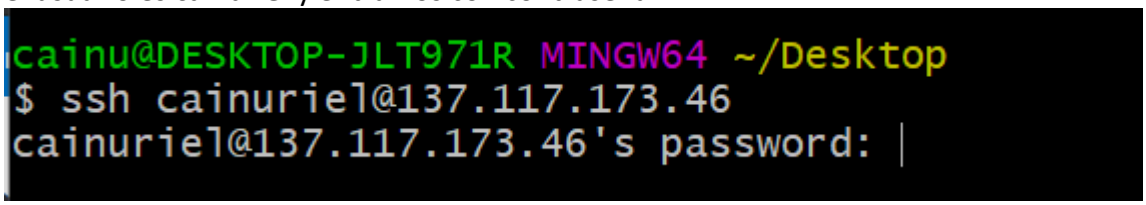
DESPLIEGUE EN SERVIDOR

Hemos seleccionado la forma de acceso a través del ***Bash*** de ***Git*** en detrimento de ***Putty***.

Anteriormente henos alquilado un servicio de máquina virtual en Microsoft Azure para desplegar nuestra aplicación web en ella:

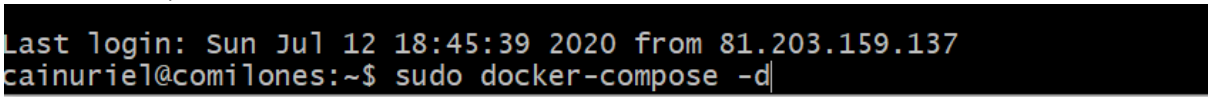


el usuario es **cainuriel** y entramos con contraseña:



Dentro de la kernel de la maquina virtual realizamos el clonado del proyecto:

Y finalmente, levantamos los contenedores:



Y listo. Ya podemos acceder a través de la ip: 137.117.173.46.

Solo queda contratar un dominio. En nuestro caso hemos decidido utilizar name.com para ello. El nombre del dominio será **www.comilones.company**

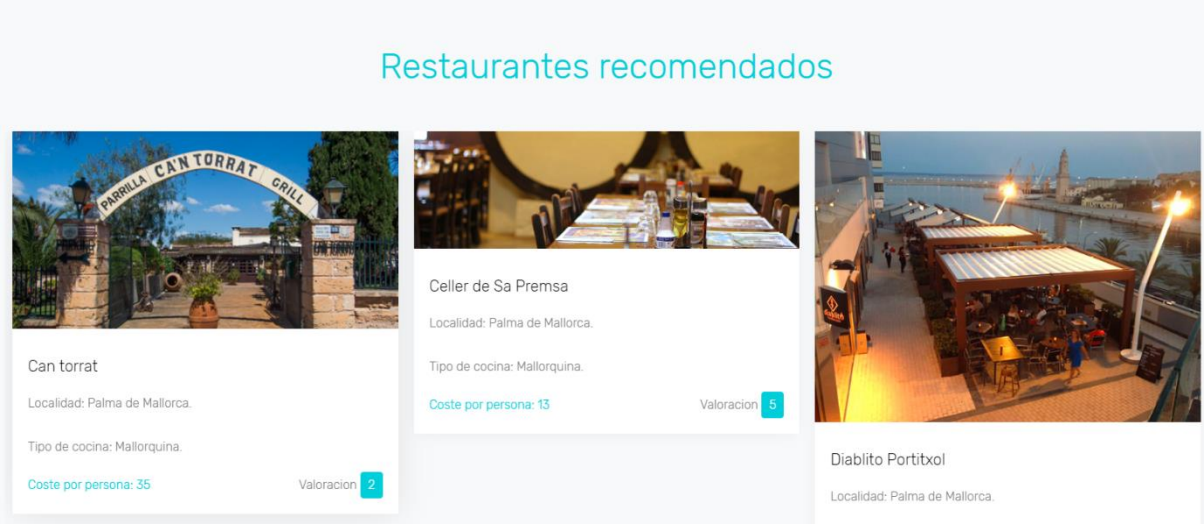
DIRECCIÓN DEL PROYECTO

http://51.178.152.198/index.php

REPOSITORIO EN GIT

https://github.com/Cainuriel/comilones

CAPTURAS DE LA APLICACIÓN



Comilones

Comilones no se hacen responsable de lo que opinen los comilones. Sobretodo si escriben con la boca llena

