DOCUMENTO GUÍA SPRINT PROYECTO HOTEL PARADISE

- Líneas generales sobre este sprint:

Bueno, como supondréis vamos a empezar a trabajar un poco más en serio en nuestro pequeño proyecto de Android, así que aquí os presento las líneas generales del trabajo para este sprint de 15 días y el reparto de tareas. No voy a hacer Google Keep salvo que lo vea conveniente, así que esto es lo que hay y espero que todos se lo lean, al menos la parte que les corresponda.

- Plazos de trabajo:

Aunque el sprint dure 15 días, en este sprint vamos a tratar de trabajar sobre una base de 10 días, con la sencilla razón de tener tiempo al final para arreglar los problemas que surjan y poder esta vez si probar las cosas antes de entregar el trabajo y que no nos pille el toro. Así que además de ir preguntando que tal vais, tened en cuenta esto… NO podéis entregar el último día del sprint, y sobre los que planeen ir contra esto vamos a hablar ahora.

- Aviso a los que no participen en el proyecto:

Si tras el plazo dado para entregar vuestro trabajo no lo habéis hecho, se os retirará del proyecto y se dará parte de este hecho. Os recuerdo que Andrés ha dicho que aquellos que no trabajen serán ELIMINADOS DEL PROYECTO, y que tendrán que hacer un trabajo/examen extra, lo que probablemente signifique un suspenso, visto que el profe no perdona. Allá vosotros los que no queráis hacer nada, yo voy a cumplir con mi obligación y no me va a temblar el pulso, os lo aseguro. De todas formas, os garantizo que en el reparto de trabajo no creo que nadie pase más de 2 horas haciendo su parte, en la mayoría de los casos incluso menos, así que no me valen excusas. Si no has leído esto, también es tu culpa, para eso me molesto en escribirlo.

- Dudas y preguntas:

Si tenéis dudas sobre el trabajo asignado me preguntáis a mi, no tengo problemas en responder a quien sea cuando sea, pero siempre DURANTE el sprint, quien me venga el último día a decir que no lo ha hecho porque “no sabe”, que se lea lo de arriba. De todas formas creo que los trabajos están bien distribuidos, de forma que sean asequibles para el nivel que tiene cada uno, así que pocas excusas quiero ver.

Sin más preámbulos distribuyo el trabajo con todo detalle, para que nadie se pierda, aún así LEED LAS NOTAS DEL FINAL, QUE SON IMPORTANTES!!!

DISTRIBUCIÓN DE TAREAS

- ALBERTO:

Trabajarás en dos campos, el main_activity, que no es otra cosa que el login falso, y la creación de los estilos del proyecto, el cual no debes molestarte tampoco mucho.

Para el main activity habrá dos campos para escribir texto, “Alias” y “password”, y habrá dos usuarios falsos del sistema: cliente, con password “cliente”, y admin, con password “admin”, imposible de olvidar. Después, como es lógico, un botón de “entrar”

En caso de que el login sea incorrecto, debes añadir un textview debajo de todo esto, que en principio esté vacío, y que diga “Login incorrecto” si el login falla.

Haz toda la codificación directamente en el main_activity.java

Si el login es exitoso, debe abrir una nueva activity que se llamara “menu_activity”, pero que tú NO DEBES CREAR, tu solo a esta activity.

Para la parte de los estilos es mu sencillo, creas un tema, le añades colores corporativos que te plazcan, una tipografía y un archivo dimen para las dimensiones. Usa chatGPT o cualquier herramienta que creas, recuerda que en interface dimos el tema este de selección de colores, que puede ser útil.

Debes añadir estas cosas en sus respectivos documentos xml en el proyecto, recuerda.

Si tienes dudas ya sabes donde estoy.

