Instituto Politécnico Nacional

ESCUELA SUPERIOR DE CÓMPUTO

Trabajo Terminal

2019-A009

Ingeniería en Sistemas Computacionales

PROTOTIPO DE SISTEMA PARA RECONOCER TEXTO EN IMÁGENES Y TRADUCIRLO A LATEX

PRESENTAN:

Carlos Tonatihu Barrera Pérez
Juan Carlos Garcia Medina
Ian Mendoza Jaimes

DIRECTORES:
Dr. Jorge Cortés Galicia

Ciudad de México, 11 de mayo de 2020

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

LaTeX es un sistema de tipografía de alta calidad que incluye características útiles para el diseño de documentos técnicos y científicos. Este software, es de facto un estándar para la comunicación y la publicación de artículos científicos.

A pesar de lo práctico que puede ser LaTeX, si el documento contiene muchas expresiones matemáticas, puede resultar tedioso para el usuario escribir dichas expresiones. Por lo que este Trabajo Terminal tiene como finalidad brindar una herramienta que permita amenizar la interacción de nuevos usuarios al escribir expresiones matemáticas en LaTeX mediante el uso de una aplicación móvil y una aplicación web, esta última capaz de reconocer expresiones matemáticas en imágenes que pueden ser tomadas desde el mismo `smartphone`.

1.1. Planteamiento del problema

En la actualidad no existe un sistema que permita reconocer expresiones matemáticas tomando como entrada una fotografía para su posterior traducción a LaTeX, si bien en los últimos años la investigación ha permitido que haya avances en cuanto al reconocimiento y traducción de expresiones matemáticas, al tener un enfoque de investigación se quedan como modelos fuera de lo práctico y los pocos existentes como se muestra en el estado del arte **??** trabajan con entradas ideales como son expresiones escritas desde dispositivos tales como tabletas digitalizadoras o plumas electrónicas sin considerar entradas como pueden ser fotografías tomadas por un smartphone en las que la resolución varía de dispositivo a dispositivo eliminando un tamaño fijo de la entrada y otros elementos como el ruido que puede incorporarse al momento de la toma de la fotografía que afectan el reconocimiento de las expresiones matemáticas. Además, no cuentan con un control sobre dichas traducciones que se realizan o sobre los usuarios.

El enfoque del presente trabajo terminal consiste en desarrollar un sistema conformado por una aplicación tanto móvil como web que en conjunto permitan tomar fotografías y extraer las expresiones matemáticas para su posterior traducción a LaTeX incluyendo la capacidad de gestionar las traducciones incluyendo la clasificación de las mismas mediante usuarios y proyectos asociados a dichas traducciones.

1.2. Objetivos

1.2.1. Objetivo general

Desarrollar un sistema que reconozca un conjunto delimitado de tipos de expresiones matemáticas en una imagen dada y las traduzca a un formato que un compilador de LaTeX pueda procesar.

1.2.2. Objetivos específicos

1. Desarrollar una aplicación móvil que permita tomar fotografías y que pueda conectarse a una aplicación web para su posterior procesamiento.
2. Desarrollar un módulo de análisis de imágenes para el reconocimiento de las expresiones matemáticas.
3. Desarrollar un módulo de traducción a LaTeX.
4. Desarrollar la interfaz que conecte el módulo de análisis de imágenes alojado en el servidor con la aplicación móvil y web.

1.3. Resultados esperados

Podemos separar el sistema en dos bloques principales, el primero el lado del cliente el cual se compondrá de la aplicación móvil y de la interfaz web con las cuales el usuario podrá interactuar. Por otro lado, se tiene la parte del servidor web que se conectará con la aplicación móvil, en el servidor se encontrará la parte de análisis de imágenes junto con el módulo de traducción. Finalmente se tiene el módulo de gestión de usuarios. Esto se puede apreciar en la Figura 1.1.

El sistema se compone de cinco módulos:

1. Módulo de análisis de imágenes. Este módulo se encargará de procesar las imágenes e identificar un conjunto delimitado de tipos de expresiones matemáticas.
2. Módulo de traducción. Este módulo tomará como entrada lo obtenido en la etapa de análisis de imágenes y regresará el respectivo código de LaTeX que represente las expresiones matemáticas reconocidas.
3. Módulo de gestión de usuarios. En este módulo se hará la gestión de la información de los usuarios de la aplicación, lo que implica tener un control de su información y de los archivos que generen al usar el sistema. Esta gestión de usuarios estará presente tanto en la aplicación móvil como en el servidor web.

4. Aplicación móvil. Este módulo consiste en el desarrollo de la aplicación móvil que el usuario final llevará en su Smartphone y con la cual podrá tomar fotos que cumplan ciertos requisitos para así obtener el código en LaTeX a través de la comunicación con la aplicación web.

