Universitat Oberta de Catalunya

uoc.edu

Web corporativa e intranet para una consultora tecnológica

Pablo Sánchez Ponce
Grado de Ingeniería Informática
Desarrollo web

Gregorio Robles Martínez
Santi Caballé Llobet

12 de junio de 2020

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Web corporativa e intranet para una consultora tecnológica*
Nombre del autor:	*Pablo Sánchez Ponce*
Nombre del consultor/a:	*Gregorio Robles Martínez*
Nombre del PRA:	*Santi Caballé Llobet*
Fecha de entrega (mm/aaaa):	06/2020
Titulación::	*Grado de Ingeniería Informática*
Área del Trabajo Final:	*Desarrollo web*
Idioma del trabajo:	*Castellano*
Palabras clave	*Web intranet consultora*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

El objeto de este trabajo es proveer a una empresa consultora de informática una nueva página web corporativa que mejore la ya existente en una mejor presentación, más facilidad de mantenimiento, mayor flexibilidad de actualización para el futuro y una gestión dinámica de las reservas de los cursos de formación por parte de los clientes.

Para tal efecto, se opta por un diseño que utiliza Python y Flask para el código, además de HTML5 y CSS. Así mismo, se opta por una base de datos relacional para la salvaguarda de los datos, implementándose con un servidor de MySQL.

El resultado es una página web funcional, con una imagen corporativa y fácilmente migrable a otros entornos, con una gestión dinámica de los cursos de formación, profesorado y reservas de los cursos.

El uso de Python ha permitido profundizar en un lenguaje menos habitual que otros más utilizados dentro del desarrollo web, como PHP o Javascript, y que gracias al framework Flask presenta una mayor facilidad de desarrollo unida a la versatilidad y potencial que tiene Python.

Así mismo, la traducción de los requisitos iniciales al producto final ha requerido un diseño previo y una búsqueda de las mejores soluciones para cumplir con las necesidades indicadas.

Abstract (in English, 250 words or less):

The purpose of this project is to provide a new corporate website to an IT consultancy company. This business has already a website but they need to improve it in order to get a better look, easy maintenance tasks, more flexibility in terms of updates and a dynamic management for the online training courses and the bookings done by the customers.

To achieve all the goals and the requirements, the solution will be based on Python and Flask for the code (also HTML5 and CSS). A relational database (MySQL) will be used to store the data permanently.

The final delivery is a functional corporate website, easily migrable between different environments which allows to manage dinamically all the trainings courses, teachers and bookings.

Using Python as code language for the development has been a good option to get deeper in a language used for different purpose and not being the most typical for web development like PHP or Javascript. The usage of the Flask framework has been a good choice to develop website instead of using Python from the scratch but being able to get all the potencial of this powerful language.

Also, the challenge to translate all initial requirements into a valid final product makes to spend time doing design tasks and search for the best solutions in order to complain with all the business and technical requirements.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

Una nueva empresa denominada Clever Consulting se ha creado en la zona realizando servicios de consultoría IT: Estrategia, Arquitectura, Proyectos de implantación y Formación. Ante el crecimiento del negocio se han dado cuenta que necesitan una web atractiva para sus clientes y con la funcionalidad de tener un área de clientes donde además puedan comprar directamente los cursos ya definidos dentro del catálogo de formación.

1.2 Objetivos del Trabajo

El objetivo principal de la empresa es crear una web accesible que exponga todos los servicios que realizan y que ofrezca la posibilidad de vender online cursos de formación a los visitantes.

Desgranando este objetivo en sub objetivos tenemos:

- Generación de la web externa/pública.

  Dentro de su web externa, se requiere que se ofrezcan claramente los servicios que ofrecen dentro de sus cuatro categorías (Estrategia, Arquitectura, Proyectos de Implantación y Formación), con un catálogo concreto para los cursos de formación.

- Generación de una web privada para clientes registrados.

  Los clientes se podrán registrar en la web para tener su propia área e incluso contratar los cursos de formación de forma directa. Para ello, se deberá tener en cuenta que habrá profesores asociados a los diferentes cursos y habrá que considerar si están disponibles y mostrarlos a través de un calendario durante el momento de la contratación para evitar solapamientos.

- Diseño accesible y *responsive*.

  Se ha hecho hincapié en que ha de ser una web accesible por cualquier persona y dispositivo, por lo que se necesitará implementar técnicas de diseño *responsive* y de accesibilidad, con una buena experiencia de usuario.

1.3 Enfoque y método seguido

Dado que la empresa ya tiene una web existente se contempla la posibilidad de adaptar la existente. Al ser una web basada en Wordpress y de la que no se quiere reutilizar ningún elemento a excepción de algunos textos se decide empezar una página nueva desde cero por los siguientes motivos:

- Unificación de código respecto etiquetas, estructura o CSS lo que permitirá un producto más fácilmente entendible y sencillo de mantener en el futuro.

- Libertad tecnológica al no depender de un programa de terceros como es Wordpress.

1.4 Planificación del Trabajo

Se utilizará como recurso personal un desarrollador web.

La planificación temporal consta de los siguientes pasos:

Se considera el período del 9 de marzo al 29 de mayo para la finalización de todos los entregables. No obstante, de forma adicional se consideran 10 jornadas adicionales de soporte posterior a la entrega, puesta en producción y validación de la web.

Nota: Por simplicidad y no hacerlo dependiente a una ubicación concreta, no se han considerado los festivos, asumiendo los días laborales de lunes a viernes.

Paso 1: Toma final de requisitos con el cliente → 2 jornadas (9-10 marzo)

Paso 2: *Kickoff* inicial. → 1 jornada (11 marzo)

Paso 3: *Mockup* de la web externa y validación → 4 jornadas (12-17 marzo)

Paso 4: *Mockup* de la intranet y validación → 3 jornadas (18-20 marzo)

Paso 5: Desarrollo de la web externa → 25 jornadas (23 marzo – 24 abril)

Paso 6: Pruebas de la web externa → 2 jornadas (27-28 abril)

Paso 7: Validación por parte del cliente de la web externa → 2 jornadas (29-30 abril)

Paso 8: Desarrollo del área de clientes → 13 jornadas (1-19 mayo)

Paso 9: Pruebas del área de clientes → 3 jornadas (20-22 mayo)

Paso 10: Validación por parte del cliente del área de clientes y del proyecto → 5 jornadas (25-29 mayo)

Paso 11: Documentación y memoria del proyecto. → 10 jornadas (1-12 junio)

Dentro de la planificación del trabajo se ha realizado la siguiente evaluación de riesgos:

Riesgo 1: Retraso en el desarrollo por causas imprevistas

Mitigación: Durante la planificación temporal se ha utilizado una estrategia conservadora en las jornadas necesarias para la realización de cada fase. En caso de alguna excepcionalidad más crítica se replanteará el retraso en la entrega de la intranet, que tiene un impacto menor para el cliente.

Riesgo 2: Retraso en el plan por cambios de los requisitos por parte del cliente.

Mitigación: Durante la planificación temporal se ha utilizado una estrategia conservadora en las jornadas necesarias para la realización de cada fase. Así mismo, en los acuerdos se recoge el alcance confirmado por ambas partes para evitar cambios de alcance y se hará hincapié durante el kickoff inicial. De esta manera, los cambios de alcance se tratarán de forma excepcional con posibilidad de generar un proyecto nuevo alternativo.

Riesgo 3: La tecnología escogida finalmente no encaja con los requisitos del cliente.

Mitigación: Se realizaran pruebas de validación antes de la entrega y se plantearan escenarios alternativos para encontrar una solución más adecuada a las necesidades del cliente.

1.5 Breve sumario de productos obtenidos

Código web en Python que hace uso del framework flask (necesario para su ejecución) que contiene los productos requeridos:

- Web externa
- Web interna de clientes con capacidad de reserva de los cursos disponibles.
- Panel de administración para la gestión de cursos, profesores y reservas.

1.6 Breve descripción de los otros capítulos de la memoria

A lo largo de la memoria introduciremos diferentes aspectos del trabajo, desde la fase inicial hasta el producto final.

Como primer punto, comentaremos las tecnologías que se han decidido utilizar para el diseño y ejecución del proyecto a lo largo de todo su ciclo.

Posteriormente se describen los requisitos del proyecto, que marcarán el diseño del producto para poder cumplir con estos. Una lista adecuada de requisitos antes de comenzar con el diseño permitirá una implementación sin correcciones y cambios en el alcance del proyecto, ya sean requisitos técnicos o funcionales.

Una vez considerados todos los requisitos de la aplicación incidiremos en el diseño realizado como paso previo a la implementación del proyecto, tanto en el aspecto visual como en la arquitectura de la aplicación o el modelado de la base de datos.
Por último, abordaremos el desarrollo del proyecto, que por su complejidad será el apartado más largo, ya que incluye desde la implementación hasta las diferentes pruebas y validaciones realizadas hasta conseguir el producto final deseado, con la estabilidad y todas las funcionalidades especificadas en los requisitos iniciales.

2. Tecnologías utilizadas

El código web se ha realizado con Python y el framework Flask utilizando como IDE para su desarrollo PyCharm y Amazon Web Services (AWS) para su alojamiento. Como base de datos se ha utilizado SQL Alchemy con MySQL de backend.

Las versiones específicas utilizadas han sido:

- Python 3.8.1 [1]

  Python es un lenguaje de programación interpretado y de propósito general. Una de sus características es que por diseño fuerza la obtención de un código fácilmente leíble y entendible. Soporta la programación orientada a objetos y la programación estructurada entre otros.

  Por diseño es altamente modular, lo que permite ir importando los módulos necesarios para la aplicación concreta y añade múltiples funcionalidades más allá del *core* propio del lenguaje.

- Flask framework 1.1.2 [2] [3]

  Flask es un entorno de trabajo para aplicaciones web escrito en Python y desarrollado a partir de herramientas como Werkzeug y Jinja2.

  Una de sus principales características es su diseño altamente modular y por ello se considera un micro *framework*, dejando a la elección del programador cada componente necesario, como puede ser la capa de abstracción de datos, validación o autenticación.

  Esta aproximación minimalista es el principal motivo de su uso frente a su principal competidor, Django, que ya incluye de forma predefinida todos los principales subcomponentes.

- PyCharm Professional 2019.3 [4]

  PyCharm es un IDE para el desarrollo en Python compatible con los principales *frameworks* de desarrollo web, entre los que se incluye Flask.

  El uso del programa facilita un entorno completo de desarrollo y pruebas, ya que permite ejecutar el código, comprobar las respuestas del servidor web, realizar el debug oportuno, configurar variables de entorno, implementar despliegues a servidores externos o integraciones con repositorios.

- MySQL 5.5 [5]

  MySQL es uno de los principales servidores de base de datos relacionales en el mercado. Utiliza el lenguaje SQL y actualmente es propiedad de Oracle aunque mantiene su uso libre y gratuito bajo licencia GPL.

  Por las características del proyecto, se ha optado por la universalidad de MySQL frente a otras soluciones como PostgreSQL o la complejidad de Oracle SQL Server, así como se han descartado soluciones de bases de datos no relacionales (No SQL), ya que el volumen de datos será pequeño y serán estructurados y poco cambiantes en el tiempo en su estructura.

Como herramientas adicionales se han hecho uso de las siguientes utilidades:

- Draw.io [6] para la realización de los mockup y los esquemas de arquitectura.

  Se ha utilizado la herramienta *online* Draw.io para la creación de los diagramas necesarios. Su uso gratuito y múltiples funcionalidades que incluyen imágenes y figuras para arquitectura IT, UX, diagramas de flujo, UML y muchas más la convierte en una herramienta con un gran potencial.

  Durante el transcurso del trabajo se ha renombrado la web de draw.io a diagrams.net

- Uxpressia [7] para la realización de los mapas de *customer journey* utilizados en la validación del cliente.

  Para la generación de los mapas se podría haber utilizado Draw.io y realizar el diseño paso a paso, pero dado que no existen plantillas para tal uso específico se ha optado por una solución diseñada únicamente para esta función.

  Uxpressia es una solución *online* y gratuita en su plan básico permite crear perfiles de clientes, diferentes escenarios y añadir diferentes canales, pruebas e indicaciones de toda la experiencia del usuario en el *customer journey*.

- Microsoft Word para la documentación.

  MS Word es el editor de texto más utilizado en la actualidad.