- BRUNO:

Debes crear 3 activities, la primera se llamará menu_activity y tendrá simplemente dos botones, “listar clientes” y “listar habitaciones”, y a está, superdifícil eh? RECUERDA!!!: las activities que crees deben estar acompañadas de su .java correspondiente, no crees solo el .xml, ya que tienes que escribir código java correspondiente, no son solo las vistas…

Cada uno de los botones llevará a una nueva activity , mediante intents, que crearás y que contendrán los recyclerview que mostrarán las listas de clientes y habitaciones. Debes crear estas dos activities con sus javas también. Se llamarán VistaHabitacion y VistaCliente. Además, estos botones llamarán al métodos en el Singleton, llamados, “obtenerHabitaciones” y “obtenerClientes”, los llamas Singleton.getSingleton().obtenerHabitaciones(), creo que es fácil.

En estas dos activities debes insertar los recyclerview correspondientes, que también has de crear, y que llamaras “listaClientes” y “listaHabitaciones”, además has de crear dos xml que son los items que van dentro del recyclerview, cuyos nombres serán itemCliente e itemHabitacion. No es necesario que te quiebres la cabeza con cuestiones de diseño, simplemente que se muestre la información y poco más, aunque si te quieres animar y hacer algo más bonito… acuérdate de darle al menos margenes para que los items se listen de forma separada, eso si. Recuerda que los atributos dentro de los items del xml debes nombrarlos como en la documentación, pero le añades tx delante, de forma que el nombre del textview que representa el id del cliente sería txIdCliente, y así sucesivamente. Mira las tablas en la documentación y crea un textview por cada valor

Por cierto, cada .java de estas dos activities, deben tener un método llamado “cargarDatos()” que recibe como parámetro un ArrayList con el objeto correspondiente, si es clientes el ArrayList será ArrayList<Cliente>, por ejemplo, y que deberán cargar en el adapter ese listado que te llega. Repito, mira los ejemplos de la AEMET y eso para que no te pierdas.

IMPORTANTE--→ Ignora las imágenes de momento, ya nos ocuparemos más adelante, solo el texto debes meter en los items.

Espero que recuerdes con los proyectos de la aemet como se instancian los recyclerview en las activities, por ejemplo en el activity de los clientes poner todo el tema de en el onCreate poner

```
miRecyclerView = findViewById(R.id.listaCLientes);
```

y esas cosas. Mira las apps que hemos hecho y te ayudas. Eso es todo.

- ADAN:

Vas a ocuparte de hacer las “clases simples” del modelo, a saber “Hotel”, “Cliente”, “Habitacion”. De momento solo esas tres, ya que la de reserva es más compleja y será más adelante. Estas clases son las de la APP, NO LAS DEL SERVIDOR, o sea, haces un pull del proyecto, te abres el android Studio abriendo el proyecto, y creas las clases. Debes crear los constructores de todas las clases, con todos los atributos. Mira la documentación para saber cuales son los atributos que tiene cada clase. Debes también crear los getters de todos los atributos de todas las clases.

NO crees atributos de imágenes, las imágenes son para más adelante, si ves atributos de fotos de habitación y esas cosas, lo ignoras.

En la clase Hotel, los atributos son dos, un ArrayList de clientes, y un ArrayList de habitaciones. Añade los getters de ambos. Debes inicializar ambos ArrayList en el constructor, o sea, los declaras como atributos y haces los new en el constructor.

- JOSEDA:

Vas a hacer algo similar a Adan, pero esta vez lo harás en el servidor. Es muy fácil. Es lo mismo que ha hecho Adan, las clases sencillas del modelo, “Hotel”, “Cliente”, “Habitacion”. Para ello el mismo método, un pull del proyecto desde powershell o donde sea, y en tu propia rama de proyecto, arrancas el intelliJ IDEA Ultimate… recuerda, el ULTIMATE… y desde allí creas las clases como cualquier otro proyecto java, no hay más misterio. Después haces un merge con git en la rama develop y arreglado. Si tienes dudas sobre como hacer esto, me pillas en clase y me preguntas, o a cualquier compañero, es fácil.

