ESCUELA INTERNACIONAL DE GERENCIA eig

Proyecto fin de Ciclo: Aplicación HuertoMatic

DESARROLLO DE APLICACIONES MULTIPLATAFORMA

TFC

Sergio Jairo Mejías Puentes

31/05/2021

Tabla de contenido

1. Objetivo 4
2. Justificación 4
3. Análisis de la competencia. 5
3.1. Huerta en casa 5
3.2. Ecohuerto 5
3.3. MacetoHuerto 6
4. Propuesta detallada 6
4.1. Sketch 6
4.2. WireFrame 7
4.3. Mockup 7
5. Planificación Temporal 8
5.1. Especificación de los objetivos de las iteraciones 8
6. Diseño de la aplicación 9
6.1. Diseño de estructuras de datos 9
6.2. Estudio de la interacción del usuario con la aplicación 12
6.3. Guía de estilo 12
7. Codificación 13
7.1. Lenguajes, herramientas y metodologías empleadas 13
7.2. Aspectos destacables de la implementación 13
8. Evaluación y pruebas 13
9. Manual de usuario 14
9.1. Descripción de la aplicación 14
9.2. Funcionalidad y características 15
10. Conclusiones 15
11. Bibliografía y documentación 16
11.1. Webs expertas en la materia 16
11.2. Android: 16

1. Objetivo

La necesidad de volver a nuestros orígenes, es un hecho en la actualidad, debido a la individualidad de las urbes, la ajetreada vida en las mismas y el agotamiento físico y mental de los ciudadanos, proliferando en las mismas los llamados “Huertos Urbanos”, que no dejan de ser parcelas de tierra cultivable donde una persona no versada puede ejecutar todo el proceso de cultivo y recolección de hortalizas.

Tras un estudio intensivo de mercado, se plantea la necesidad de volver a retomar los conocimientos perdidos a lo largo de generaciones socializadas en ambientes urbanos.

Como beneficio de los mismos están que los niños al ver y tocar con sus propias manos lo que se van a comer, lo sienten más amigable que su incorporación automática a nuestra despensa desde un envase.

La propuesta es bien sencilla:

Como **objetivo general**, se creará una aplicación mediante la cual se pueda resolver cualquier duda sobre los periodos de producción de un huerto en determinada zona geográfica comenzando por la zona de Granada.

Dentro de esta aplicación encontraremos varios apartados referidos a:

- **Biblioteca de variedades:** para un mayor conocimiento de lo que se quiere sembrar
- **Detalles individuales de cada vegetal:** donde podremos obtener la información de siembra, cosecha y datos importantes de cada uno como asociaciones beneficiosas y perjudiciales de cada uno de los productos

2. Justificación

La base para la creación de esta aplicación nace en una conversación familiar, en la cual, se discutía sobre cuando era la mejor época para plantar zanahorias. El resultado de la conversación (los dos tenían razón) fue lo que determinó que la quisiera llevar a cabo.

Los periodos de producción de la zona Norte de Granada son distintos de la zona centro o la zona Sur… ¿Por qué no tener una aplicación donde pudieras consultar los datos que te hicieran falta para producir tus propios vegetales a pequeña escala?

3. Análisis de la competencia.

3.1. Huerta en casa

Es una aplicación bastante buena donde puedes encontrar múltiples variedades de plantas, y datos sobre su cultivo, asociaciones beneficiosas, perjudiciales, y datos curiosos sobre sus aplicaciones culinarias y en otros ámbitos. Como desventaja, es que la aplicación no contempla

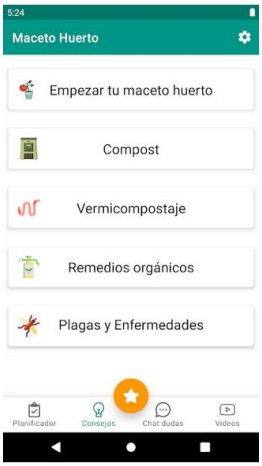
muchas de las hortalizas y que se centra en los cultivos de zonas tropicales.