3. Requisitos del proyecto

- Web externa (HTML5 / CSS)
    - Interfaz clara y sencilla en inglés con las siguientes secciones:
        - Home
        - Services, que incluirá los siguientes apartados:
            - Strategy
            - Architecture
            - Implementation
            - Training. Este apartado mostrará un contenido dinámico de los cursos que están disponibles desde la base de datos.
        - Customer cases. Aquí se incluirán casos de éxito y referencias de clientes sobre proyectos ya realizados.
        - Contact us. Se establecerá un formulario de contacto, así como otros canales de contacto: dirección de correo electrónico y teléfono.
        - Access (Login/Register). La web ha de permitir el registro y autenticación de los usuarios.

    Los diferentes servicios serán una simple descripción de los mismos, a excepción de la formación que ha de incluir el catálogo de cursos y la posibilidad de su reserva.

    Así mismo, las diferentes secciones han de ser visibles a través de un menú en la parte superior que sea fijo independientemente del *scroll* que se haga en la web. El menú ha de ser simple y no generar confusión en la navegación de la web o del mismo menú.

    Aparte del enlace a las diferentes secciones, se habrá de incluir el logo de la compañía en dicha barra superior.

    Por último, se añadirá una barra al final donde se incluirá el copyright y el eslogan de la empresa: “Intelligent Consultancy for a digital society”

- Intranet / Área de clientes

  Los usuarios se han poder registrar para reservar los diferentes cursos que se ofrecen en el catálogo en las fechas seleccionadas por el propio cliente.

  Los requisitos específicos son los siguientes:

  - Usuarios normales (clientes identificados por su email) que podrán reservar el curso que deseen siempre que esté disponible. El catálogo de cursos será dinámico y ha de mostrar el precio (euros) y la duración (horas). El cliente, previo registro y acceso, podrá seleccionar la fecha indicada y la reserva se hará por semana completa, independientemente de su duración.

  - El mismo cliente no podrá realizar la misma reserva varias veces para evitar posibles errores por su parte, a lo que la web indicará un mensaje de error indicando el motivo.

  - El mismo curso puede ser reservado la misma semana siempre que tenga profesores asociados disponibles y un mismo profesor no puede impartir dos cursos la misma semana.

  - Los profesores serán identificados por su número de identidad (en España el DNI), que será único pero no se restringirá el formato al poder tener profesores de origen extranjero y no residentes en España.

  - Un mismo profesor pueden estar asociado a diferentes cursos.

  - Existirá un usuario administrador que accede a un panel sencillo para administrar el catálogo de cursos, los profesores, las reservas, pudiendo añadir, eliminar o editar todos los elementos.

  - El panel de administración no ha de estar enlazado desde la web externa, para evitar su visibilidad.

  - Se realizarán notificaciones de las reservas al cliente, al administrador de la web y al profesor asignado en la reserva.

4. Diseño del proyecto

4.1 Diseño de la interfaz web

Se realizan varios diseños de mockup y se validan satisfactoriamente con el cliente. Ejemplo:

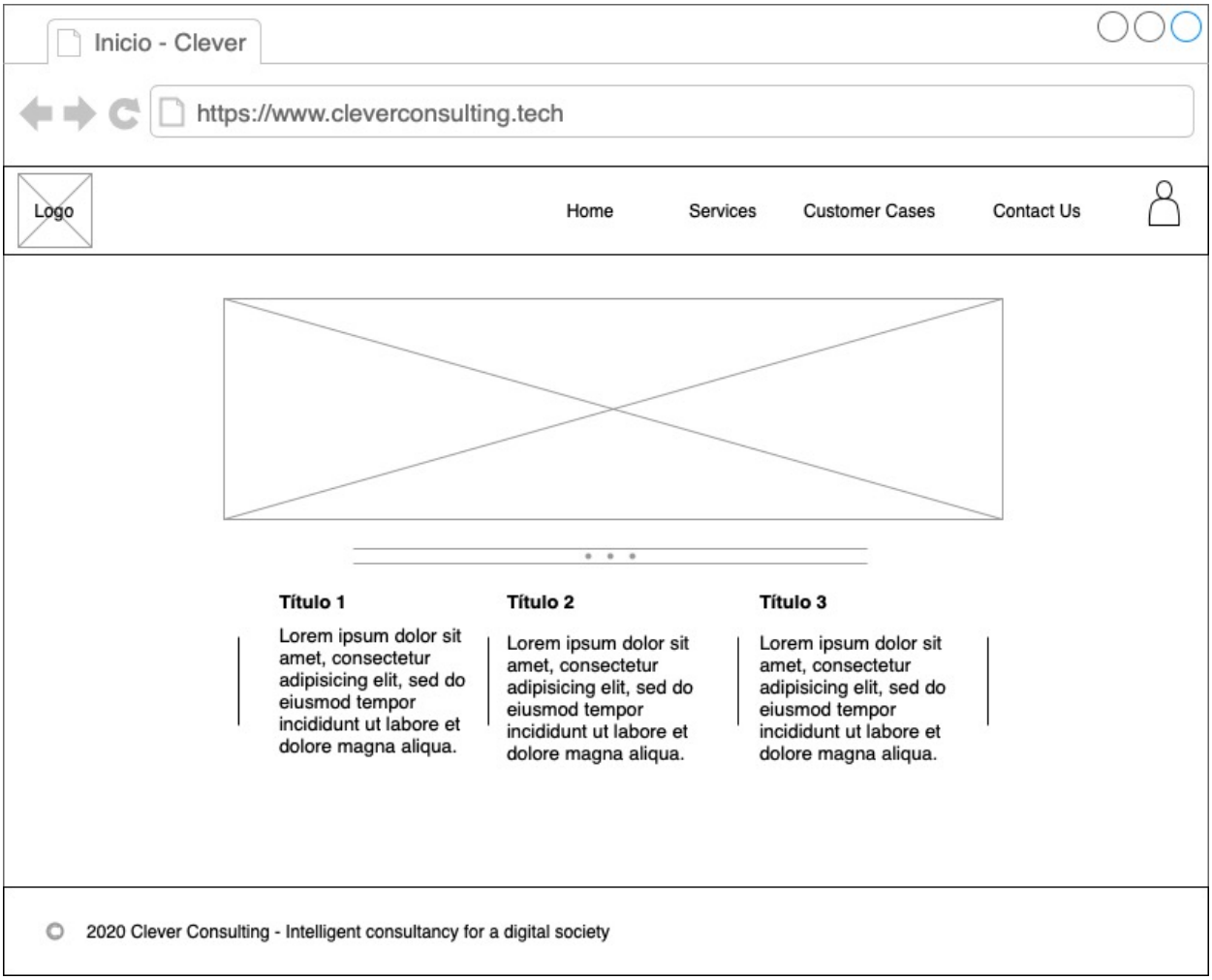


Ilustración 1: Mockup de la web principal

4.2 Arquitectura de la aplicación web

La aplicación constará de un frontend con un servidor web para atender las peticiones HTTP sirviendo HTML y CSS y un backend con un servidor de Flask para la aplicación en Python y un servidores MySQL para la base de datos. La base de datos podrá recibir peticiones tanto del frontend como del backend, aunque no desde el exterior.

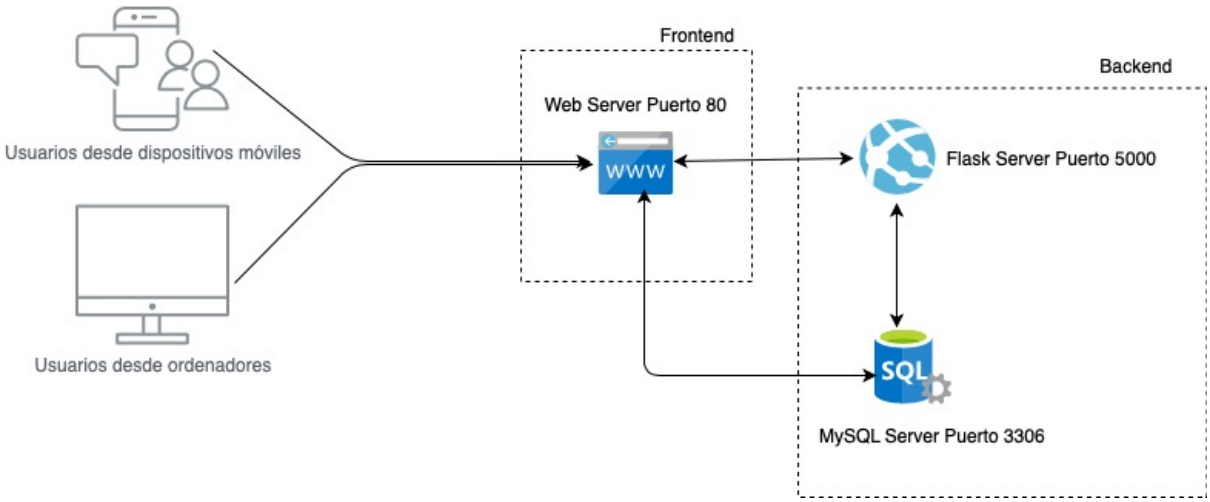


Ilustración 2: Diagrama de arquitectura

4.3 Modelo ER de la base de datos

La aplicación hace uso de las siguientes tablas:

- Users para los usuarios/clientes de la intranet donde el email ha de ser único.
- Courses para el catálogo de cursos.
- Teachers para los profesores de los cursos.
- Coursesteachers para la relación de los profesores asociados a los diferentes cursos.
- Bookings para las reservas realizadas desde la aplicación web.

- De forma auxiliar, la aplicación generará la tabla alembic_version como tabla auxiliar para el uso de SQLAlchemy en Python.

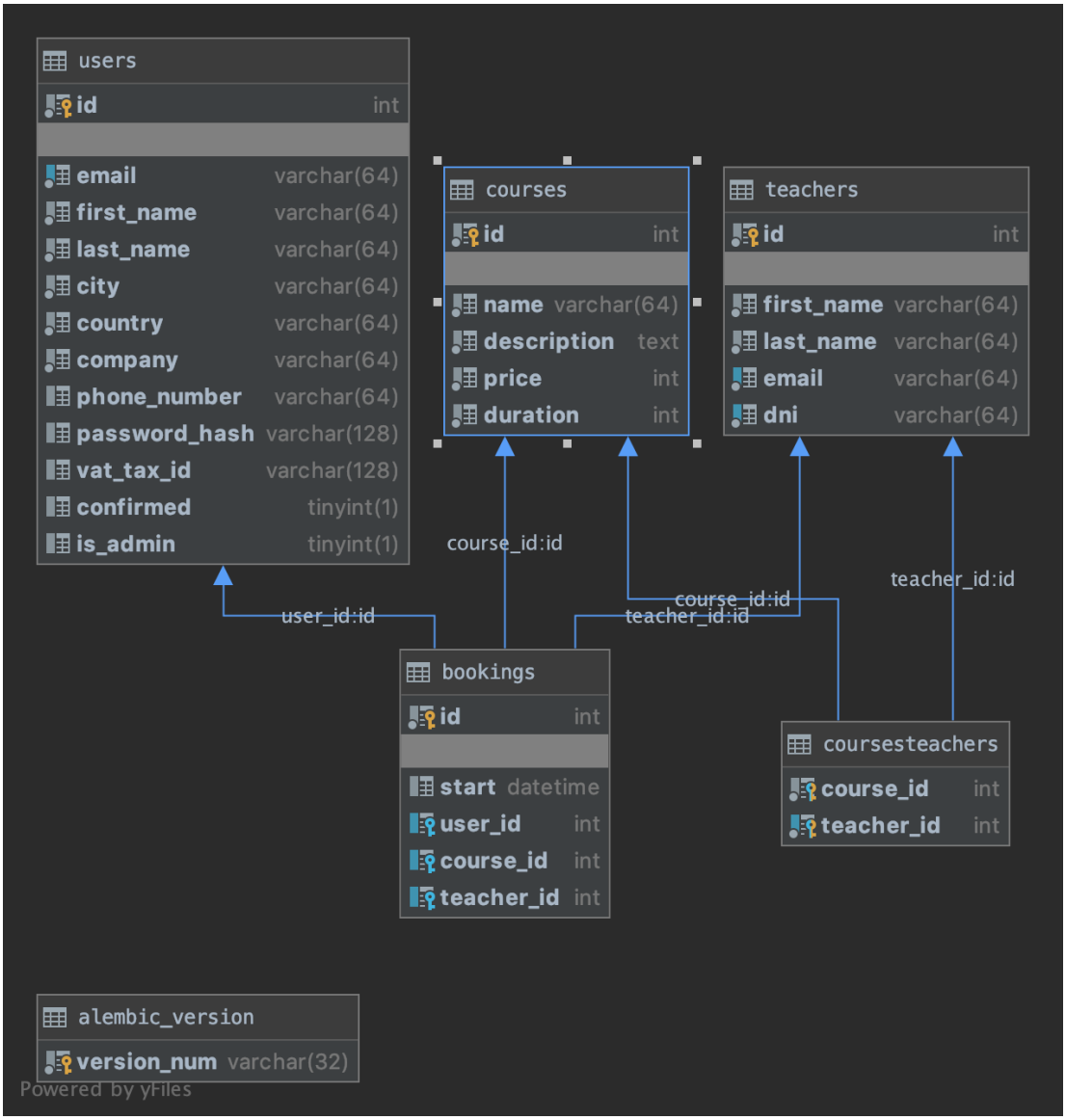


Ilustración 3: Modelo ER de la BBDD

5. Desarrollo del proyecto

5.1 Consideraciones iniciales

Se han cumplido con todos los plazos asignados en el inicio, sin ningún retraso sobre los mismos. La realización ha seguido el mismo orden, desarrollando primero la web externa y posteriormente la intranet.