Recuerda que los atributos de cada clase están en la documentación del proyecto, los típicos nombre, dni y tal para clientes y eso. Para la clase “Hotel” creas dos atributos, que son dos arrayList uno de clientes, y otro de habitaciones. Crea todos los getters para todos los atributos, además de su constructor obviamente. Inicializa los arrayList de Hotel en el constructor, por cierto.

Dudas ya sabes donde estamos. Suerte!

- ÁNGEL:

Tienes que crear los viewHolder y viewAdapter del proyecto, en este caso dos de cada, uno para listar clientes, y otro para listar habitaciones. Puedes copiar lo que hemos hecho de AEMET y adaptarlo, es fácil. Recuerda que para los atributos, estos están en la documentación del proyecto, y que para el caso de la información que te llega de los item.xml, QUE NO DEBES CREAR, no es tu trabajo, estos se llamarán igual que en la documentación del proyecto, solo que añadiendo un tx

delante, de forma que IdCliente en el item debería llamarse txIdCliente, para que lo tengas en cuenta a la hora de adaptar el ViewHolder.

Además, como es poca cosa esto de copiar y pegar los adapter y holder, vas a crear un Singleton que tenga métodos para recibir y ejecutar órdenes, e instanciar las clases Peticion y Respuesta. Te los resumo.

Debes instanciar la clase petición, como un atributo, tipo private Peticion peticion;
Debes instanciar la clase respuesta, como un atributo, tipo private Respuesta respuesta;

Inicializa esas instancias en el constructor, es lo mejor creo.

Dos atributos que seran ArrayList, uno de ellos de clientes, que se llamará listadoClientes, y otro de habitaciones, que se llamará listadoHabitaciones.

Y por último instanciar las dos clases donde estarán los recyclerview, creando objetos de ambas, que se llaman “VistaHabitacion” y “VistaCliente”, que es donde enviaras los listados de los clientes y habitaciones.

Pasamos a los metodos:

Un método que se llamará “obtenerClientes” que llamará a un método de la clase Peticion, mediante el objeto instanciado, llamado “listarClientes”. No se le envía ningún parámetro, lo haremos así

Un método llamado “parsearListadoClientes” que recibe de parámetro un String, y que llamará a un método de la clase Respuesta llamado “parsearClientes”, mediante el objeto respuesta, enviándole este String, y que devolverá un ArrayList<Clientes>, que cargaremos en nuestro “listadoClientes”, que tenemos como nuestro atributo de clase. Después llamaremos con el objeto instanciado de la clase VistaCliente, llama al método “cargarDatos()” enviándole este ArrayList.

Por otro lado, un método que se llamará “obtenerHabitaciones” que llamará a un método de la clase Peticion llamado “listarHabitaciones”. No se envían parámetros.

Un método llamado “parsearListadoHabitaciones” que recibe como parámetro un String, y que mediante el objeto respuesta, llamará a un método de la clase Respuesta llamado “parsearHabitaciones” enviándole el String que hemos recibido por parámetro, y que devolverá un ArrayList<Habitaciones>, que cargaremos en nuestro “listadoHabitaciones” que es un atributo de la clase Singleton. Después en ese mismo método llamaremos con el objeto instanciado de la clase VistaHabitacion al método “carcarDatos()” enviándole este ArrayList.

Suerte, es fácil, y si tienes dudas ya sabes donde estoy.

- RUEDA:

Voy a darte dos tareas sencillas. Primero, vas a crear una clase Peticion en el Android Studio, lo que es el proyecto del movil, que tenga dos métodos “listarClientes” y “listarHabitaciones”, que no reciben parámetros, pero que usarán como constante las URL que vamos a definir para hacer las peticiones al servidor Tomcat. Crea una constantes pues que sea del tipo
“localhost:7777/HotelServlet/HotelServidor?peticion=”
Y luego, dentro de cada método, una variable String cuyo valor sea “clientes” para listarClientes, y “habitaciones” para “listarHabitaciones”, de modo que al crear la URL de cada método la petición sea igual a clientes o habitaciones en cada caso.