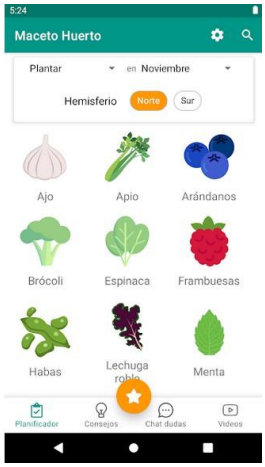
3.2. Ecohuerto

La aplicación fue desarrollada por un particular, y se basa en una agenda de huerto donde insertar tus observaciones y dependiendo del mes en el que estés darte consejos y buenas prácticas para poder crear tu huerto ecológico reciclando materiales, lleva sin actualizarse desde 2014.

3.3. MacetoHuerto

Esta aplicación Está desarrollada por el Cero Ideas Mobile Content Development y fue la primera aplicación para huertos en español. Está muy centrada para el desarrollo de huertos en pequeños espacios, por lo que para mucho de





vosotros puede que sea la aplicación ideal.

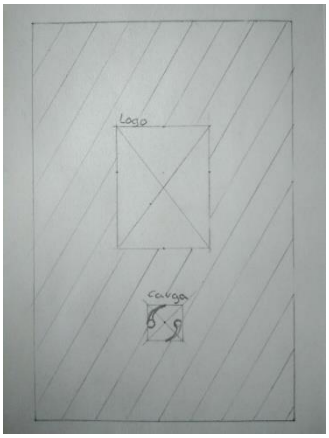
Posee un gran contenido de información para comenzar a sembrar, desde tipos de hortalizas, propiedades, tipos de cultivo, información de sustratos, consejos para el riego… Como aplicación está muy bien estructurada y con una interfaz intuitiva y sencilla.

Inconvenientes: como aplicación se centra demasiado en (como su propio nombre indica) el cultivo en terrazas y lugares muy reducidos, dejando el cultivo de mayor extensión sin cobertura.

Hasta día de hoy, Lleva sin ser actualizada desde 2013.

4. Propuesta detallada

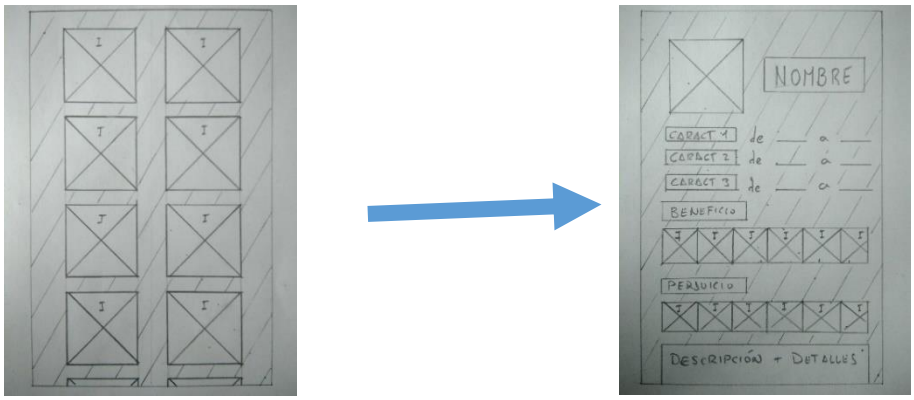
4.1. Sketch



La primera imagen corresponde a la Splash Screen, donde en un futuro se cargarán si es necesario todas las dependencias y se ejecutará lo necesario para que la aplicación funcione.

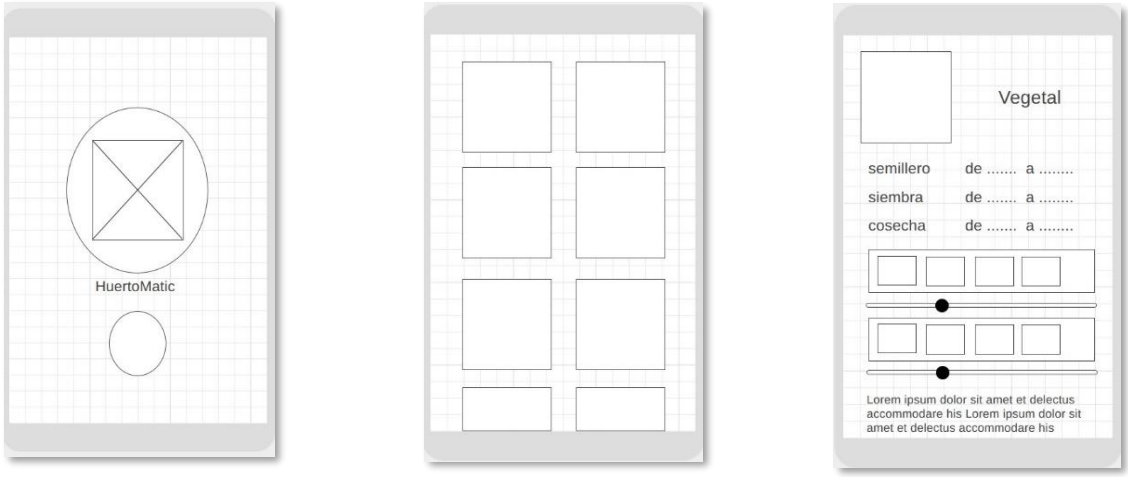
El diseño es bien sencillo, el logo de la aplicación, con un símbolo de carga.