El desarrollo de la aplicación web se ha realizado tal como se había previsto utilizando HTML5, CSS y Python Flask con PyCharm como IDE tal cómo se había planificado en las tecnologías a utilizar.

Como estructura principal de directorios tenemos la siguiente organización:

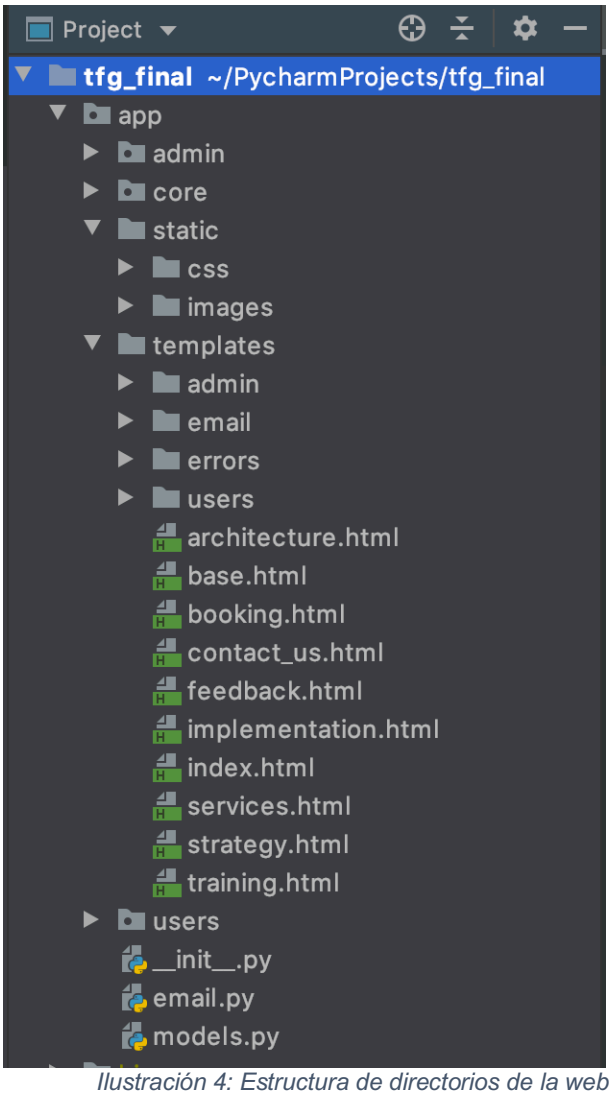


Ilustración 4: Estructura de directorios de la web

Tal como se ha comentado, tanto Python como especialmente Flask, se basan en un diseño muy modular en el que es necesario ir incluyendo los diferentes componentes que necesitemos a nuestro desarrollo en forma de módulos.

La lista de los diferentes requisitos con sus respectivas versiones es la siguiente:

```
alembic==1.4.2
amqp==2.5.2
billiard==3.6.3.0
blinker==1.4
cffi==1.14.0
click==7.1.2
cryptography==2.9.2
dnspython==1.16.0
email-validator==1.1.0
Flask==1.1.2
Flask-Admin==1.5.6
Flask-Celery==2.4.3
Flask-Login==0.5.0
Flask-Mail==0.9.1
Flask-Migrate==2.5.3
Flask-Script==2.0.6
Flask-SQLAlchemy==2.4.1
Flask-WTF==0.14.3
gunicorn==20.0.4
idna==2.9
itsdangerous==1.1.0
Jinja2==2.11.2
kombu==4.6.8
Mako==1.1.2
MarkupSafe==1.1.1
mysql-connector-python==8.0.20
phonenumbers==8.12.3
protobuf==3.11.3
pycparser==2.20
PyMySQL==0.9.3
python-dateutil==2.8.1
python-dotenv==0.13.0
python-editor==1.0.4
pytz==2020.1
rq==1.3.0
six==1.14.0
SQLAlchemy==1.3.16
vine==1.3.0
Werkzeug==1.0.1
WTForms==2.3.1
```

5.2 Desarrollo de la web externa

De acuerdo a la planificación, primero se realizó la página web externa. Para ello, se utilizó una estructura de templates con un *layout* base (base.html) con los componentes comunes a toda la estructura web, como son el menú o las barras de navegación (*footbar*).

Por ello, todo el resto de ficheros .html de la aplicación empiezan con:

```
{% extends 'base.html' %}
```

Dentro del código HTML se categoriza de forma adecuada en clases y div para un posterior uso de CSS de la forma más óptima y granular. Además para añadir una mejor representación gráfica se opta de forma adicional por utilizar CSS de Bootstrap en algunas secciones.

El uso de Bootstrap se decide realizar a través de una llamada directa en vez de instalar los módulos por tener una mayor simplicidad y mejor mantenimiento en el futuro. Esta manera de configuración es lo que llaman BoostrapCDN.

La referencia queda reflejada en el fichero base.html y actualmente se utiliza la versión 4.3.1 de la misma:

```
<!-- Bootstrap CSS -->
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap
.min.css"
      integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
```

Cabe recordar que como base.html es extendido por el resto de templates, este CSS será accesible por los diferentes html que componen la web externa.

Las imágenes se almacenan en el subdirectorio images dentro de static, donde también tendremos cualquier fichero css en el subdirectorio css. De esta forma, la organización será más eficiente y facilita el futuro mantenimiento de modificación de las imágenes o reemplazo de las mismas.

Se opta por un diseño que sea limpio y funcional y que realce los servicios de la consultoría por delante de todo. Para las imágenes de prueba se utiliza un banco externo de imágenes libres de derechos y en algunas secciones se utiliza una técnica de carrusel para poder incluir diferentes fotos que vayan rotando de forma dinámica.

Para el envío de correos, punto que será utilizado tanto por la web externa (formulario de contacto) como por la intranet (registro o reservas) se opta por la utilización del servicio de Sendgrid. De esta forma se consigue una abstracción mayor al ser independiente de servidores de correos locales donde se ejecute la aplicación web.

No obstante, el cambio a un servicio de correo diferente es sencillo ya que el código de envío de emails no deja de ser la generación del mensaje y el envío por SMTP.

Se incluyen a continuación algunas capturas de pantalla de la web:

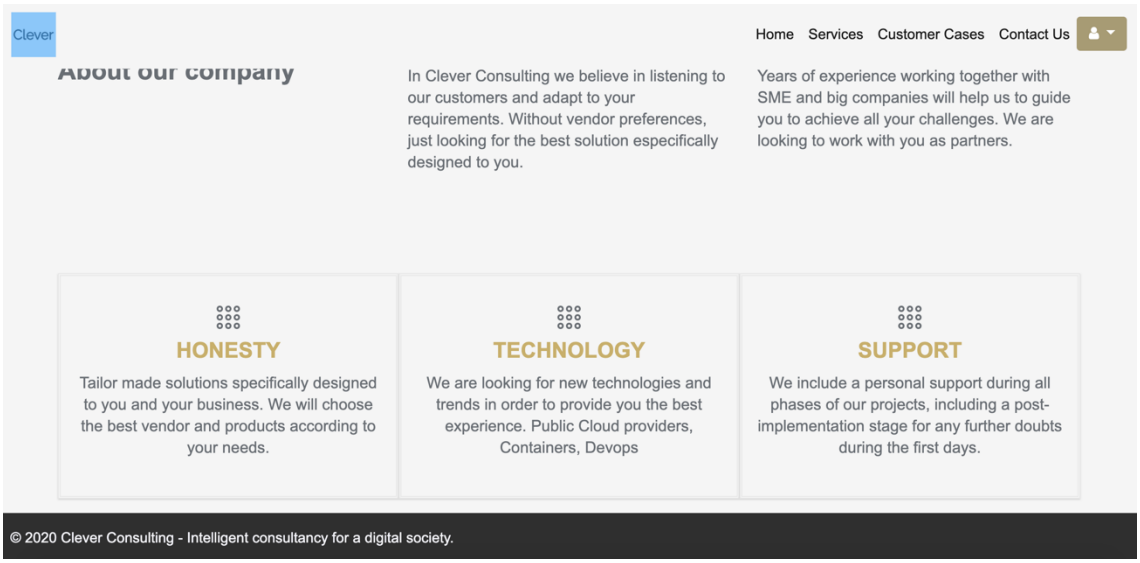


Ilustración 5: Ejemplo gráfico de la web (I)

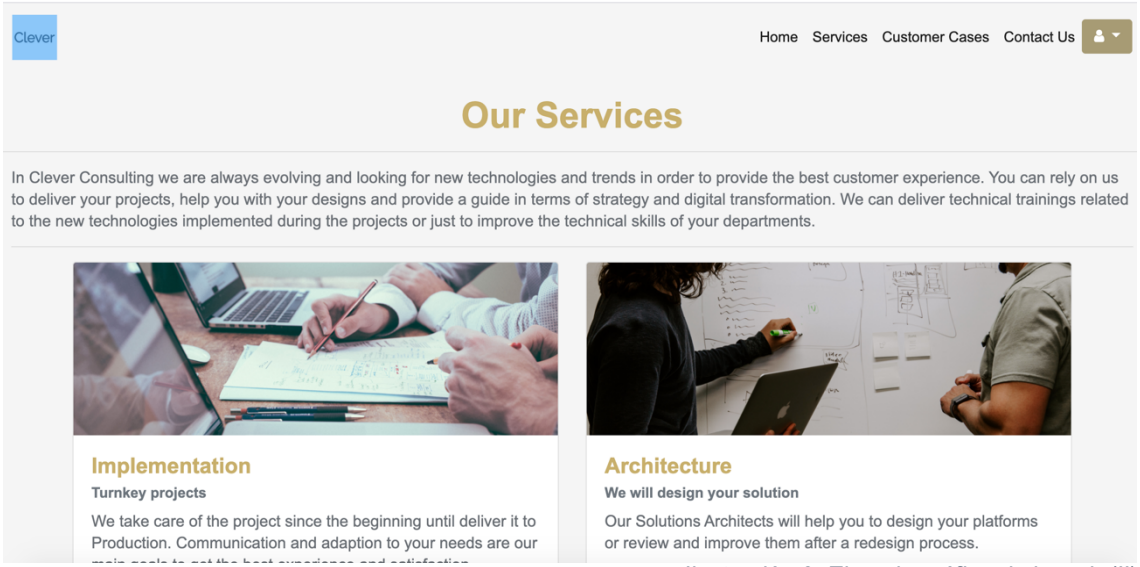


Ilustración 6: Ejemplo gráfico de la web (II)

5.3 Desarrollo de la intranet

Una vez realizada la web externa se comienza con la intranet. Para ello como punto inicial se trata la gestión de datos a través de la base de datos.

Para ello declaramos los modelos a ser utilizados con SQLAlchemy, que quedan definidos en el fichero models.py. Esto nos permitirá que nuestra aplicación sea capaz de crear las tablas de forma automáticamente al inicializarla (previa definición de los datos de conexión a la base de datos).

