**Colores:**

**Habla maria Jose**

**Habla Tobias**

**Numero diapositiva**

**PRESENTACION COMPUTACION MOVIL: “Garage Sale”**

**Introducción (Diapositiva 2)**

Introducir la presentación…

**Proposito (Diapositiva 3)**

GarageSale es una aplicación de subasta sencilla. Hemos creado un sistema que permite comprar y vender productos usados a buen precio (por eso “venta de garage”) en base a las ofertas que sugieren los compradores. Es decir: uno quiere vender un cierto producto (digamos un Iphone), sube la publicación con un determinado “precio base” y tiene determinado tiempo (pueden ser 3, 7 o 14 dias dependiendo la necesidad urgente o no de venderlo) para que los compradores miren el producto y sugieran un precio.

A lo largo de ese tiempo los compradores podrán ver la oferta mas alta (y tratar de superarla) ya que, la persona que se quedara con el producto al final es la que ofrezca un precio mas alto.

Como no conocíamos android decidimos orientar la aplicación a un sistema sencillo pero completo para aprender a utilizar un web service que nos permita hacer las tres tareas basicas: añadir, eliminar y modificar.

**Mockups (Diapositiva 4 y 5)**

(mostrar mockups y como funciona cada pantalla) Mencionar que nos basamos en eso pero hemos hecho énfasis en la interfaz de usuario (mejorado).

**Estructura (Diapositiva 6)**

Hablar del mind map de la presentación.

**Recursos Android (Diapositiva 7 y 8)**

Mencionar los recursos de la ppt

**Herramientas utilizadas (Diapositiva 9 y 10)**

En primer lugar, cabe destacar que utilizamos Java, el lenguaje nativo que utiliza Android. Hemos hecho uso de Android Studio.

 Creamos una aplicación android que consume los datos de un servidor externo a través de **Php**, **Mysql** y **Json**.

Antes de crear la aplicación Android debíamos desarrollar primero el Web Service con cualquiera de los estándares que nos interesaran. Simplemente decidimos crear las implementaciones Php necesarias para realizar operaciones sobre una base de datos en Mysql a través de peticiones GET y POST.

Para desarrollar la aplicación utilizamos [el entorno de desarrollo XAMPP](https://www.apachefriends.org/es/index.html), el cual provee automáticamente una configuración de un servidor *Apache* local, el intérprete de *Php* y el gestor *Mysql es decir, es un software que integra todo en una sola aplicación*.

Creamos una nueva base de datos en la aplicación *phpMyAdmin* que te otorga tu distribución *XAMPP.*

**Que es Xampp? (Diapositiva 11 y 12)**

XAMPP es un servidor independiente de plataforma de código libre. Te permite instalar de forma sencilla Apache en tu propio ordenador, sin importar tu sistema operativo (Linux, Windows, MAC  o Solaris). Y lo mejor de todo es que su uso es gratuito. Incorpora también el intérprete de PHP.

XAMPP te permite probar tu trabajo (páginas web o programación por ejemplo) en tu propio ordenador sin necesidad de tener que accesar a internet.

La mayor ventaja de Xampp es que es muy fácil de instalar y las configuraciones son mínimas o inexistentes

Como desventaja, podemos mencionar la seguridad.

Xampp no es para uso en producción, sólo es un entorno de desarrollo local. Xampp está configurado lo más abierto posible, permitiendo que el desarrollador trabaje sin restricciones.

Para entornos de desarrollo esto es muy útil, pero en un entorno de producción es poco seguro ya que tiene ciertos detalles como por ejemplo el administrador de MySQL (*root*) no tiene contraseña. Tambien, Xampp trae PhpMyAdmin para administrar las bases de datos de MySQL,  sin embargo para tareas más específicas es mejor utilizar la consola (linea de comandos) y Xampp no la soporta.

**Estructura de la Base de datos (Diapositiva 13)**

Creamos las respectivas clases que mapean la estructura de la tablas de la base de datos. El objetivo de ello es proveerlas de comportamientos de inserción, actualización, eliminación y consulta a través de la conexión a la base de datos.