5. Servidor web. El servidor web hará uso de los módulos de análisis de imágenes, de traducción y de gestión de usuarios por lo que deberá de mantener una comunicación entre ellos y la aplicación móvil, así como con la base de datos. Además, es el punto que permite visualizar el resultado final de todo el procesamiento de imágenes a través de una interfaz web.

Los productos esperados del Trabajo Terminal son:

6. Código fuente del trabajo desarrollado (todos los módulos).

7. Documentación técnica del sistema.

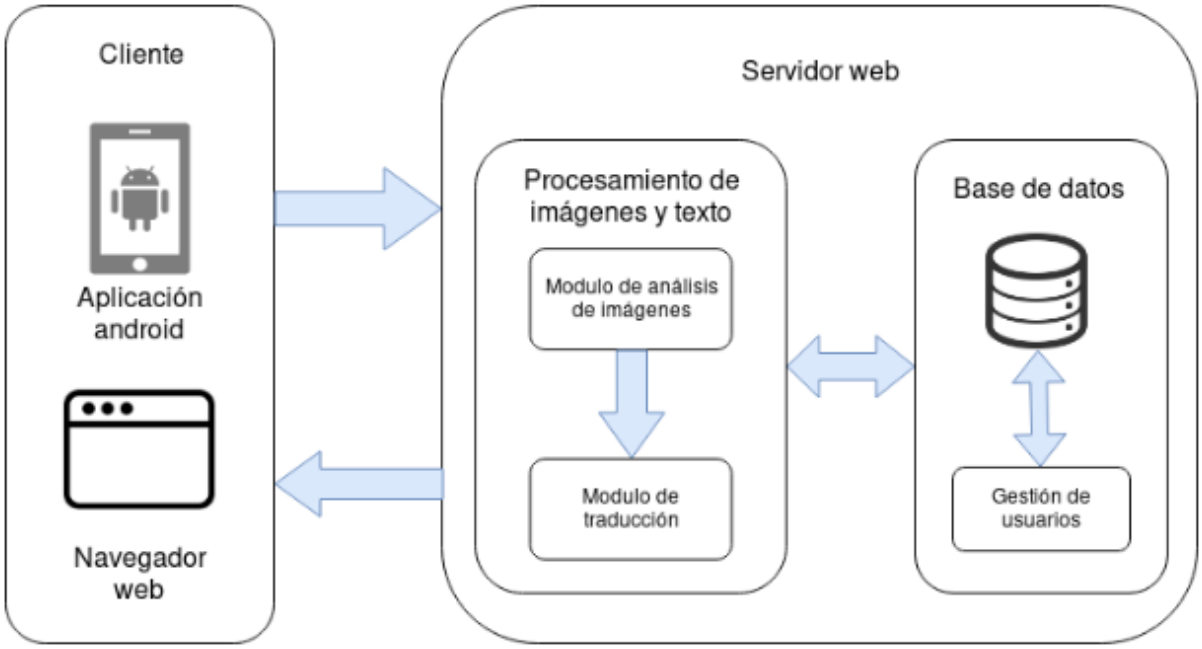

Figura 1.1: Arquitectura del sistema.

1.4. Metodología

Para el desarrollo del proyecto se eligió la metodología incremental debido a que permite dividir el proyecto en pequeños incrementos [6]. Este modelo nos permite flexibilidad a la hora de decidir que requerimientos funcionales se trabajaran primero, es decir, primero se trabajan aquellos con mayor prioridad en los primeros incrementos del sistema para después trabajar el resto o realizar un análisis más profundo a dichos requerimientos de incrementos futuros [7]. De esta forma, se plantean los siguientes incrementos con el objetivo de que al final de cada uno de estos se obtenga la funcionalidad requerida y validar que lo que se trabajo sea correcto para que al final de todos los incrementos se tenga el sistema completo.

- Se desarrollará la aplicación móvil la cual su principal funcionalidad es la obtención de las imágenes y su comunicación con el servidor web.
- Se desarrollará la aplicación web ya que es lo que nos servirá para conectar con la aplicación móvil con el resto de los módulos que se desarrollaran en un futuro.
- Se hará la integración entre la aplicación móvil y la aplicación web.
- Se detallaran a fondo requerimientos si es necesario, se realizara el análisis y diseño del módulo de análisis de imágenes, construcción del módulo y finalmente se harán pruebas sobre este.
- El módulo de traducción va de la mano con el módulo anterior, al tener esto listo y con sus respectivas validaciones aprobadas se realizará el desarrollo y con ello realizar pruebas sobre todos los componentes que se tengan hasta este punto.
- Este incremento final tendrá como objetivo el integrar todo lo desarrollado hasta este punto.

Capítulo 2

Desarrollo del sistema

A continuación, se explica el desarrollo del sistema, dicha explicación se encuentra dividida en la sección del desarrollo de la aplicación móvil y por otra parte se encuentra el desarrollo de la aplicación web.

2.1. Android

Esta sección tiene objetivo presentar las principales características en el desarrollo de la aplicación para Android.