Ejemplo de la tabla Users que recoge los usuarios, ya sean clientes o los administradores (definidos por el booleano is_admin):

```
class User(db.Model, UserMixin):



    __tablename__ = 'users'



    id = db.Column(db.Integer, primary_key=True)

    email = db.Column(db.String(64), nullable=False,
unique=True)

    first_name = db.Column(db.String(64), nullable=False)

    last_name = db.Column(db.String(64),nullable=False)

    city = db.Column(db.String(64), nullable=False)

    country = db.Column(db.String(64), nullable=False)

    company = db.Column(db.String(64), nullable=False)

    phone_number = db.Column(db.String(64))

    password_hash= db.Column(db.String(128))

    vat_tax_id = db.Column(db.String(64))

    confirmed = db.Column(db.Boolean, default=False)

    is_admin = db.Column(db.Boolean, default=False)
```

Si bien para producción utilizamos una conexión de base de datos con MySQL para desarrollo se establece la posibilidad de establecer un entorno de preproducción a través de la variable de entorno FLASK_CONFIG y que hará que se apunte a una base de datos local sqlite (dev-database.sqlite).

Es decir, únicamente tendremos que darle el valor “production” a FLASK_CONFIG para que la aplicación ataque a la base de datos de MySQL (producción) o darle el valor “development” para que ataque a la base de datos local (preproducción). Esto podría ser modificable en un futuro para que atacará a otro servidor de base de datos MySQL en vez de un sqlite por ejemplo.

Una vez con la base de datos establecida, los usuarios son capaces de registrarse a través del formulario de registro:

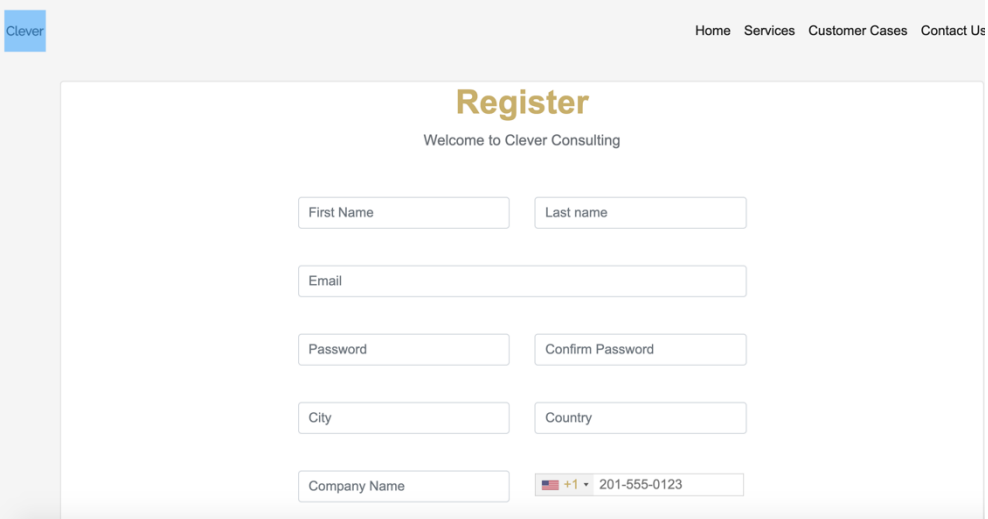


Ilustración 7: Ejemplo del registro

Una vez el usuario cliente está registrado y correctamente autenticado se le permite realizar reservas de los cursos online. De no estarlo, únicamente puede ver el catálogo pero no acceder a la web de reservas.

Catálogo visible para los visitantes no autenticados, en caso de pulsar en Get Started se les dirige para autenticarse o registrarse en la web.

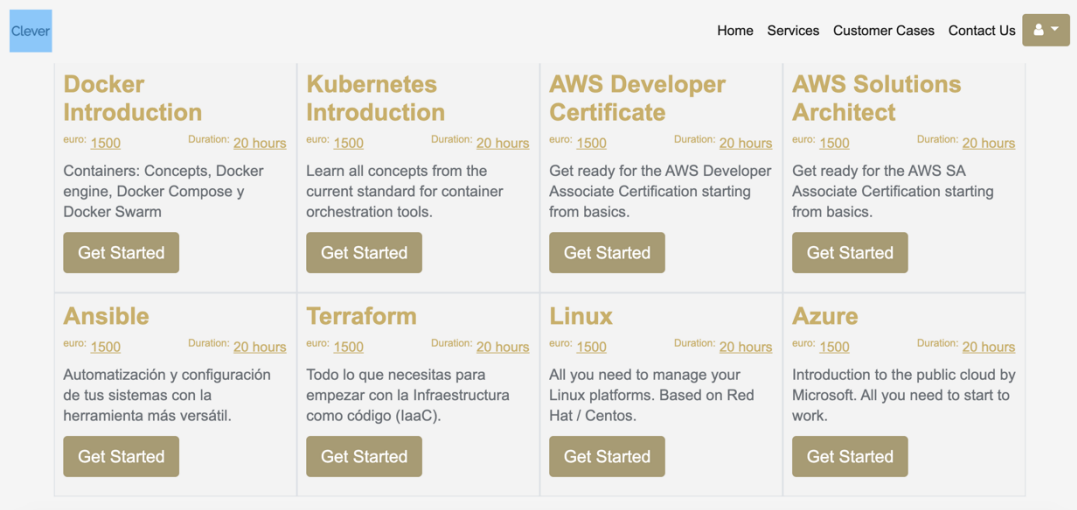


Ilustración 8: Catálogo de cursos sin reserva

Usuario autenticado y por tanto con capacidad de realizar reservas:

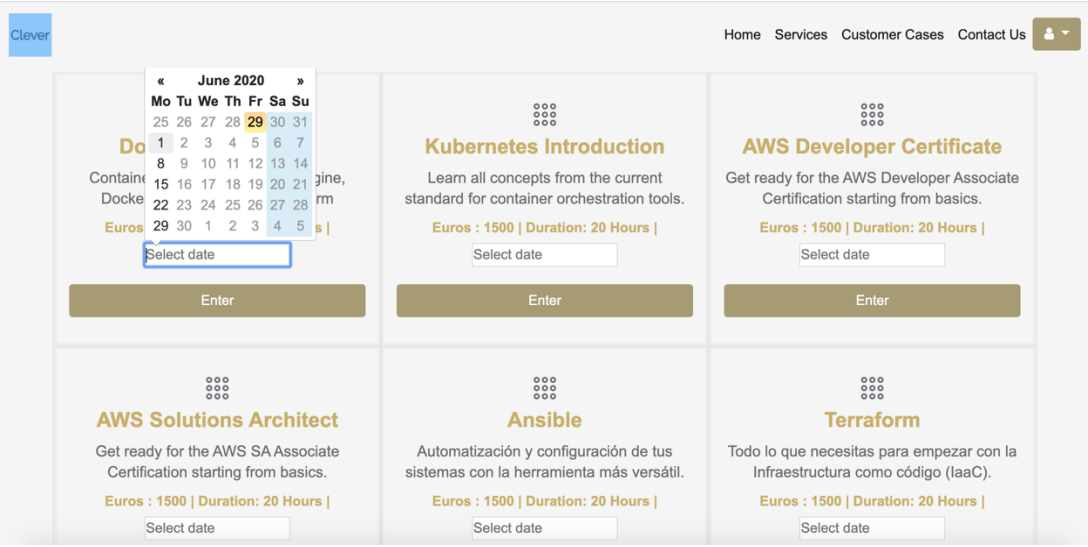


Ilustración 9: Catálogo de cursos con reserva

Las reservas (definidas en core) son posibles en caso de que haya profesores disponibles que estén asociados a ese curso (relación de 0 a n). Para ello realizamos las consultas necesarias en la base de datos y tratamos la excepción de que no haya ningún profesor con un try.

Para la gestión interna se ha habilitado un panel de administración (dentro de admin) que sólo es accesible para los usuarios definidos como administradores. Desde aquí se podrán realizar diferentes tareas como listar, crear, editar o eliminar cursos, profesores, usuarios, reservas, etc.

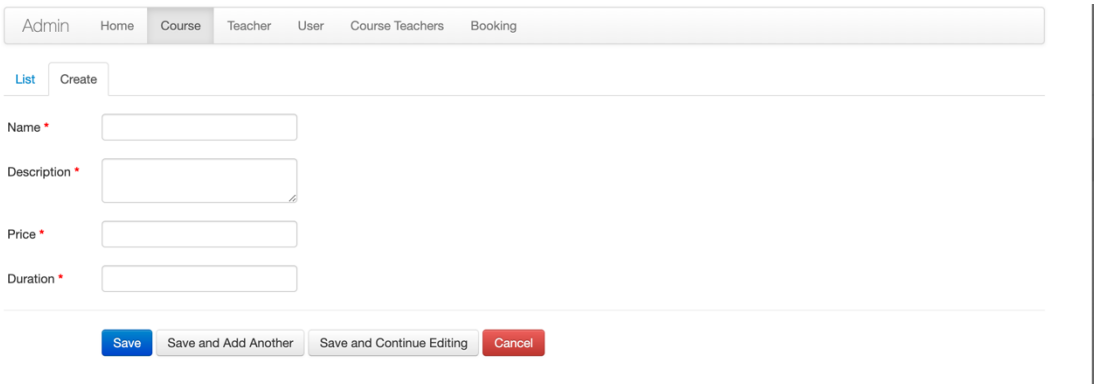


Ilustración 10: Ejemplo del panel de admin (I)

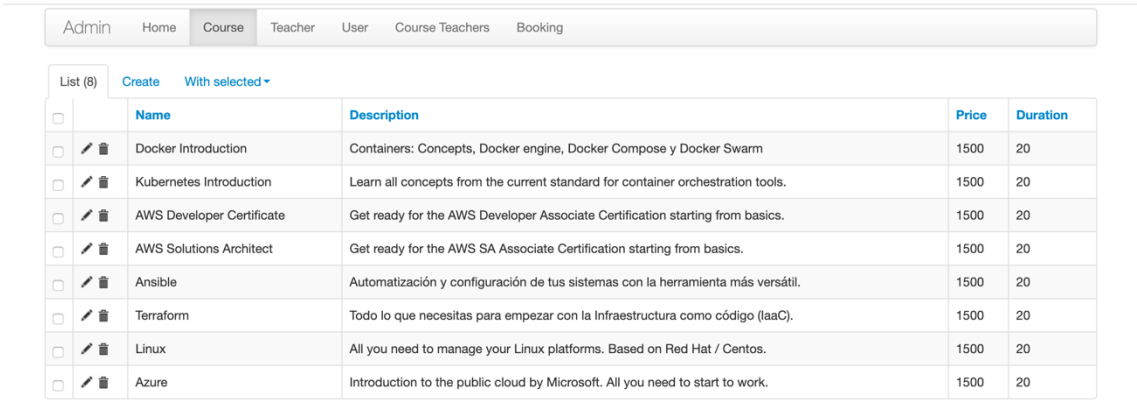


Ilustración 11: Ejemplo del panel de admin (II)

Si un usuario intenta acceder a la URL directamente (aunque no está enlazado desde la web externa accesible al público) se le denegará el acceso:

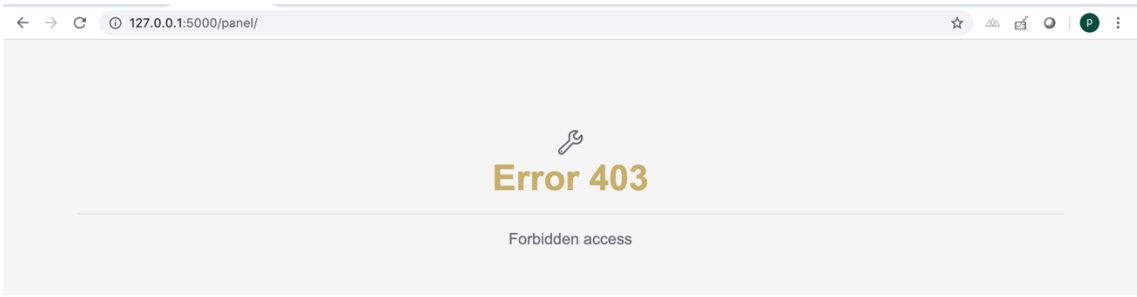


Ilustración 12: Ejemplo de control de acceso admin

Esta restricción se realiza a través de la clase AdminLockView:

```
class AdminLockView(ModelView):
    def is_accessible(self):
        if current_user.is_admin and not current_user.is_anonymous:
            return True
        else:
            return abort(403)
```

Los usuarios que son administradores están definidos en la base de datos, pero para no depender del contenido de la base de datos en una fase inicial se declara una variable APP_ADMIN que recoge el valor de una variable de entorno:

```
APP_ADMIN = os.environ.get("APP_ADMIN")
```

De esta manera, podemos definir el email de nuestro administrador y logarnos en el panel de administración sin haber de modificar la base de datos de forma interna, lo que supone una mejora de experiencia para el usuario.