De ahí pasamos a la bibliotefca de variedades donde se mostrarán todos los vegetales disponibles.



Y una vez que se pulse cualquier vegetal, nos llevará a los detalles de cada uno de los objetos, donde se observará una imagen descriptiva, los peridodos de cada característica, y dos scrolls donde existirán objetos beneficiosos y perjudiciales que a su vez conducirán a cada una de sus características.

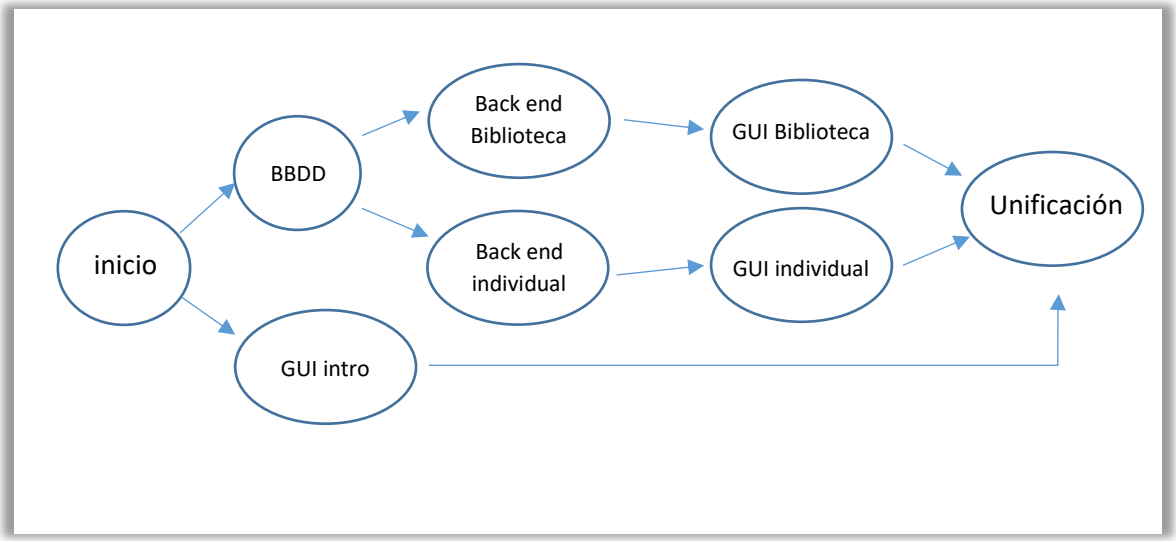
4.2. WireFrame



4.3. Mockup

5. Planificación Temporal

5.1. Especificación de los objetivos de las iteraciones

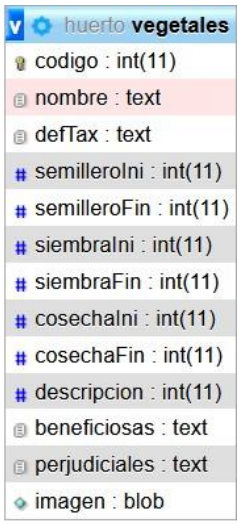


Tarea	Función a desarrollar	Duración	Dependencias
1	Diseño de la BBDD	3 días	-
2	Back end Vegetal Individual	7 días	BBDD
3	Back end Biblioteca	7 días	BBDD
4	GUI Biblioteca	7 días	Back end Biblioteca
5	GUI Individual	7 días	Back end Individual
6	GUI intro	4 días	-
7	Unificación de Fragments y animaciones	7 días	Todo lo anterior

6. Diseño de la aplicación

6.1. Diseño de estructuras de datos

BBDD: La base de datos consiste en una única tabla tal y como se muestra en la imagen.