La base de datos….. (hablar de la estructura)

**Conexión cliente- BDD (Diapositiva 14)**

Como creamos nuestra conexión a la BDD? En nuestro caso creamos una conexión con PDO, la cual nos permite proteger los datos de inyecciones sql.PDO es el acrónimo de ***PHP Data Object***.

Que es PDO? PDO es la capa de abstracción de objetos para poder manipular los datos de las bases de datos de forma estándar. Es decir, no existen métodos propios de cada base de datos, si no que PDO define un interface estándar que las bases de datos tienen que soportar.

De esta forma conseguimos que un programa [PHP](http://www.manualweb.net/tutorial-php/) construido con PDO sirva para acceder a cualquier base de datos. O, al menos, en teoría así sería.

El uso de PDO depende del enfoque que tengan tus proyectos, puedes crear una clase que represente la conexión hacia la base de datos o simplemente crear una nueva conexión en cada script de Php que tengas.

**(Diapositiva15)**

Para este caso utilizamos un *patrón singleton de PDO* para limitar el número de aperturas a la base de datos en una sola. Con ello podremos disponer de un solo objeto a través de todo el proyecto.

Finalmente implementamos scripts Php para gestionar las peticiones que lanzan los clientes. La idea es parsear los datos en formato Json para que nuestra aplicación Android interprete los resultados de forma legible.

***¿Cómo envío una respuesta de vuelta a la aplicación Android?* (Diapositiva 16)**

*Es justo donde entra* ***Json*** *para actuar como formato de comunicación. En cada respuesta enviaremos una serie de elementos Json que puedan ser interpretados del lado del cliente. Esto es posible usando las funciones json\_encode() y json\_decode(). La primera parsea un tipo de dato a un string en formato json y la segunda es el procedimiento contrario.*

***Google volley* (Diapositiva 17)**

Luego hicimos uso de **Volley** de google. **Volley** es una librería desarrollada por Google para optimizar el envío de **peticiones Http** desde las aplicaciones Android hacia servidores externos. Volley está totalmente enfocado en las peticiones, evitando la creación de código repetitivo para manejar tareas asíncronas por cada petición o incluso para parsear los datos que vienen del flujo externo. Gracias a esta biblioteca podemos hacer peticiones a servidores web que nos devuelvan un JSON o una imagen.

Entre sus características mas potenciadoras podemos encontrar:

▪ Procesamiento concurrente de peticiones.

▪ Priorización de las peticiones, lo que permite definir la preponderancia de cada petición.

▪ Cancelación de peticiones

▪ Gestión automática de trabajos en segundo plano

▪ Implementación de caché en disco y memoria.

▪ Capacidad de personalización de las peticiones.

▪ Provee información detallada del estado y flujo de trabajo de las peticiones en la consola de depuración.

Para utilizarlo se crea una cola y añadiendo después peticiones a esa cola, y Google nos recomienda que mantengamos una sola cola para toda la aplicación, por lo que haremos un Singleton con este propósito.

***Flujo de datos* (Diapositiva 18)**

Mostrar el esquema de la ppt

**Mas librerías utilizadas (Diapositiva 19)**

Luego usamos la **Google Places API** para brindar la ubicación de un producto a la hora de concretar una venta.

Añadimos el **FAB** (Floating Action Button) haciendo uso de la siguiente librería que existen en la web: [Floating Action Button Library For Android](https://github.com/shell-software/fab)

Finalmente, mencionamos la libreria **butterknife**. [Butter Knife](https://github.com/JakeWharton/butterknife) es una librería que nos facilitará la tarea de **relacionar los elementos de las vistas con el código** en nuestras aplicaciones [Android](http://jonsegador.com/category/android/). Nos evitará tener que utilizar siempre **findViewById** y simplificará el código.

**Problemas mayores (Diapositiva 20)**

Fotos? Chat?

**DEMO (Diapositiva 21)**

(Mostrar video) Podemos hablar los dos.