5.4 Consideraciones finales

Para evitar la modificación del código web la configuración de algunas variables necesarias, como puede ser la configuración del correo o la base de datos, se realiza a través de variables de entorno que son recogidas desde la clase Config en config.py

De esta forma conseguimos que la aplicación pueda ser desplegada sin necesidad de modificar ningún fichero de su código, lo que facilitará su posterior mantenimiento al evitar posibles cambios por error

Como punto final se ha subido el contenido del código a un repositorio de GitHub bajo la licencia GPL v3 para que sea accesible y revisable:

https://github.com/pablosponce/tfg-web-uoc

6. Resultado y validación

6.1 Validación por parte del cliente de la web externa

Para la validación del cliente se realiza un ejercicio de *customer journey*

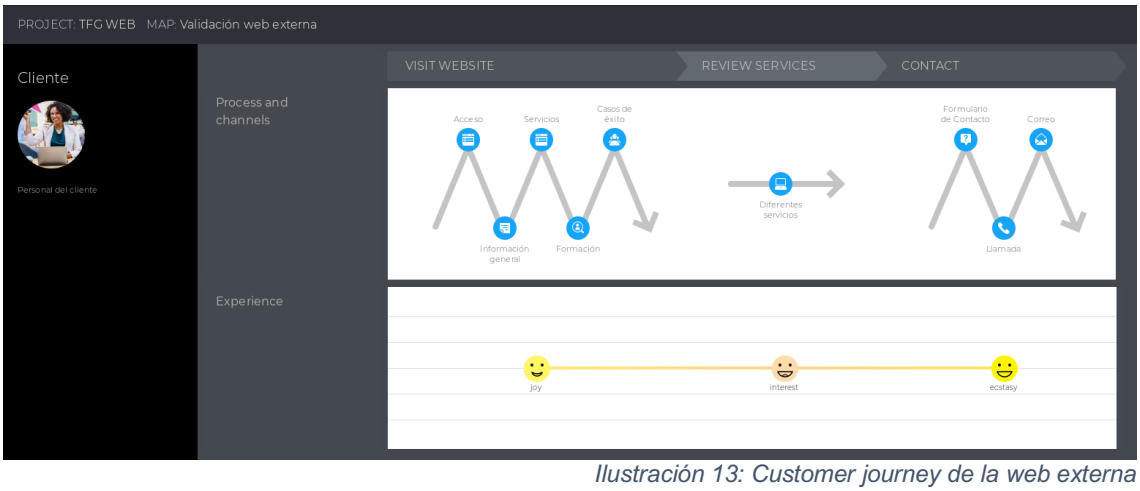


Ilustración 13: Customer journey de la web externa

El usuario (en estas pruebas el cliente) accede a la web principal donde verá la información general de la empresa y se hará la primera impresión de la empresa y los posibles servicios que ofrece.

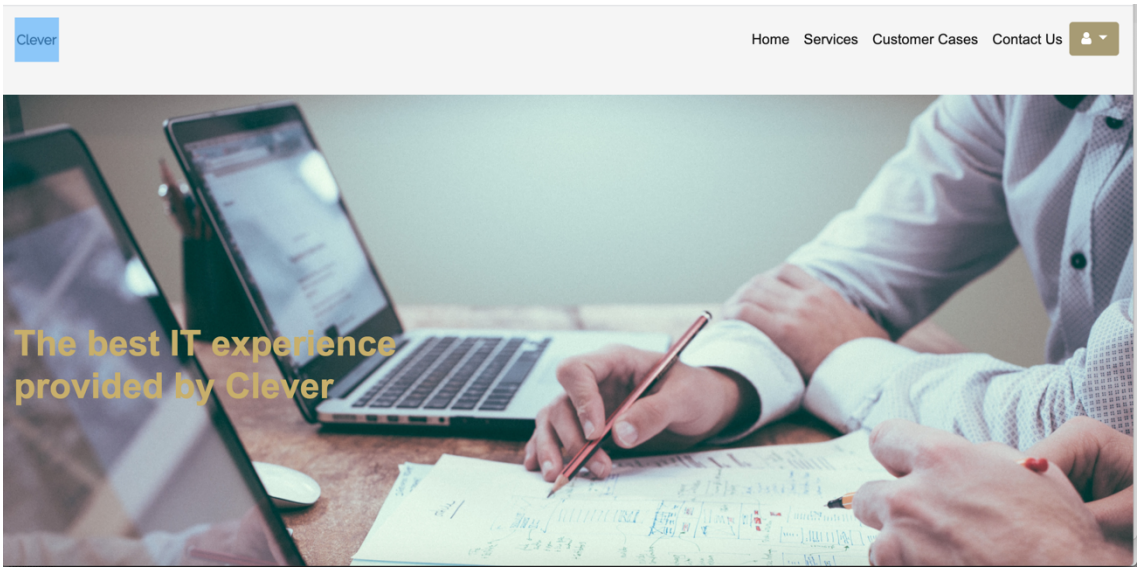


Ilustración 14: Captura de la web principal (I)

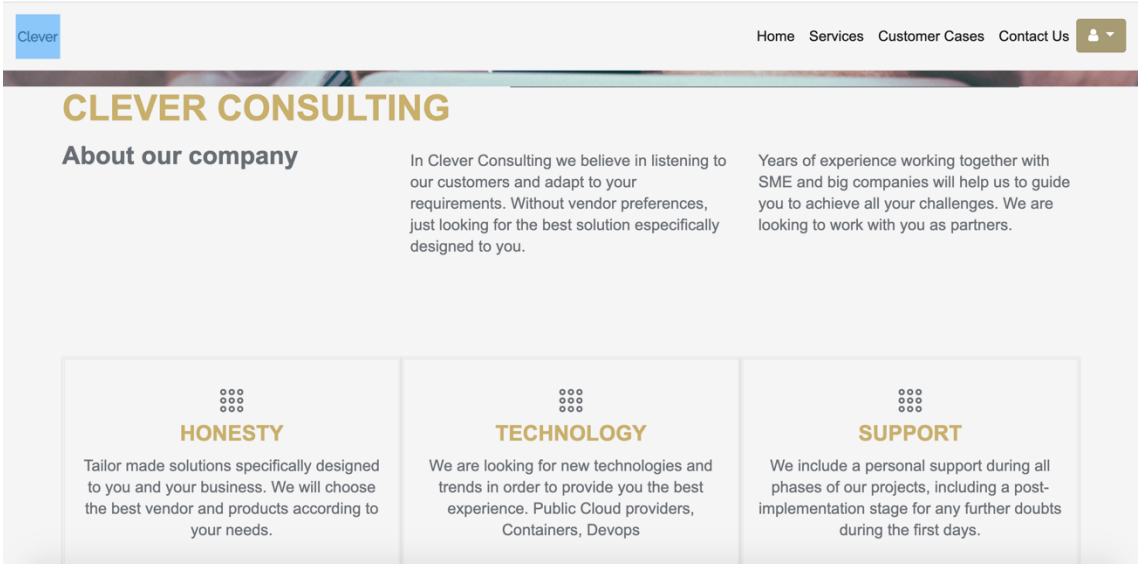


Ilustración 15: Captura de la web principal (II)

A continuación revisará la lista de servicios que ofrece la empresa, la oferta de formación y los casos de éxito publicados en la web.

