**INSTITUTO SUPERIOR POLITÉCNICO CÓRDOBA**

**TECNICATURA SUPERIOR**

**EN DESARROLLO WEB Y APLICACIONES DIGITALES**

*Proyecto Final EN CALIDAD DE* ***LIBRE***

**MÓDULO PROGRAMADOR WEB**

Este documento está dirigido a los/as estudiantes que deben rendir en mesa examinadora final **MÓDULO PROGRAMADOR WEB** en ***condición de LIBRE.***

#### **Objetivos**

Elaborar una aplicación web con el fin de lograr que los estudiantes demuestren consolidar, integrar y /o ampliar las capacidades y saberes que corresponden al Programador Web.

#### **Criterios de Evaluación**

El estudiante deberá presentarse con la aplicación web desarrollada de acuerdo a los 3 proyectos designados por los Docentes, la cual estará funcionando en un servidor local y el entorno de programación, el día de examen para hacer la defensa del mismo. Ese día, en la sala de las mesas examinadoras luego de su acreditación, se harán algunas preguntas técnicas sobre lo solicitado en el proyecto.

#### **Consignas**

Para los estudiantes en condición **LIBRE**, se les solicitará un proyecto **(\*\*\*)** publicado en Github aplicando todos los contenidos dados en los módulos de la cursada (HTML, CSS, BOOTSTRAP, ANGULAR Y Python y Django), pueden recurrir a los recursos web oficiales y material brindado en la cursada.

**(\*\*\*) Propuestas de Proyectos:**

* Veterinaria, con registro de mascotas y vacunación aplicada.
* Coworking (app de reservas de espacio para uso reuniones, oficina, etc).
* TaxiUber ( App para reservar para taxis/remis Córdoba).

Todos deben simular formas de pago para cerrar el proceso de e-commerce/carrito.

#### **Lineamientos Generales**

Desarrollar una aplicación Web con funcionalidades completas (frontend + backend integrado) Se pretende concluir con el desarrollo de la misma siguiendo el ciclo completo de desarrollo de software

La aplicación deberá implementar el contenido mínimo dictado en clases sincrónicas, asincrónicas y material de la plataforma pudiendo anexar lo que crea necesario justificándose; y todos los requerimientos necesarios para la organización de trabajo en equipo, simulando el trabajo en una empresa real.

## **CONSIGNAS PARA PRESENTAR EL EXAMEN**

El estudiante deberá presentarse el día del examen con la aplicación web desarrollada y el proyecto en su totalidad en el repositorio de trabajo, desarrollados de acuerdo a los ítems designados por los docentes, la cual estará funcionando en un servidor local y con el entorno de programación (IDE necesario para correr Front y Back), el día de examen para hacer la defensa del mismo.

Debiendo realizar la defensa del trabajo en el espacio disponible **MESA EXAMINADORA PROGRAMADOR WEB - Sala de Examen Sincrónico**, mediante videoconferencia que presenta la plataforma (MEET) en la correspondiente mesa examinadora.

La defensa del trabajo consiste en que el/la estudiante muestre el desarrollo de la aplicación Web en el entorno de trabajo para que se pueda apreciar: la estructura del proyecto, el código y los aspectos de diseño y recursos utilizados. Posteriormente se procederá a ejecutar la aplicación y se realizarán algunas preguntas técnicas sobre lo solicitado en el proyecto.

## **PUNTOS A REALIZAR**

#### **Frontend (HTML - CSS - Bootstrap - Angular):**

* Logra crear SPA responsive con componentes y servicios en Angular (HOME, REGISTRO, LOGIN, DASHBOARD, PRODUCTO o SERVICIOS, entre otros), incorporando el sistema de ruteo para la navegación.
* Logra implementar formularios (basados en template o reactivos) con su respectiva validación (maxlenght, minlenght, min, max, email, etc) y mensajes personalizados.
* Logra el enlace de datos (data binding) en componentes a fin de mostrar los datos de manera dinámica en las vistas.
* Logra implementar servicios para separar la lógica de conexión a ApiRest (backend) de los componentes. Debe implementar observables y manejo de excepciones.
* Logra implementar un tablero de control (dashboard) para usuarios cliente: Debe mostrar catálogo (mínimo 2 productos o servicios o suscripción) y permitir la comprar. Simular Pago con API Mercado Pago o API similar.
* Logra implementar tablero de control administrador (dashboard) que permita la gestión CRUD básico funcional de productos y clientes.