- Como datos de interés trataremos los meses del año como int para posteriormente convertirlos en la aplicación a texto.

- En la misma guardaremos todas las imágenes que nos hagan falta para cada vegetal (una única imagen).

- Las asociaciones beneficiosas y perjudiciales, las tratamos como strings y en el back end convertimos cada uno de los nombres en una query para seleccionar el objeto

En el back end de biblioteca, creamos un GridLayout porque el scroll view no permite posicionar de manera sencilla los objetos en columnas y le pasamos los datos mediante un bundle

```
@Override
public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);
    // preparamos nuestros elementos para posteriormente rellenar lo que necesitemos
    sc = (ScrollView) view.findViewById(R.id.ScrollViewBiblio);
    // utilizamos Grid Layout porque las demás opciones no permiten eel posicionamiento en columnas sin tener que anidar
    GridLayout GrL = (GridLayout) sc.getChildAt( index: 0);
    Bundle data = new Bundle();
    // creamos una instancia a la base de datos
    AdminSQLiteOpenHelper administrador = new AdminSQLiteOpenHelper(getContext(), name: "huerto.db", factory: null, version: 4);
    SQLiteDatabase BaseDeDatos = administrador.getWritableDatabase();

    // variables donde guardaremos los datos que nos hacen falta.
    ArrayList<byte[]> imagenes = new ArrayList<>(); // imagenes descargadas de la base de datos
    ArrayList<String> nombres = new ArrayList<>(); // los nombres de los vegetales


    Cursor fila = BaseDeDatos.rawQuery( sql: "SELECT * FROM vegetales", selectionArgs: null); // hacemos la consulta y la ejecutamos
    while(fila.moveToNext()) {
        SQLiteDatabase BaseDeDatos = administrador.getWritableDatabase();
        String nombreVegetal = fila.getString( columnIndex: 1);
        byte[] imagen = fila.getBlob( columnIndex: 12);
        // lo que obtengamos lo añadimos a los ArrayList anteriormente creados
        imagenes.add(imagen);
        nombres.add(nombreVegetal);

    }
        int size = 600; // parametrizamos los tamaños para no tener que cambiar 4 veces lo mismo
        lp.width= size;
        lp.height= size;
        lp.setMargins( left: size*20/100, top: size*20/100, right: size*20/100, bottom: size*20/100); // establecemos los margenes externos
        iconoVegetal.setLayoutParams(new ViewGroup.LayoutParams(lp)); // seteamos los parametros anteriores
        iconoVegetal.setPadding(padding,padding,padding,padding); // establecemos los margenes internos
        iconoVegetal.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                data.putInt("0", finalI);
                Navigation.findNavController(v).navigate(R.id.vegetalIndividual, data);
                Toast.makeText(getContext(), text: "Vamos a plantar...." + nombres.get(finalI-1), Toast.LENGTH_SHORT).show();
            }
        });
        GrL.addView(iconoVegetal);
    }
```