Menu rápido

Home
Sobre Nosotros
Blog de cocina
Contacto

Blog de cocina

Comida sana rápida
May 29, 2018 Admin 19

Recetas Post-covid
May 29, 2018 Admin 19

Cocina Mallorquina: La Sopa de Nadal
May 29, 2018 Admin 19

Otras vias de contacto

Calle Manacor nº 43 1º B, Palma de Mallorca, Islas Baleares, España

+34 622 646 626

flopez@cifpfbmoll.eu

Detalles del Restaurante

- Valoracion: 5
- Telefono: 666666666
- Localidad: Palma de Mallorca
- Tipo de cocina: Mallorquina
- Precios: €13 por persona.

Reserva tu Mesa

Llama ahora

Horario cenas: 2030h - 2300h.

Horario comidas: 1300h - 1600h

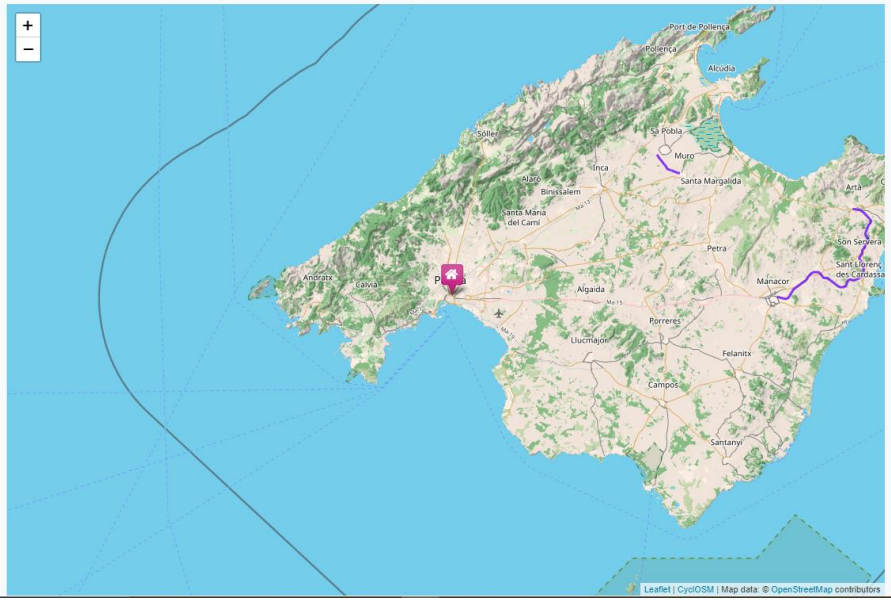
Nombre

Introduzca su nombre

correo electrónico

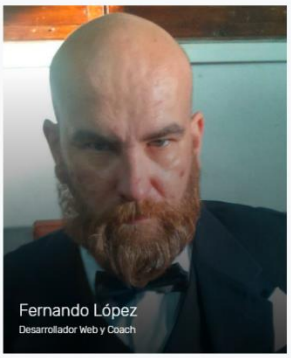
Descripción

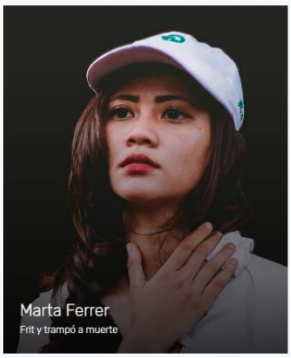
Los inicios de El Celler Sa Premsa se remontan a mayo de 1958 cuando Antonio Mayol, banquero de formación, apostó por convertir un antiguo almacén en el centro de Palma de Mallorca en un acogedor restaurante. Sin conocimiento de restauración ni hostelería, decidió crear un espacio donde quien quisiera pudiera disfrutar de la comida mallorquina, sin importar su procedencia, sus gustos o inquietudes. El boca oreja durante los primeros años propició el éxito del restaurante e hizo que fueran muchos los mallorquines que hicieran una parada en el Celler Sa Premsa. Una tradición que fue también transmitida a los turistas de la isla logrando que, tanto para locales como para visitantes, degustar una comida en el Celler Sa Premsa se convirtiera en una visita obligada. Más de medio siglo después, el sueño de Antonio sigue siendo una realidad y el restaurante una parada obligada. De la mano de la tercera generación familiar, El Celler Sa Premsa no ha perdido su esencia: el precio, la calidad y el servicio. La fórmula del éxito del negocio desde sus inicios.