#### **BACK (Python - Django - MySQL)**

* Logra adjuntar el script sql (MySQL) para la creación de la base de datos con los productos/servicios/suscripción iniciales en stock.
* Logra crear un usuario del tipo administrador para poder usar Django Admin.
* Logra usando Django crear los modelos necesarios para la aplicación web.
* Logra en el Tablero de control (Gestión) o Dashboard de Django Admin permitir visualizar y hacer un CRUD de las tablas de productos/servicios/suscripción de su modelo de datos.
* Logra usando Django Rest Framework crear los serializados,rutas y vistas necesarias para ser consumidas por el frontend.
* Logra permitir la autenticación de usuario mediante la vinculación del front con el back usando Django Rest Framework.
* Logra implementar la lógica necesaria en comunicación con frontend para recuperar los productos/servicios/suscripciones desde el backend y crear un carrito de compras.
* Logra simular Pago con API Mercado Pago o similar.

#### **DESARROLLO DE SOFTWARE - Se valora el MAIN**

* Logra conectarse al repositorio GitHub (en este caso será el único contributor).
* Definir un sistema de **Branching** con un mínimo de dos ramas. Ejemplo: Main, Develop.
* Logra agregar **Historia de Usuarios /tareas**, (mínimo 10 issues) asignarlos a colaboradores (a Ud) y **por Sprint**, (sprints con fechas de inicio y final, se sugiere sprint 1 y 2) respetando la nomenclatura de las issues sugeridas.
* Logra realizar **Commit** en todo el recorrido de Sprints, en las ramas feature, dev. (No se considerarán los que digan “add file upload”, descontando puntos)
* Logra realizar el release.
* Logra crear **Wiki** con registro de ceremonias (daily, review, retrospective) y novedades del equipo.
* Utiliza **kanban** (Product Backlog, ToDo, In process, Finished).
* Logra Crear/Actualizar **documentación IEEE830**
* Logra mergear /pull al main actualizando todo el proyecto.
* Logra diseñar el **modelo de datos conceptual** (DER): Entidad, Atributos y cardinalidad.(imagen/pdf)
* Logra diseñar el modelo de datos lógico **(Modelo Relacional)** a partir del **DER**: (debe estar mínimo en 3ra forma normal) (imagen/pdf)
* Logra diseñar el diagrama de clases UML: Relaciones.(imagen/pdf)

Se puede crear **Un video corto** enviado con antelación donde el estudiante evidencie y explique con lenguaje técnico la aplicación funcionando y el código de manera alternada de forma tal que se pueda visibilizar el proceso que él mismo hizo codificando para obtener el resultado (El objetivo del Video es poder tener más en claro el proceso del desarrollo y lo aplicado en el proyecto, de lo contrario tendría que librarlo a incluirlo en el dia del examen)

#### **CONSIDERACIONES FINALES**

* El proyecto puede ser presentado con anterioridad para revisión, enviando el link (enlace) del repositorio GitHub: **LOS ENVIOS SE RECIBIRÁN HASTA 96 hs antes de la MESA EXAMINADORA.**
* La modalidad de trabajo en esta instancia es individual, aunque se permitirá la modalidad ***Pair Programming*** para facilitar la organización y gestión del proyecto dentro del repositorio.
* Cada desarrollador debe conocer por completo el código de cómo fue creada la aplicación.

#### **CONSIDERACIONES PARA DIA DE LA MESA EVALUADORA**

* En el momento del examen, **cada uno** debe tener todos los recursos necesarios para poder realizar la demostración integral del proyecto: Cámara, micrófono, Visual Studio Code (IDE correctos) y software necesario para así poder compartir pantalla y realizar la demo.
* TIEMPO DE TOLERANCIA 15 min. Se recomienda tener instalado Google Chrome, Mozilla Firefox, o cualquier navegador estable, que funcione correctamente al momento del examen.
* **No se resolverán problemas técnicos en el momento del examen.**

##### **Preguntas orientadoras en el momento del examen, que el estudiante mencione:**

* Hacer un circuito de compra, que descuente stock, y pasarela de pago.
* Mostrar en backend (Django) donde hace la conexión a la DB y mostrar la API (MVC) - json NOTA que pare el spring. postman
* Mostrar en frontend (Angular) donde se consume la API de pago y API Rest de Django.
* Todos deben brindar formas de pago para cerrar el proceso de e-commerce/carrito.

\*\*\*Nota:Sintetizar el recorrido por la gestión del proyecto en GitHub.

**OPCIONALES**: Sitio publicado en Servidor Remoto. Token.