En el back end de vegetal individual, hay detalles importantes como la recuperación de los datos de la BBDD y la transformación de los mismos para conseguir propósitos distintos a mostrarlos únicamente utilizando ArrayList en su mayoría donde almacenamos

```
    valorBundle = getArguments().getInt( key: "0");
    ArrayImagenesBeneficiosas = new ArrayList();
    ArrayImagenesPerjudiciales = new ArrayList();
    idsBeneficio = new ArrayList();
    idsPerjuicio = new ArrayList();
    AdminSQLiteOpenHelper administrador = new AdminSQLiteOpenHelper(getContext(), name: "huerto.db", factory: null, version: 4);
    SQLiteDatabase BaseDeDatos = administrador.getWritableDatabase();
    Cursor fila = BaseDeDatos.rawQuery( sql: "SELECT * FROM vegetales WHERE codigo = "+ valorBundle, selectionArgs: null);

    while(fila.moveToNext()) {
        ArrayBeneficiosas = new ArrayList();
        ArrayPerjudiciales = new ArrayList();
        //obtenemos de la BBDD todos los datos que nos hacen falta

        String strBeneficiosas = fila.getString( columnIndex: 10);
        String [] vectorBeneficio = strBeneficiosas.split( regex: ", ");
        for(int k = 0; k< vectorBeneficio.length; k++){
            ArrayBeneficiosas.add(vectorBeneficio[k]);
        }
        String strPerjudiciales = fila.getString( columnIndex: 11);
        String [] vectorPerjuicio = strPerjudiciales.split( regex: ", ");
        for(int l = 0; l < vectorPerjuicio.length; l++){
            ArrayPerjudiciales.add(vectorPerjuicio[l]);
        }
    }
    imageV = (ImageView) view.findViewById(R.id.imagen);
    Buscar();
    crearArrayImagenes(ArrayImagenesBeneficiosas, ArrayBeneficiosas, idsBeneficio);
    crearArrayImagenes(ArrayImagenesPerjudiciales, ArrayPerjudiciales, idsPerjuicio);
    rellenarScroll(LinLBen,ScrollBeneficios,ArrayImagenesBeneficiosas, idsBeneficio);
    rellenarScroll(LinLPer,ScrollPerjudiciales,ArrayImagenesPerjudiciales,idsPerjuicio);
}
```

```
/**
 * Método para crear un Array de imagenes que posteriormente se mostrarán en los Scrolls de beneficiosas y perjudiciales
 * @param ArImagen Parámetro del array de imagenes que queremos mostrar
 * @param ArrayNombres Array de nombres donde añadioremos las asociaciones
 * @param ArrayIds Este nos hace falta para posteriormente enlazar mediante un bundle al siguiente fragment
 */
public void crearArrayImagenes(ArrayList<byte[]> ArImagen, ArrayList<String> ArrayNombres, ArrayList<Integer> ArrayIds){
    AdminSQLiteOpenHelper administrador = new AdminSQLiteOpenHelper(getContext(), name: "huerto.db", factory: null, version: 4);
    SQLiteDatabase BaseDeDatos = administrador.getWritableDatabase();
    for (int i = 0 ; i< ArrayNombres.size() ; i++ ){
        Cursor fila = BaseDeDatos.rawQuery( sql: "SELECT * FROM vegetales WHERE nombre = \""+ArrayNombres.get(i) +"\""  , selectionArgs: null);
        while(fila.moveToNext()) {
            int id = fila.getInt( columnIndex: 0);
            byte[] imagen = fila.getBlob( columnIndex: 12);
            ArImagen.add(imagen);
            ArrayIds.add(id);
            System.out.println("--> " + imagen);
        }
    }
}
```

```java
/**
 * Método para rellenar los Scrolls ya que tenemso distintos Arrays
 *
 * @param linLay es el LinearLayout donde queremos añadir las imagenes
 * @param hsc el Scroll Horizontal dentro del Linear Layout donde posicionaremos las mismas
 * @param ArImagen El array de imagenes a añadir
 * @param ArrayIds Los ids, para pasarmos por bundle.
 */
public void rellenarScroll( LinearLayout linLay, HorizontalScrollView hsc, ArrayList<byte[]> ArImagen, ArrayList<Integer> ArrayIds){
    linLay = (LinearLayout) hsc.getChildAt( index: 0);
    Bundle data = new Bundle();
    for (int i = 0; i < ArImagen.size(); i++) {
            int finalI = ArrayIds.get(i);
            ImageView iconoVegetal = new ImageView(getContext());
            iconoVegetal.setImageBitmap(BitmapFactory.decodeByteArray(ArImagen.get(i),  offset: 0,ArImagen.get(i).length));
            LinearLayout.LayoutParams lp = new LinearLayout.LayoutParams( width: 200,  height: 200);
            iconoVegetal.setLayoutParams(new ViewGroup.LayoutParams(lp));
            iconoVegetal.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    data.putInt("0", finalI);
                    Navigation.findNavController(v).navigate(R.id.vegetalIndividual, data);
                }
            });
            linLay.addView(iconoVegetal);
        }
    }
```

