1. (5 pts) ¿Qué es una arquitectura de software y cuál es su relación con los sistemas de información? Explique en detalle.

Para mí, la arquitectura se software habla acerca de las tendencias tecnológicas, que influyen en el mercado para las empresas y por medio su relación con los sistema de información, se comienza a construir una base que se permita, interactuar con varios grupos de trabajo de servicios, como las bases de datos, administración de servidores, el flujo de la información, desarrollo de las páginas web, la interfaz gráfica de los aplicativos, programas corporativos, entre miles de cosas más, todos estos componentes, áreas, programas, utilidades, se complementan entre sí, permitiendo el control, en las diferentes áreas que tienen las empresas, como pueden ser las ventas, las finanzas, el control de un inventario, rapidez y efectividad en las consultas, todo esto con el fin de poder tener mayor claridad a la hora de tomar decisiones, para dirigir una empresa, ya que los altos directivos, al momento de tomar decisiones oportunas, permite competir en el mercado, permite llevar un control de los ingresos y egresos, permite apuntar a nuevas metas, nuevos retos, se pueden proyectar en un futuro, les permite expandirse, todo con base a los resultados de sus aplicaciones, y a los sistemas de información.

1. (5 pts) Describa el proyecto que realizará en el curso. Apóyese en alguna figura.

EL proyecto se base en un Sistema de turnos, ya que lo que se pretende llegar, es a un sistema control en los horarios, del personal para una organización, dicho sistema deber soluciona las congestiones que se forman en el área administrativa como el departamento de financiero cuyos usuarios corren un gran riesgo de contagio frente a la pandemia actual además de que agiliza los tiempos de usuario por ventanilla y aumenta productividad de servicio por ventanilla, agendar turnos desde la comodidad de su casa llenando los datos correspondientes a su identidad y registrándolos, se tiene como premisa implementarlo en la un ámbito universitario más específicamente en el área de financiero pero el concepto y funcionalidad se puede aplicar a otros sectores que requieran y donde se pueda de este sistema.

Funcionalidades del sistema:

* Registro de identidad.
* Registro turno.
* Registrar los datos de una área.

Funcionalidades del usuario:

* Eliminación de turno.
* Selección de horario de turno.
* Cambio de turno.
* Diligenciamiento de formulario virtual. (En caso de ser un certificado de estudio)

1. (5 pts) Argumente en un audio de mínimo un minuto y máximo tres, por qué el Django y Angular son apropiados como backend y frontend, respectivamente, para el desarrollo de su proyecto.



1. (5 pts) Explique las ventajas y desventajas de usar una API Rest en el proyecto que está desarrollando.

Para hablar de API Rest (transferencia de estado representacional) es importante mencionar, que luego de que apareció PHP, donde todo se programaba como cliente\servidor, no había una división entre el programador BACKEND y FRONENTD, entonces aparece HTML5 en el 2008 en donde aparece las primeras API.  
Una ventaja de las API Rest se puede utilizar como ejemplo, que quieres diseñar una aplicación con UBER, entonces se requiere un geo localizador resulta que Google Maps, ya tiene el código que tú necesitas, lo que se necesitaría es comprar el código de algún desarrollador, el cual permite aprovechar el desarrollo que han hecho otras empresas para tu aplicación, esta es una gran ventaja.  
Y una desventaja del API Rest, es que, si uno necesita datos de los usuarios, siempre toca solicitarle a BACKEND un ENPOINT(URL) para cada dato que FRONTEND necesite, después, por lo tanto debes tener claro alguno que otro concepto básicos para que funcione correctamente, y no te lance error.

1. (10 pts) Los proyectos y aplicaciones generados con el CLI de Django generan una serie de carpetas y archivos base. Explique la estructura de ficheros y carpetas, describa los elementos clave y por qué son importantes desde el punto de vista arquitectónico.

El archivo Manage.py es una increíble herramienta de línea de comandos que te ayudará mucho. No solo es útil para crear nuevas aplicaciones dentro de su proyecto, sino para todas estas otras tareas y mas, La primera parte del código que trata con la Solicitud será el Distribuidor de URL (urls.py). Su objetivo principal es comprender qué APIView correspondiente debe instanciarse en función de la URL que se llamó y, finalmente, llamar al método GET, POST, PUT o DELETE de ese APIView según el tipo de solicitud.

Como segundo paso, APIView recibe los parámetros de la URL (si los hay) a través de los parámetros del método y luego ejecuta cualquier lógica codificada para procesar la solicitud o las interacciones del modelo de la base de datos. Por ejemplo: Obteniendo alguna información de la base de datos; guardar o borrar alguna otra información; enviando emails, etc.

El siguiente paso, debe ser decidir qué JSON debe devolverse dentro del "Body" de la Respuesta. Para eso tenemos el "serializer".