El resto es fácil, lo copias de nuestros proyectos de AEMET para hacer ambas peticiones. Eso si, el String que te de de vuelta, se lo mandas al Singleton, mediante los métodos

Singleton.getSingleton().parsearListadoClientes(respuesta) y Singleton.getSingleton().parsearListadoHabitaciones(respuesta)… Super fácil no?

- MORALES PERICH:

Entra con el IntelliJ IDEA Ultimate dentro del servlet y crea una nueva clase, llamada Respuesta, que creo que será lo mejor. En esta clase debes tener dos métodos, que reciben dos ArrayList de cada tipo de objeto como parámetro, que vendrán de lo que haga la clase ConexionDB. Cada uno de los métodos se han de llamar clientesJson y habitacionesJson, y deben tener como valor de retorno un String que contenga ese JSON, supongo que con un asString o lo que tenga la librería para enviarlos como strings.

No hará falta que te diga que el trabajo de esos métodos es parsear estos ArrayList de objetos en Jsons, y enviarlos como Strings… donde? Pues vamos a hacer trampas y nos saltaremos el Controlador esta vez porque me da la gana y soy el jefe. Así que el string lo mandas en el return al método que lo llamó desde ConexionDB y asunto arreglado jijijij…

Y ahora viene la segunda parte del trabajo, y es ir al proyecto de Android y hacer lo contrario, tomar el string que te viene de respuesta y en dos métodos de la clase que también debería llamarse Respuesta, que has de crear, y que los nombres de estos métodos serán “parsearClientes” y “parsearHabitaciones”, que son llamados desde el Controlador, y que como parámetro reciben el string json, devolver un ArrayList de objetos Cliente y Habitacion en el return, en cada método respectivamente. De todas formas, ya iremos viendo si es demasiado o no, lo vamos hablando ok?

- JUANDE:

Vamos a hacerte trabajar en el servlet, pero tranquilo, que es fácil. Abre el proyecto con IntelliJ IDEA ULTIMATE, pues tienes que hacer dos cosas. La primera es trabajar en lo que ahora es la clase Servlet, la cual refactorizas y le cambias el nombre a HotelServidor, que a Andrés no le gustaba la otra. Después verás un switch en el método proccessRequest que es el que hace la magia, ya que los doPost y doGet no los usamos aquí. Qué hacer con esto? Cambia el switch y que ahora tenga dos opciones en los “case”, si “cadenaPrueba” es “clientes” haz que el Controlador Singleton vaya a un método que se llame “obtenerClientes”, y si es “habitaciones”, haz que el Controlador singleton vaya a un metodo que se llame “obtenerHabitaciones”. Recuerda que ambos métodos recibiran una variable String llamada “respuesta”, que la necesitamos. Pon un mensaje de error en el default si quieres, y deja el resto por ahora.

En la clase Controlador, crea los métodos que hiciste antes y si quieres comenta los métodos “hacerprueba” y “hacerinsert” que ya no sirvan de antes, pero deja lo que sirva ojo. Los métodos que has creado lo que han de hacer es mediante el objeto “conectar” de la clase “ConexionDB”, llama a dos métodos, que han de devolver un String que le puedes llamar “respuesta” si quieres. Estos métodos de ConexionDB se llamarán “selectHabitacion” y “selectCliente”, recuerda que devuelven un String. Estos métodos deben devolver en un return ese String de respuesta que irá a la clase HotelServidor, desde la que hemos llamado al controlador.

Y ya por último vuelve a la clase de antes, que ahora debería llamarse HotelServidor, baja más abajo del switch y cambia todo lo que hay dentro de “out.print” dentro del try, y en su lugar que imprima la variable “respuesta” que nos llega del controlador. EEEEAZYY!!!

- MARIANO:

Debes abrir el servlet con IntelliJ IDEA Ultimate, que está en la carpeta servlet, y modificar la clase ConexionDB. Lo primero es que tienes que cambiar en la llamada a la base de datos en el getConnection el “pruebashotel” por “hotelparadise”, que es como vamos a llamar a la base de datos.