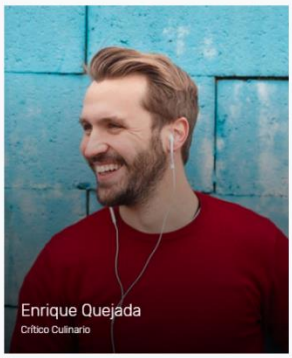


Comentarios Clientes

Para poder dejar comentarios tiene que darse de alta como usuario







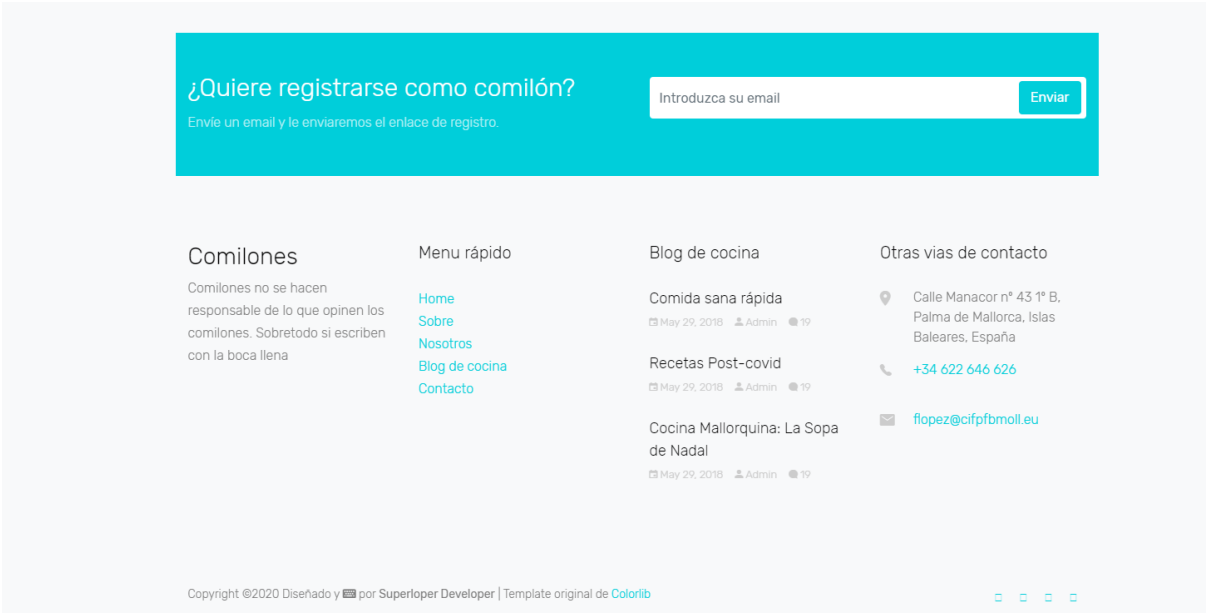

¿Quiere registrarse como comilón?
Envíe un email y le enviaremos el enlace de registro.
Introduzca su email
Enviar
Comilones
Comilones no se hacen responsable de lo que opinen los comilones. Sobretodo si escriben con la boca llena
Menu rápido
Home
Sobre
Nosotros
Blog de cocina
Contacto
Blog de cocina
Comida sana rápida
May 29, 2018 Admin 19
Recetas Post-covid
May 29, 2018 Admin 19
Cocina Mallorquina: La Sopa de Nadal
May 29, 2018 Admin 19
Otras vias de contacto
Calle Manacor nº 43 1º B, Palma de Mallorca, Islas Baleares, España
+34 622 646 626
flopez@cifpfbmoll.eu
Copyright ©2020 Diseñado y por Superloper Developer | Template original de Colorlib

Tabla de contenido