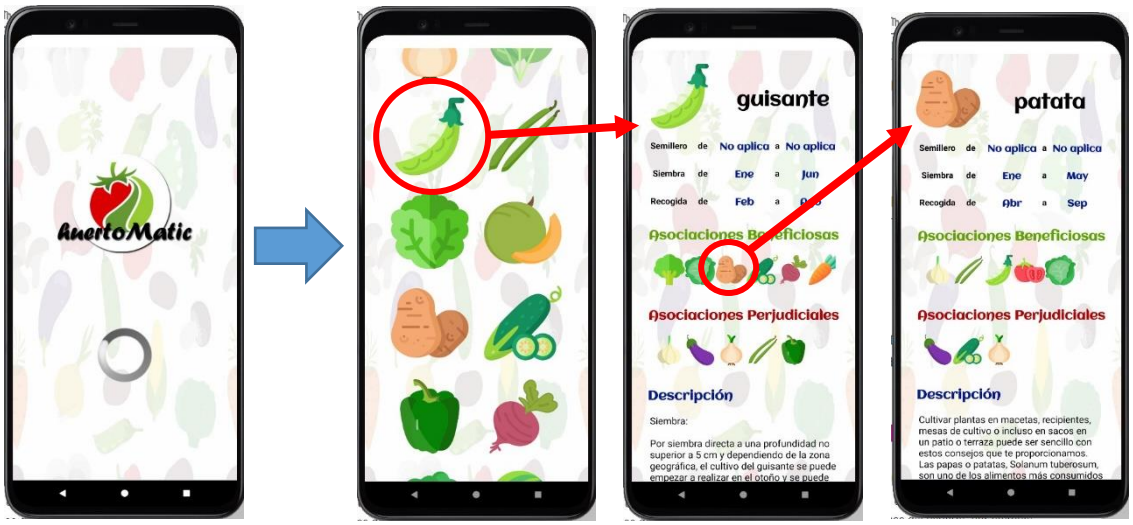
```java
/**
 * Método para Obtener todos los datos del vegetal individual y mostrarlos, con excepción de las asociaciones.
 */
public void Buscar() {
    AdminSQLiteOpenHelper administrador = new AdminSQLiteOpenHelper(getContext(),  name: "huerto.db",  factory: null,  version: 4);
    SQLiteDatabase BaseDeDatos = administrador.getWritableDatabase();

    mesesNombre = new ArrayList <String> ();
    mesesNombre.add("No aplica");
    mesesNombre.add("Ene");
    mesesNombre.add("Feb");
    mesesNombre.add("Mar");
    mesesNombre.add("Abr");
    mesesNombre.add("May");
    mesesNombre.add("Jun");
    mesesNombre.add("Jul");
    mesesNombre.add("Ago");
    mesesNombre.add("Sep");
    mesesNombre.add("Oct");
    mesesNombre.add("Nov");
    mesesNombre.add("Dic");

    if (valorBundle != 0) {
        Cursor fila = BaseDeDatos.rawQuery( sql: "SELECT * FROM vegetales WHERE codigo=" + valorBundle,  selectionArgs: null);
        if (fila.moveToFirst()) {
            for (int j = 0; j < mesesNombre.size(); j++) {
                tvMesIniSemillero.setText(mesesNombre.get(fila.getInt( columnIndex: 3)));
                tvMesFinSemillero.setText(mesesNombre.get(fila.getInt( columnIndex: 4)));
                tvMesIniSiembra.setText(mesesNombre.get(fila.getInt( columnIndex: 5)));
                tvMesFinSiembra.setText(mesesNombre.get(fila.getInt( columnIndex: 6)));
                tvMesIniRecogida.setText(mesesNombre.get(fila.getInt( columnIndex: 7)));
                tvMesFinRecogida.setText(mesesNombre.get(fila.getInt( columnIndex: 8)));
            }
            tvNombreVegetal.setText(fila.getString( columnIndex: 1));
            tvDescripcion.setText(fila.getString( columnIndex: 9));
            byte[] imagen = fila.getBlob( columnIndex: 12);
            imageV.setImageBitmap(BitmapFactory.decodeByteArray(imagen,  offset: 0, imagen.length));
        }else{
            Toast.makeText(getContext(),  text: "El artículo no existe", Toast.LENGTH_SHORT).show();
        }
    }else{
        Toast.makeText(getContext(),  text: "Escriba el código del artículo", Toast.LENGTH_SHORT).show();
    }
    BaseDeDatos.close();
}
```

6.2. Estudio de la interacción del usuario con la aplicación

La aplicación estás ideada de una forma sencilla, intuitiva y que no queda lugar a dudas donde te va a dirigir.