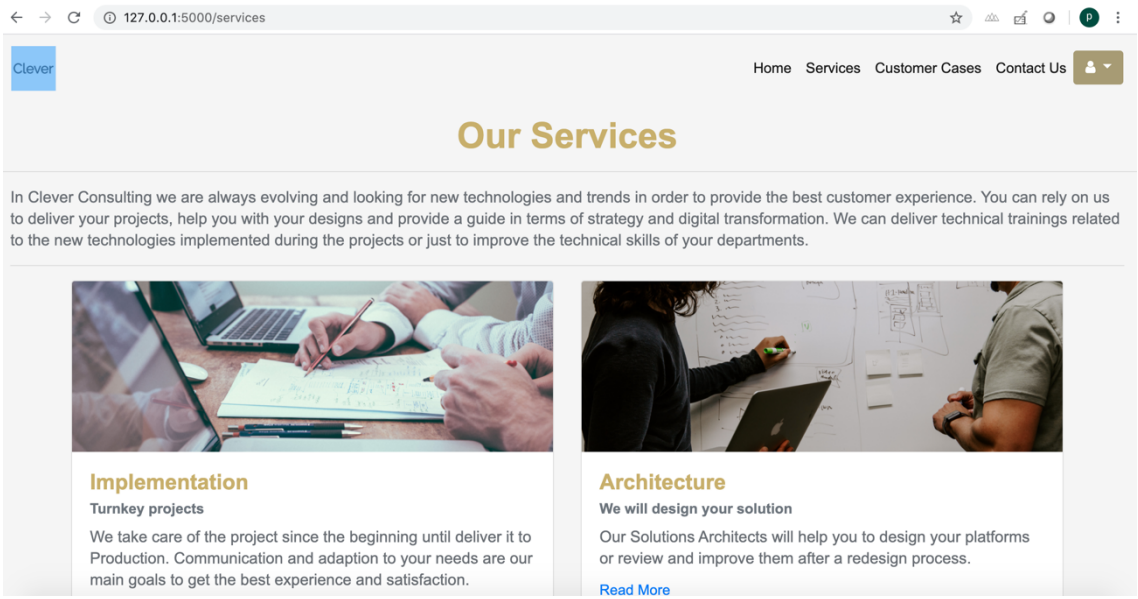


Ilustración 16: Captura de la lista parcial de servicios ofrecidos

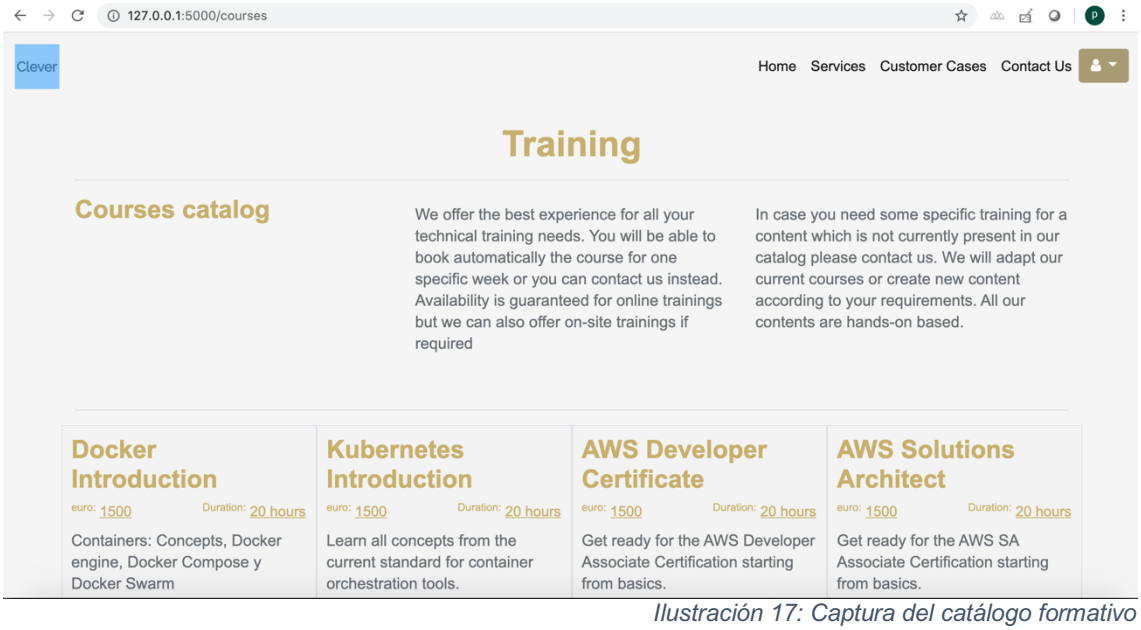


Ilustración 17: Captura del catálogo formativo

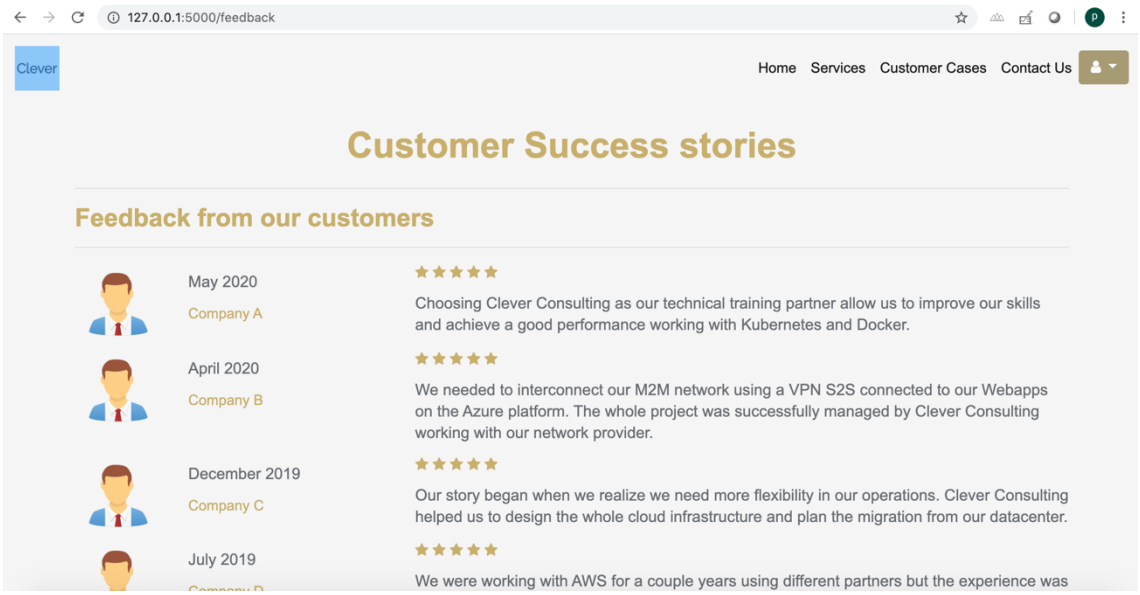


Ilustración 18: Captura de los casos de éxito

Una vez realizada la primera inspección de la web, volverá a la sección de servicios y revisará de forma detallada el servicio o servicios en los que esté interesado.

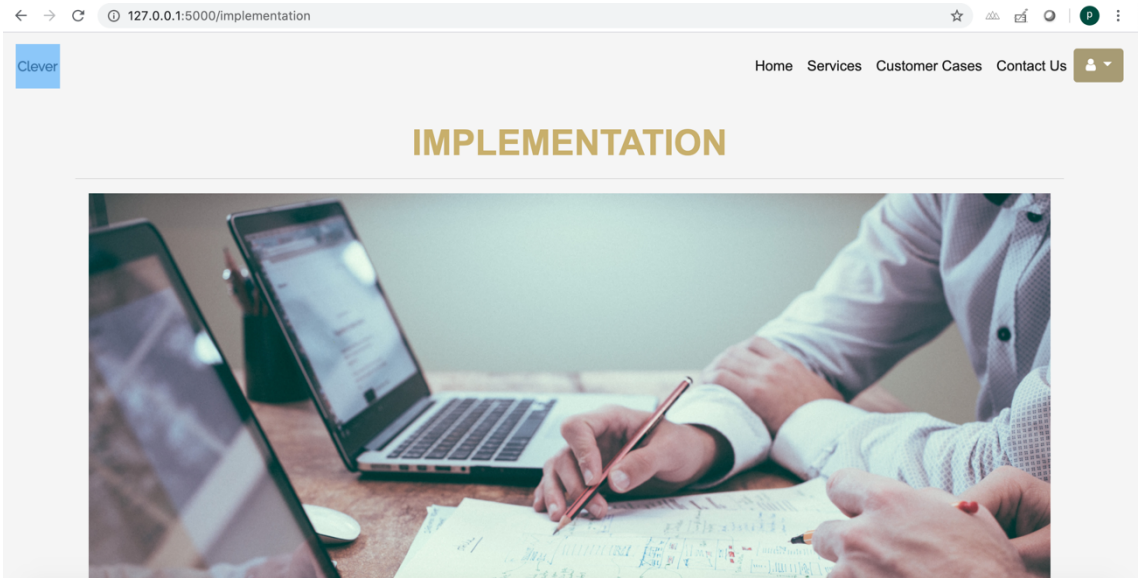


Ilustración 19: Captura de la sección del servicio Implementation

Por último, procederá a contactar con la empresa para solicitar más información, pedir un presupuesto o realizar alguna propuesta de colaboración de forma que termine el flujo de la navegación web.

En este caso, podrá optar por tres diferentes vías:

- Escribir a través del formulario de contacto
- Enviar un correo electrónico a la dirección indicada
- Llamar por teléfono.

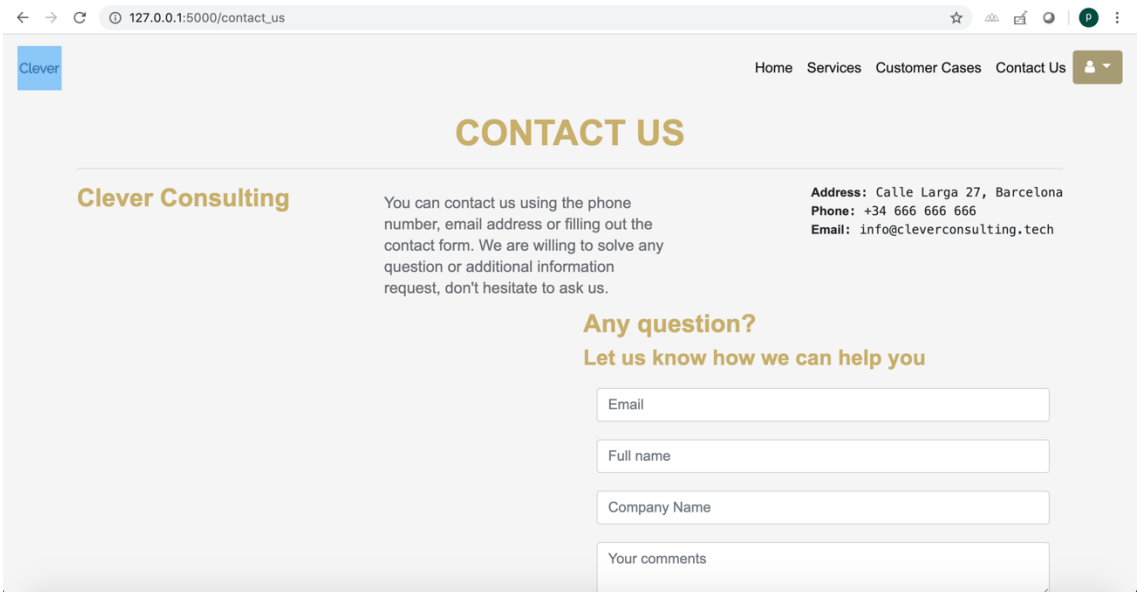


Ilustración 20: Captura de la sección de contacto

6.2 Validación por parte del cliente de intranet

Al igual que en la web externa, la validación se realizará a través de un *customer journey* para comprobar la experiencia del usuario y que se cumplan las funcionalidades requeridas.

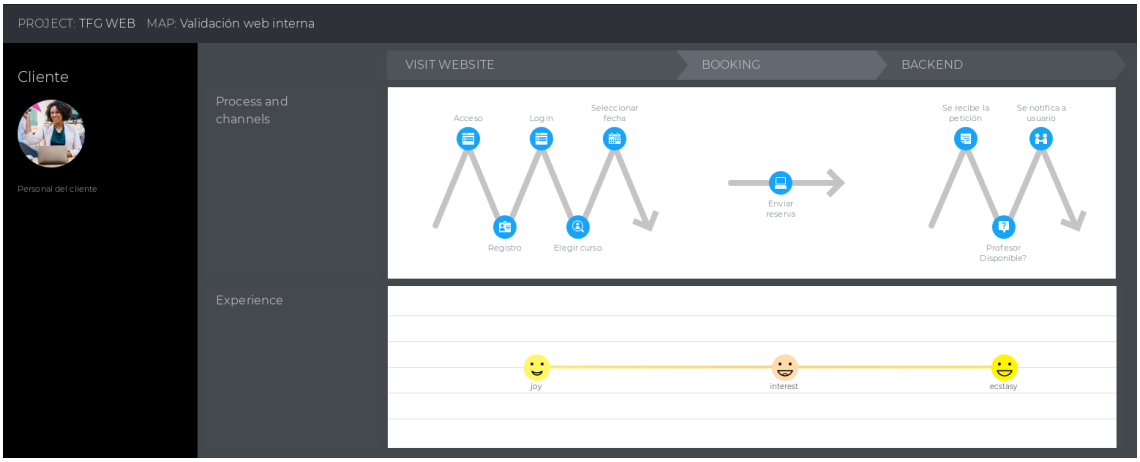


Ilustración 21: Customer journey de la web interna

El usuario accederá a la web inicial donde verá la lista del catálogo de cursos

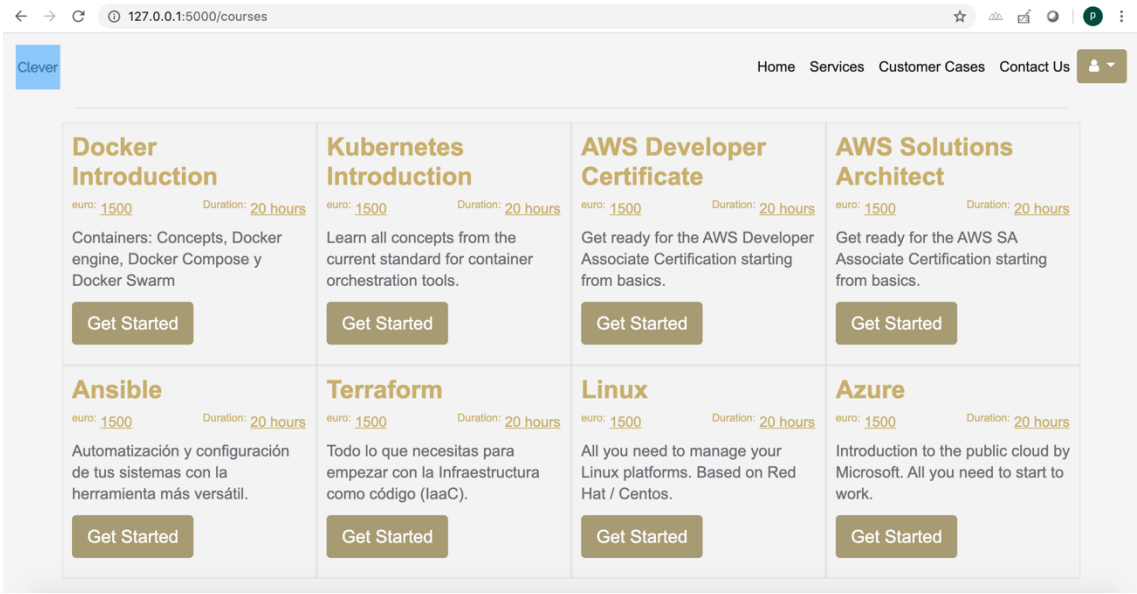


Ilustración 22: Captura del listado de cursos de un usuario no autenticado

Si acciona alguno de los botones de *Get Started* de uno de los cursos se le indica al usuario que ha de autenticarse

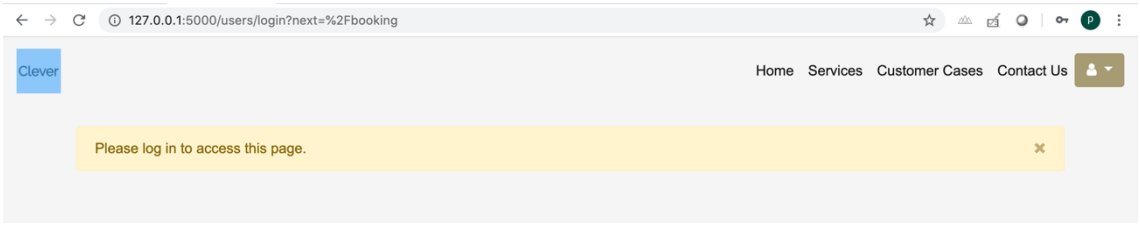


Ilustración 23: Captura del mensaje que indica la necesidad de autenticarse para las reservas

Pulsando en el enlace de registrarse, podrá rellenar el formulario con los datos necesarios para su identificación y contacto