De momento, creo que debemos dejar que sea el usuario postgres y la contraseña que puse por defecto los que manejen la base de datos, ya veremos más adelante.

Comenta los métodos de prueba si quieres, y crea dos nuevos métodos: selectHabitacion y selectCliente. Que devuelven un String. Lo que estos métodos hacen es obvio, llaman a la base de datos y hacen un select * from cliente y habitacion respectivamente. Después recorre el resulset y crea objetos de las clases cliente o habitacion según sea y ve metiéndolos en un ArrayList , el cual al final lo vas a enviar a un método de la clase Respuesta. Instancia mejor la clase respuesta private Respuesta respuesta; e inicializala en el constructor. Es incorrecto hacer esto, pero lo haremos porque estamos en modo pruebas, luego habría que cambiarlo, pero de momento vamos a tirar así.

Como decía, llama a los métodos de la clase respuesta, estos se llaman clientesJson() y habitacionesJson() enviándoles su ArrayList respectivo. Estos métodos devuelven un String, así que guárdalos en una variable, ya que este String es el return de estos métodos que has creado en la base de Datos. De momento eso es todo. Ya veremos más adelante.

- MARCO Y LOS CHICOS DE PRIMERO:

Siento llamaros a todos de golpe y porrazo sin nombres ni nada para los de primero, pero es por abreviar. Además designo a Marco de segundo para que trabaje con vosotros en lo que queda, que es ni más ni menos que la base de datos.

El trabajo es sencillo, y esta vez espero que os lo repartais porque no es gran cosa.

Primero, haced el docker-compose. Podeis tomar como referencia el mio, que está en la carpeta Servlet/Tomcat del proyecto. Adaptadlo pero tened en cuenta algunas cosas. Debéis mantener la persistencia en volúmenes como está ahí (más o menos, podeis cambiar algo), pero que ha de tener persistencia para la base de datos y para meter el war del tomcat, mirad bien el docker compose para daros cuenta de como lo hago yo. Por cierto, cambiad el nombre de la base de datos, debe ser “hotelparadise”, no pruebashotel como está ahora. El password de momento dejadlo igual.

Después, cread las tablas como en la documentación, sin olvidar las claves foráneas y tal. Un truco está en arrancar el docker-compose, y con el PGAdmin4, conectais a la base de datos arrancada en docker, y podeis meter los datos muy fácilmente desde la interfaz.

Debeis no solo crear las tablas, si no también introducir al menos 3 clientes de prueba, y 2 habitaciones de cada tipo, que serán listadas

NO debeis tocar el tema de imágenes, si es que hay en el diseño, de momento pasad de ese tema.

Cuando terminéis de crearlo todo y meter datos, teneis que hacer un dump de la base de datos, o sea, obtener el .sql, que llamares igual que la base de datos “hotelparadise.sql”

Y ya para cerrar, una vez tengais todo hecho, cread una nueva carpeta que sea solo para el docker, la llamais como queráis, y subís todo a github, en la rama develop.

Si teneis dudas ya sabeis donde estoy, y si no los compañeros… PERO NECESITAMOS LA BASE DE DATOS, POR FAVOR NO OS RETRASEIS!!!!

NOTAS IMPORTANTES: LEER!!!!!!!

- Respetad los nombres de clases, activities, variables y métodos que se os dicen, porque otros las van a usar en su trabajo, si os equivocáis nos equivocamos todos!!!

- NO pongáis tildes o acentos o como los llaméis en los nombres de las cosas, que después da problemas, o sea, la clase no se llama “Habitación”, si no “Habitacion”, sin tilde/acento.

- Trabajad en vuestras propias ramas de git, y después hacéis un merge a la rama develop cuando terminéis, easy.

Por último, **SE QUE VAN A SURGIR PROBLEMAS, PERO CUANDO LOS TENGAMOS, LOS HABLAMOS Y LOS RESOLVEMOS, NO PODEMOS ACERTAR A LA PRIMERA, PERO SI NO NOS PONEMOS, NO LO HAREMOS NUNCA.**

A currar esclavos!!!!