6.3. Guía de estilo

Fuentes:

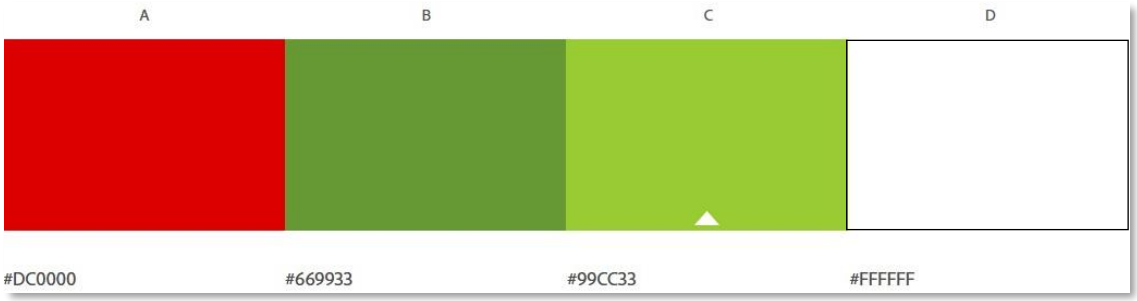
Títulos: font “Aclonica” descargada de Google Fonts

Lorem ipsum dolor sit amet, consectetuer

Texto genérico: predeterminada de Android Studio

Paleta de colores:

- En el logo



- En el texto (Títulos y texto a resaltar)

7. Codificación

7.1. Lenguajes, herramientas y metodologías empleadas

La aplicación ha sido desarrollada por completo en Android Studio native con Java, sin ningún tipo de framework.

Para la base de datos se ha utilizado Sqlite3, ya que, aunque no tiene mucha potencia, es suficiente para manejar datos sin tener que estar conectados a una API ni siquiera a tener una conexión a internet, ya que el archivo .db se encuentra dentro del mismo dispositivo.

Se ha tomado esta decisión porque no se requieren continuas actualizaciones de la base de datos y en las zonas donde se encuentran los terrenos a cultivar, puede que no haya cobertura.

Para generar la base de datos se ha utilizado la aplicación “DB Browser (SQLite)” en la cual podemos crear una base de datos con todo lo necesario dentro de lo que permite SQlite, para posteriormente importarla a la aplicación.

7.2. Aspectos destacables de la implementación

En la implementación, lo más complicado ha sido el tratamiento de los datos, recuperarlos de la base de datos en un tipo, y localizar mediante ese dato el objeto que queremos utilizar.

8. Evaluación y pruebas

- Problema:

  En la integración de la base de datos con el programa, surgió el problema de como modificar una base de datos que ya estaba creada en la app.

- Solución:

  La base de datos al montarla de manera externa a la app, había que importarla al proyecto en una carpeta de assets, donde mediante lógica, recuperamos e intercambiamos por la ya existente, a modo de Actualización, con lo cual para una actualización de contenido en la aplicación no hace falta nada más que modificar la base de datos.

- Problema:

En el componente de Biblioteca, en un principio se desarrolló de forma estática posicionando las imágenes de una manera concreta y aplicando de forma manual los márgenes entre los iconos. El gran problema de hacerlo así es que, si querías insertar una imagen en medio de las demás, tenías que reorganizar todo el Layout y aunque quisieras insertar un nuevo vegetal al final, deberías crear el componente a mano.

- Solución:

Recoger las imágenes de la base de datos y gestionarlo de manera dinámica, así la única parte que habría que modificar es la base de datos añadiendo un registro nuevo.

- Problema:

Al crear un Scroll View y posicionar los iconos dentro, no se le podían pasar parámetros para adaptarlo en varias columnas.

- Solución:

Al crear un Scroll View, hay que insertarle y Layout, que por defecto es Linear Layout, si modificamos la opción e integramos un Grid Layout únicamente con añadirle un parámetro en el XML, conseguiremos tantas columnas como queramos o necesitemos.

Los parámetros de posicionamiento los creamos en la lógica.

- Problema:

Qué tamaño elegir tanto de iconos, como de texto.

- Solución:

Escogemos a un grupo reducido de personas con habilidades visuales mermadas por el paso del tiempo y testeamos la aplicación, recibiendo el feedback y tomando notas sobre las modificaciones a realizar gestionando las mismas para que el diseño inicial y la estructura no se viera muy afectada, si no era necesario.

9. Manual de usuario

9.1. Descripción de la aplicación

HuertoMatic, es una sencilla App que consiste en darte unas pautas básicas sobre como poder gestionar tu huerto urbano, o tus plantaciones en el balcón de tu casa.