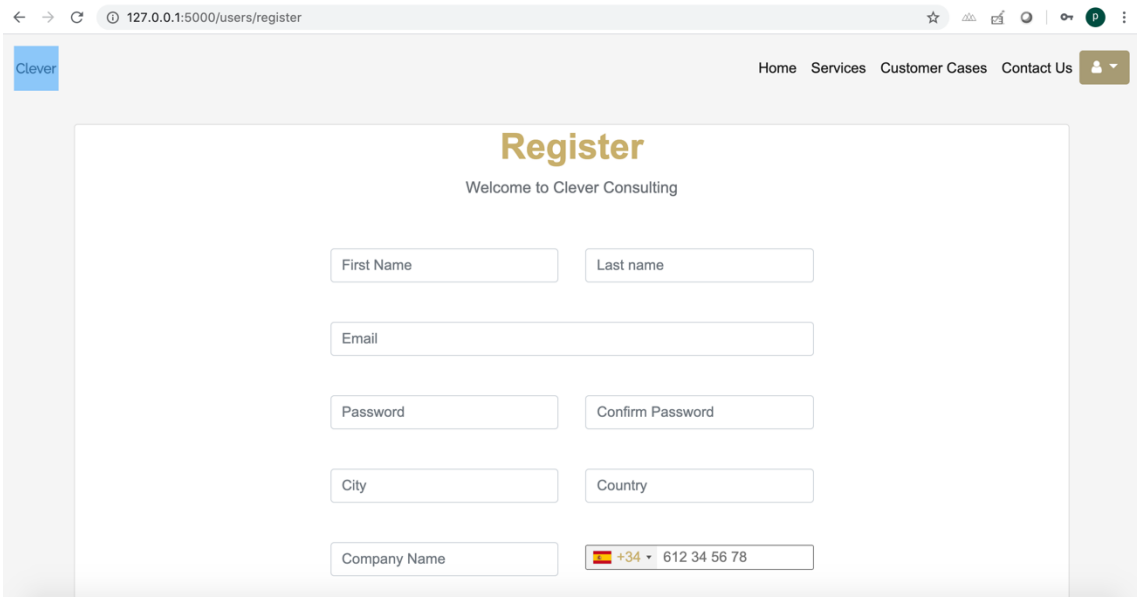


Ilustración 24: Captura del formulario de registro

Una vez registrado, el usuario recibirá un correo para validar su dirección y podrá autenticarse en la web a través del formulario de inicio de sesión:

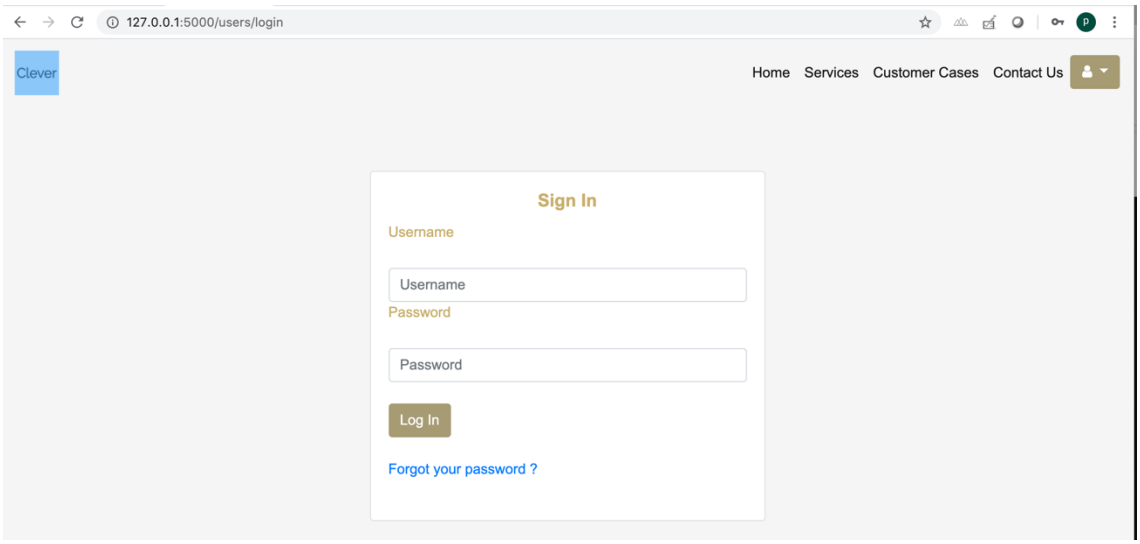


Ilustración 25: Captura del formulario de inicio de sesión

Tras la autenticación el usuario podrá seleccionar la fecha en la que desea la reserva de los diferentes cursos ofrecidos

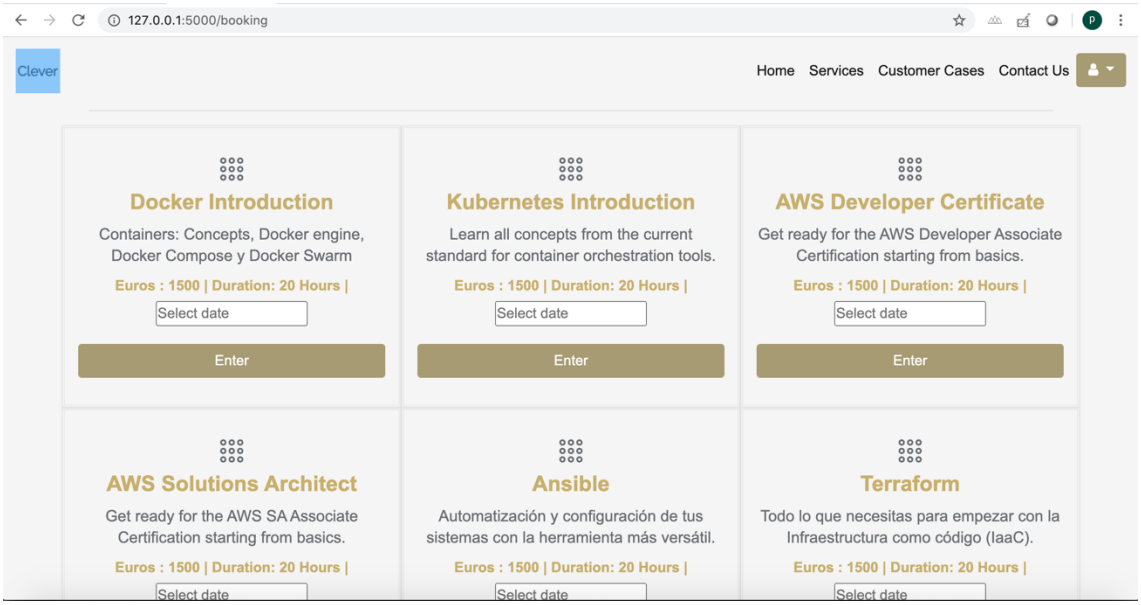


Ilustración 26: Captura del catálogo de cursos para un usuario autenticado

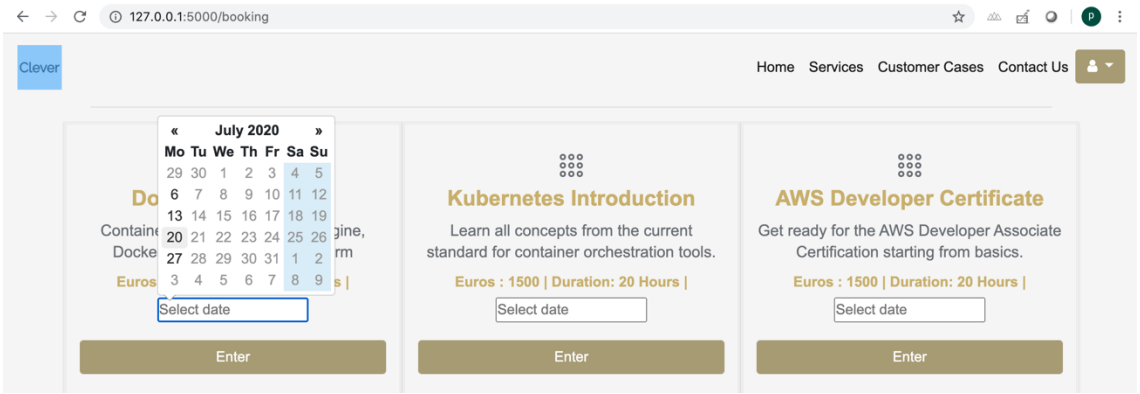


Ilustración 27: Captura de selección de fecha de un curso

La aplicación web comprueba la disponibilidad para la fecha indicada de los profesores que están asociados a ese curso específico y genera la reserva de ese curso para ese cliente asignando a uno de los profesores disponibles. Una vez realizada la reserva, la aplicación devuelve al usuario un mensaje de confirmación, además de proceder a enviar un correo electrónico con los detalles de la reserva.

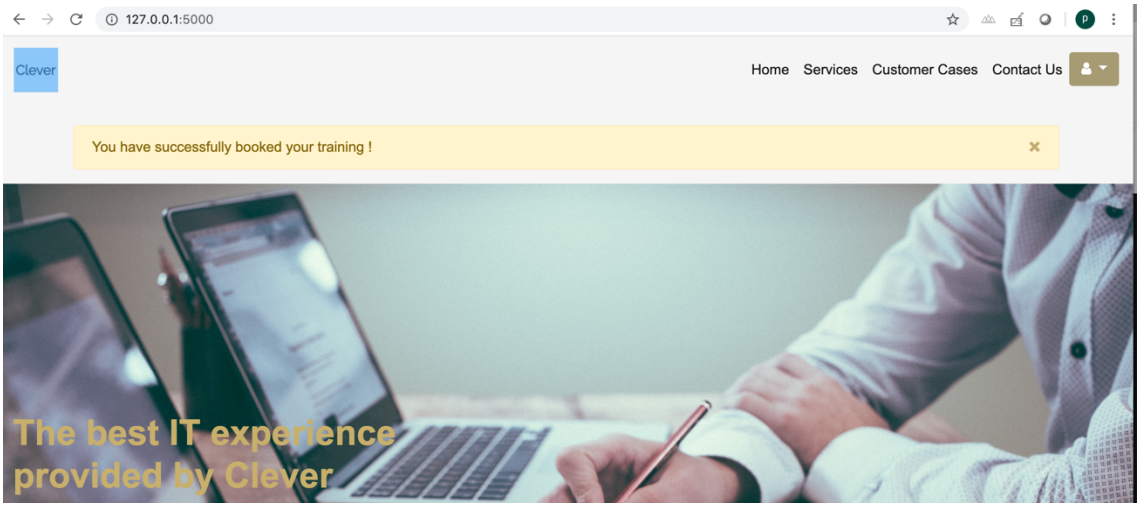


Ilustración 28: Captura del mensaje de confirmación de una reserva

7. Pruebas

7.1 Pruebas de la web externa

De acuerdo a las fases establecidas en el inicio del proyecto, una vez realizado el desarrollo de la web externa se ha procedido con una batería de pruebas para comprobar el correcto funcionamiento de la misma.