Finalmente, APIView debe devolver un objeto de respuesta que contenga el objeto JSON preparado por el serializer.

1. (10 pts) Grabe un audio en el que explique cómo instalar y crear un primer proyecto con Django. No hay límite de tiempo.



1. (10 pts) Realice un cuadro comparando las arquitecturas Model-View-Controller, Model-View-Presenter, Model-View-ViewModel y Action-Domain-Response.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CUADRO COMPARATIVO | | | | |
|
|  |  | VISTA | MODELO | Controlador |
| MVC | (modelo vista controlador) es uno de los proyectos más usados en la industria, para poder crear proyectos escalables y modulares, separa una aplicación en 3 componentes lógicos, la vista el modelo y el controlador, y se relacionan entre sí, ayudando a los programadores, a crear mejores aplicaciones que puedan ser entendidas y mantenidas en un futuro. | Vista: cuando se habla de la vista, se refiere a la interfaz gráfica, esta primera capa se encarga de obtener la información de lo que quiere el usuario a los cuales se llama eventos, la vista muestra siempre la información del modelo | el modelo tiene la responsabilidad de relacionar los datos, con los cuales una aplicación va a operar, esto a su vez se le conoce como la lógica de negocio es decir es la parte de una aplicación que se encarga de mapear las actividades del mundo real, a la forma que se va a modificar la información. | el cual responde a eventos o acciones que realice el usuario, por medio de la vista para poder solicitar una operación de información, también es responsable de elegir la vista que tiene que mostrarle al usuario, de acuerdo a la solicitud recibida, por lo cual es el puente que une al modelo con la vista. |
|  |  | VISTA | MODELO | Presentador |
| MVP | (modelo vista presentador) es un patrón de diseño, es un análisis y diseño el MVP que tiene con fin, separar la interfaz gráfica de la lógica de negocio que tiene una aplicación, los cuales son la vista el presentador y el modelo | es la interfaz de usuario, el cual tiene ventanas que interactúan con el usuario, también tiene como responsabilidad actualizar la interfaz que requiere el usuario. | es la lógica de negocio, es donde se realizan todas las funciones y toda la lógica que hace la aplicación, el procesa y devuelve un resultado a la vista. | el cual tiene funciones específicas, escuchar los eventos que vienen de la vista, validar la información que reciba de la vista, ejecutar lar acciones necesarias, para pasar la información al modelo y que los proceso. |
|  |  | VISTA | MODELO | MODELO-VISTA |
| MVVM | modelo-vista-modelo de vista, un es una arquitectura que maneja diferentes patrones, los cuales divide los subsistemas que tiene una aplicación, son los patrones arquitectónicos que los definen, como están compuestos en los sistemas y las diferentes interacciones que ocurre entre ellos, los módulos que los componentes son 3: | el producto que el usuario final ve al final de una aplicación, y a través de la cual interactúa con ella, se debe definir la estructura la apariencia y el comportamiento de los datos que genera la aplicación, la vista se relaciona con el modelo vista y esta relación ocurre a través de enlaces de datos, se establece un canal comunicación entre la vista y el modelo vista, el cual le informa de los cambios interacciones que hayan sucedido en ella, y el modelo vista le comunica los cambios que hayan ocurrido en el modelo. | es donde se encuentra la lógica de la aplicación y el acceso a datos, se pueden ver la conexión de la base de datos, validaciones, algún proceso que ejecute la lógica del negocio, maximiza la cantidad de código que podemos compartir del modelo. | es un intermediario entre la vista, y el modelo, en su patrón intermediario, encapsula la lógica de la vista de los datos que produce el modelo, tiene 3 responsabilidades, controlar las interacciones que suceden en la vista, traslada los datos que ingreso el usuario en pantalla hacia el modelo, tiene una capa de validación, como la cantidad de caracteres que se pueden ingresar en un campo, la segunda consiste en recuperar y transformar los datos generados en el modelo y enlazarlos con la vista, la tercera responsabilidad es notificar a la vista, de los cambios que sucedan en él y en sus propiedades. |
|  |  | ACTION | DOMAIN | RESPONDER |
| ADR | El patrón ADR es un refinamiento del patrón MVC, propuesto por Paul M. Jones, que encaja mejor para aplicaciones web allí donde más falla el MVC. El patrón emula la comunicación HTTP (request-response) de una forma más eficiente. Sus actores principales son: | Es el equivalente al controlador de MVC. Toma el input de la request y la utiliza para interactuar con el dominio, pasándole a un único responder el output resultante. | Equiparable al modelo de MVC. Interactúa con la base de datos y manipula la información, es de nuevo quien contiene la lógica de negocio. | Representa lo mismo que las vistas de MVC. Es quien muestra el resultado de la acción. |