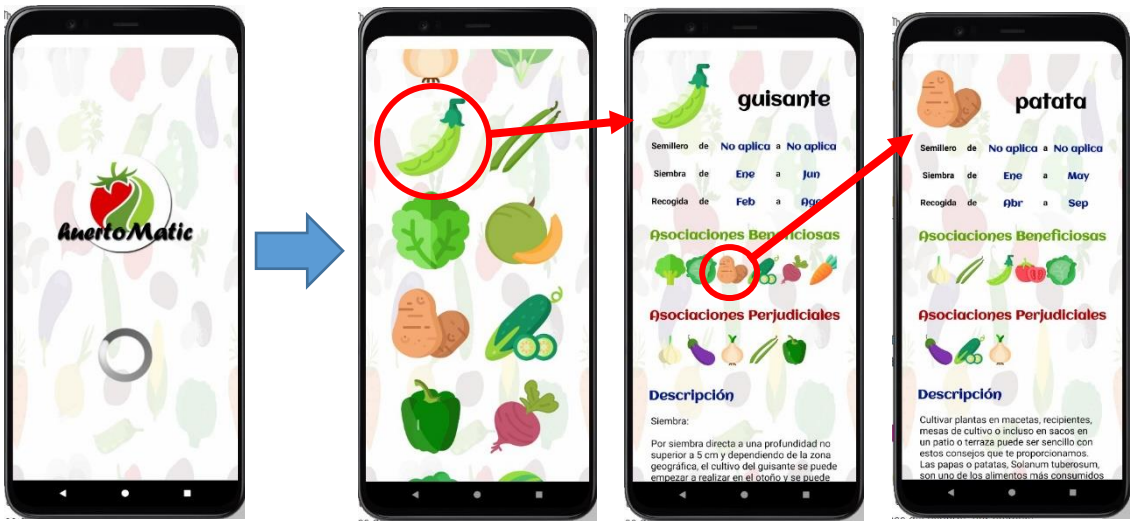
Te facilita conocimientos sobre cuando plantar, tanto en semillero como en plantón o cuando es la fecha óptima de recogida.

Su interfaz gráfica sencilla e intuitiva la hace una muy buena opción para un público de todas las edades.

Su funcionamiento es tan sencillo como pulsar el icono de lo que quieres plantar.

9.2. Funcionalidad y características

La funcionalidad es como se ha explicado en el apartado anterior, claramente intuitiva, donde tenemos un conjunto de vegetales que puedes recorrer en un scroll vertical (imagen 2).



1 2 3 4

Al pulsar el vegetal del cual quieres ver las características, este te lleva a los detalles (imagen 3).

Si observamos la imagen numero 3, al pulsar en alguna de las asociaciones, tanto beneficiosas como perjudiciales, te lleva directamente a la descripción del vegetal pulsado (imagen 4).

10. Conclusiones

Como conclusión podemos sacar que ha sido interesante trabajar con java en Android studio, aunque la tendencia es utilizar Kotlin como lenguaje de programación.

A la hora de buscar información sobre contenido, la gran mayoría están programadas en este último lenguaje, aunque Java sigue luchando por un segundo puesto bastante digno.

En cuanto a la aplicación, su sencillo diseño de front end no debe engañar a nadie sobre el back end que conlleva modificar cualquier funcionalidad.

Este apartado se hubiera solucionado si se hubiera dedicado más tiempo a la planificación del proyecto en cuanto a funcionalidades y componentes futuros, aunque se han dejado abiertas muchas posibilidades de continuar con nuevas opciones.

Con respecto a las metodologías ágiles, no interesan gestionarlas para proyectos de poca envergadura como este, ya que la documentación y la planificación a desarrollar es tan extensa o más que el proyecto en sí. Motivo por el que se debe utilizar una metodología clásica de cascada, y repito: únicamente en proyectos pequeños.

11. Bibliografía y documentación

11.1. Webs expertas en la materia

- https://www.frutas-hortalizas.com
- https://www.ecoagricultor.com

11.2. Android:

- https://www.youtube.com/watch?v=huxvGwRtt5Q (como utilizar ScrollViews)
- http://www.flipandroid.com/simple-exportacin-e-importacin-de-una-base-de-datos-sqlite-en-android.html (importar y exportar BBDD SQLite Android)
- http://www.aprendeandroid.com/l5/sql4.htm (creación de una BBDD externa e importarla a Android)