ID Test	Descripción Test	Comentario	Resultado
EXT1	Formulario de contacto	Se recibe el correo correctamente. Ver imágenes adjuntas	OK
EXT2	Web adaptativa	Se visualiza la web con dispositivos móviles IOS y Android adaptándose a las resoluciones.	OK
EXT3	Catálogo de cursos	Se visualizan adecuadamente y de forma proporcionada	OK
EXT4	Errores personalizados	Se muestran páginas personalizadas de errores, ej: 404	OK

Prueba del formulario de contacto (EXT1):

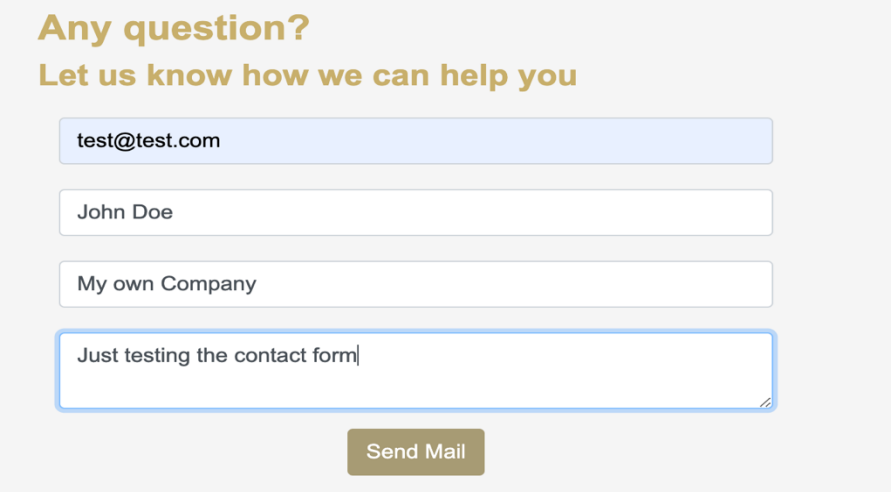


Ilustración 29: Prueba del formulario de contacto (I)

Recepción del correo:

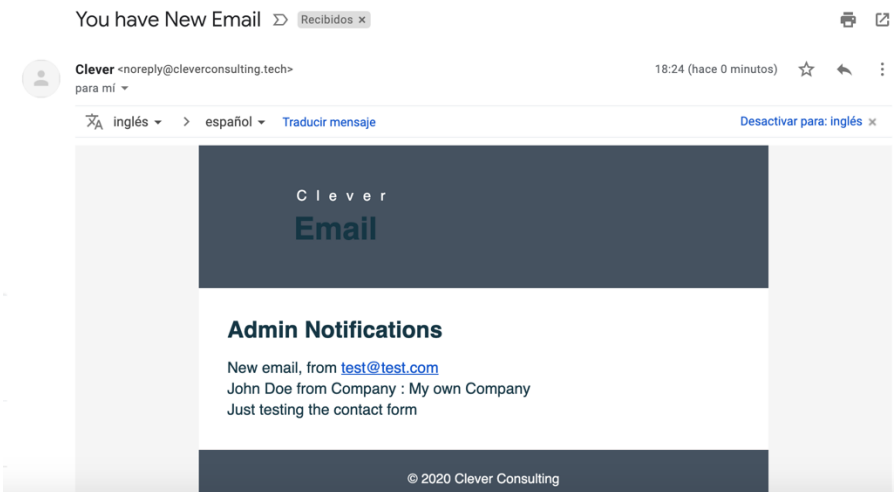


Ilustración 30: Prueba del formulario de contacto (II)

Prueba de errores personalizados (EXT4):

Ejecutamos una llamada a una página que sabemos que no se encuentra en nuestra aplicación web, en este caso “noexiste”:

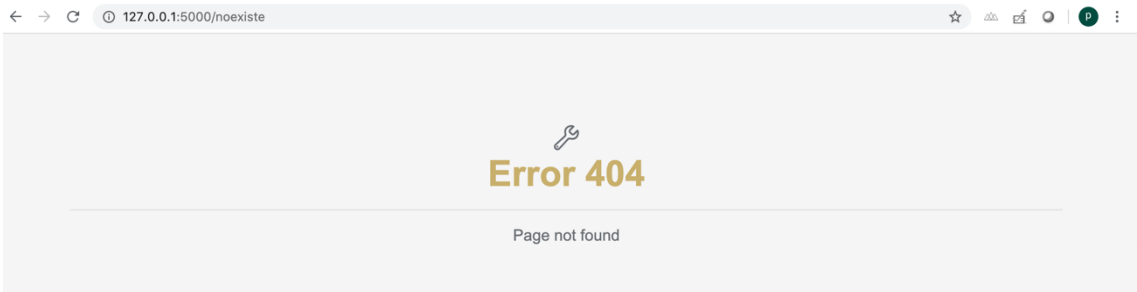


Ilustración 31: Prueba de errores personalizados

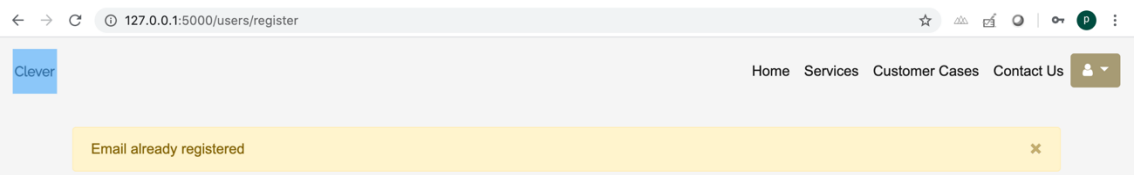
7.2 Realización de pruebas de la intranet

Una vez realizado el desarrollo de la web interna se procede con la siguiente batería de pruebas, todas ellas satisfactorias:

ID Test	Descripción Test	Comentario	Resultado
INT1	Registro usuario	El usuario se registra y es capaz de autenticarse	OK
INT2	Usuario duplicado	Un usuario no puede registrarse si existe otro usuario con el mismo email.	OK
INT3	Recuperación contraseña	Se recibe el correo y se puede restaurar la contraseña.	OK
INT4	Reserva curso	El curso se reserva de forma correcta	OK
INT5	Curso no disponible	Un curso sin profesores libres en esa fecha muestra que no está disponible para su reserva.	OK
INT6	Notificaciones reserva	Se reciben email con los detalles de la reserva, tanto en el cliente como el admin y el profesor asignado.	OK
INT7	Profesores duplicados	No se pueden crear dos profesores iguales.	OK
INT8	Asignaciones duplicadas	Un mismo profesor no puede estar asignado varias veces en el mismo curso	OK

Prueba de usuario duplicado (INT2)

Intentamos registrarnos a través del formulario con un usuario ya existente, en este caso el email pabloponce@uoc.edu y recibimos un mensaje indicando que ese email ya está en uso dentro de la aplicación web:



Prueba de profesores duplicados (INT7)

Lista de los profesores ya añadidos al sistema

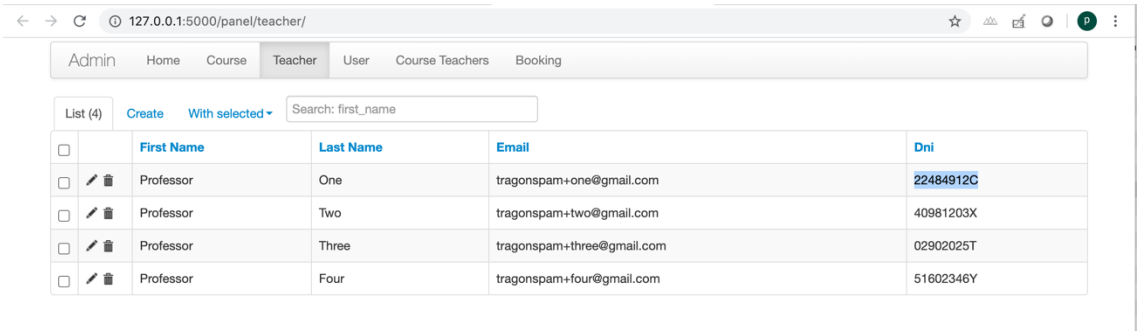


Ilustración 32: Prueba de profesor duplicado (I)

Si intentamos añadir un profesor con las mismas características, nos dará un error indicando que tanto el email como el DNI ya están en nuestra aplicación:

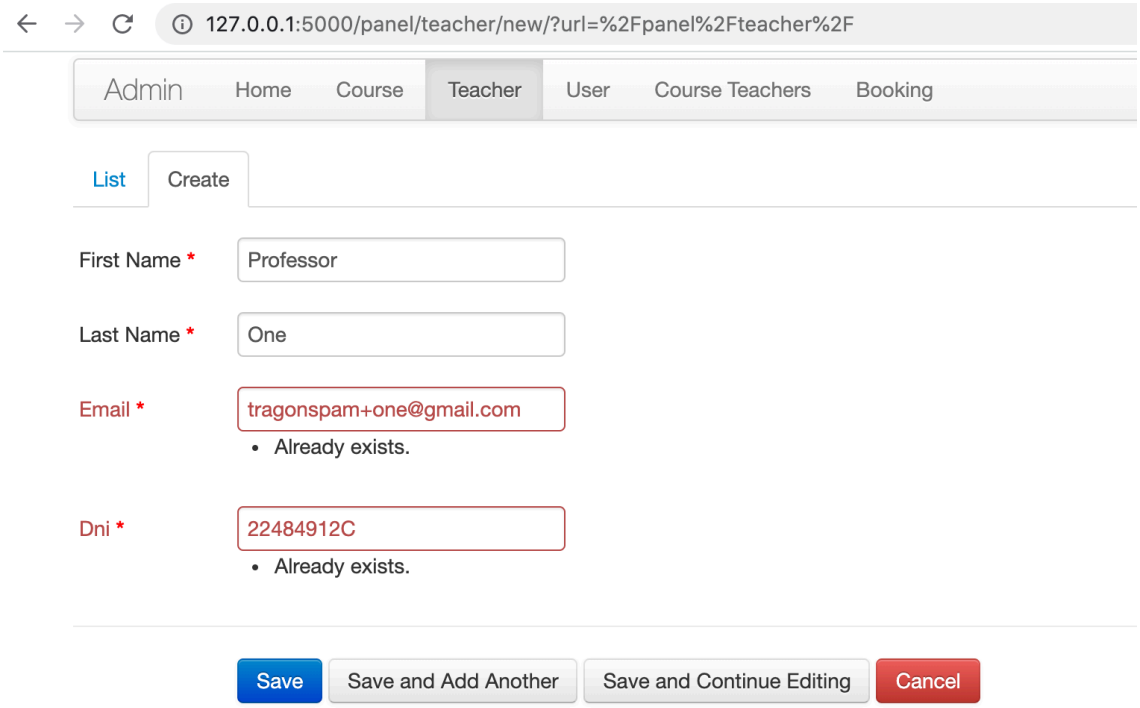


Ilustración 33: Prueba de profesor duplicado (II)

8. Conclusiones

La realización del trabajo ha sido muy interesante, ya que me ha permitido aprender muchas nociones nuevas sobre el diseño web en general y Python Flask en particular, debido a la falta de experiencia previa en estas tecnologías antes de comenzar el TFG.

Los objetivos iniciales han quedado cubiertos si nos atenemos a los requisitos y al enfoque con el que estaba descrita la página web en su inicio. No obstante, siempre hay puntos a mejorar, como por ejemplo en el diseño gráfico de la web, la presentación de algunos datos o un área de cliente más detallada y más similar a un portal de usuarios.

La planificación se ha seguido de forma bastante similar a la prevista y no ha habido ninguna incidencia o retraso a destacar, aunque tampoco ha sobrado tiempo respecto a las previsiones iniciales. Un mejor diseño previo habría evitado algunas correcciones posteriores, como por ejemplo en el modelo ER de la base de datos, ya que no todos los campos necesarios fueron previstos en el inicio y algunos se han debido añadir a posteriori.

Así mismo, la aplicación web tiene futuro para implementar nuevas mejoras y funcionalidades, especialmente dentro de la interacción de los clientes. No obstante, habría que plantear si el tipo de clientes es el adecuado o no para ellas, ya que la mayoría tiene un mejor encaje en clientes con una relación permanente y con diferentes interacciones a lo largo del tiempo. Algunas posibilidades serían las siguientes:

- Contratación y pago online de los cursos.
- Área personal con el listado de los servicios contratados
- Área personal de facturación, con el histórico de las facturas enviadas descargables.
- Servicio de chat o soporte técnico integrado

9. Glosario

- Backend. Capa de la aplicación que se ejecuta internamente para dar servicio a la capa de interacción con el usuario (*frontend*).

- Customer journey. Proceso que define la experiencia del usuario al realizar una tarea, donde se revisa cada paso necesario para la ejecución de la misma.

- Flask. Framework para el desarrollo web con Python.

- Framekork. Plataforma para el desarrollo de *software* que incluye diferentes componentes para la programación.

- IDE. Aplicación que provee un entorno completo de programación, su nombre viene de las siglas en inglés de entorno integrado de desarrollo (Integrated Development Environment)

- Jinja2. Lenguaje de plantillas moderno que facilita la reutilización de código en Python.

- Wekzeug. Librería de WSGI (*Web Server Gateway Interface*) para aplicaciones web en Python.

10. Bibliografía

[1] Recurso online: https://www.python.org/doc/ Visitado marzo 2020.

[2] Recurso online: https://flask.palletsprojects.com/en/1.1.x/ Visitado marzo 2020

[3] Miguel Grinberg, “Flask Web Development: Developing Web Applications with Python”, Editorial O’Reilly, 2014

[4] Recurso online: https://www.jetbrains.com/pycharm/documentation/ Visitado abril 2020

[5] Recurso online: https://dev.mysql.com/doc/refman/5.7/en/ Visitado abril 2020

[6] Recurso online: https://app.diagrams.net Visitado marzo 2020

[7] Recurso online: https://uxpressia.com/ Visitado mayo 2020