2.1.1. Arquitectura de la aplicación

Para el desarrollo de la aplicación se implemento la arquitectura Clean, la cual como ya se ha mencionado antes se ha mencionado se ha vuelto muy popular en el desarrollo de aplicaciones móviles para android debido a que es una solución que produce sistemas que presentan las siguientes características.

- Escalables, por lo que se pueden agregar más funcionalidades de forma sencilla.
- Presentan modularidad.
- Presentan independencia en cuanto a frameworks, interfaz de usuario y bases de datos.
- El proyecto es más fácil de mantener por lo que es más sencillo hacer cambios.

Al utilizar esta arquitectura el proyecto queda separado en tres capas como se observa en la figura 2.1 con lo cual cada una de ellas tiene su propósito definido.

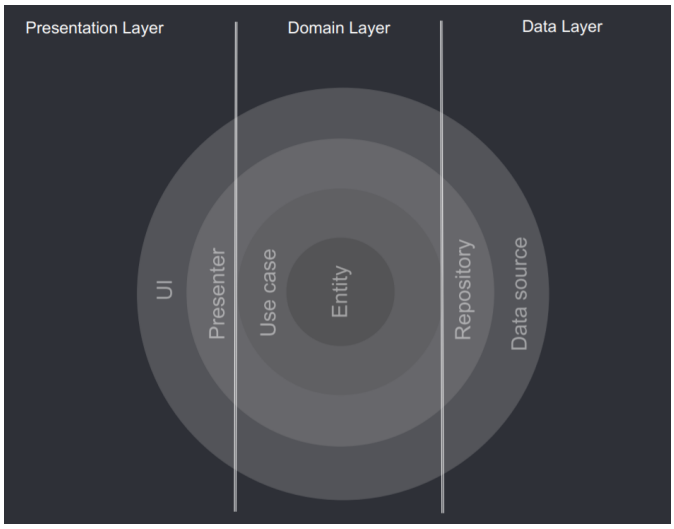

Figura 2.1: Tres capas que se tienen al utilizar la arquitectura Clean [3]

2.1.1.1. Capa de datos

La información que se utiliza en el resto de capas proviene de esta capa. Esta capa a su vez se encuentra dividida en la capa de repositorio y en la capa de fuente de datos.

Capa de repositorio En esta capa se utiliza el patrón de repositorio como se muestra en la figura 2.2. Gracias a este patrón se puede tener acceso a diferentes fuentes de datos que se encuentran en la capa más baja de nuestra arquitectura, esto nos permite un acceso a los datos de forma transparente para el usuario bajo las condiciones que se presenten.

La forma de utilizar este patrón en la aplicación desarrollado crear una clase en la cual se hace uso de la interfaz que se tiene para el acceso a la fuente de datos. En el siguiente código se puede apreciar el como se crea una instancia de APIService que es nuestra interfaz para fuente de datos.

Después, en nuestro método findAllProyecttosByUser se recupera la información necesaria para mandarla a las capas superiores.

```
1 public class ProjectRepositoryImpl implements ProjectRepository{
2     private APIService service = ServiceGenerator.createService(APIService.
    class);
3     private static final String TAG = ProjectRepositoryImpl.class.
    getCanonicalName();
4
5     @Override
6     public MutableLiveData<BusinessResult<ProyectoModel>>
    findAllProyectosByUser(Integer id, String key) {
```

```
7       MutableLiveData<BusinessResult<ProyectoModel>>
    proyectoDataMutableLiveData = new MutableLiveData<>();
8
9       try {
10          service.getProyectosByUsuario(id, key).enqueue(new Callback<List<
    ProyectoData>>() {
11              @Override
12              public void onResponse(Call<List<ProyectoData>> call,
    Response<List<ProyectoData>> response) {
13                  BusinessResult<ProyectoModel> model = new BusinessResult
    <>();
14                  List<ProyectoModel> modelos = new ArrayList<>();
15                  if (response.isSuccessful()) {
16                      for (ProyectoData data : response.body()) {
17                          ProyectoModel proyectoModel = new ProyectoModel()
    ;
18                          proyectoModel.setRate(data.getCalificacion());
19                          proyectoModel.setId(data.getId());
20                          proyectoModel.setName(data.getNombre());
21                          proyectoModel.setTextDate(data.getFecha());
22                          modelos.add(proyectoModel);
23                      }
24                      model.setResults(modelos);
25                      model.setCode(ResultCodes.SUCCESS);
26                  }
27                  proyectoDataMutableLiveData.setValue(model);
28              }
29              @Override
30              public void onFailure(Call<List<ProyectoData>> call,
    Throwable t) {
31                  BusinessResult<ProyectoModel> model = new BusinessResult
    <>();
32                  proyectoDataMutableLiveData.setValue(model);
33              }
34          });
```

```
35          } catch (NetworkOnMainThreadException e) {
36              Log.e(TAG, "findAllProyectosByUser ", e);
37          }
38          return proyectoDataMutableLiveData;
39      }
40  }
```

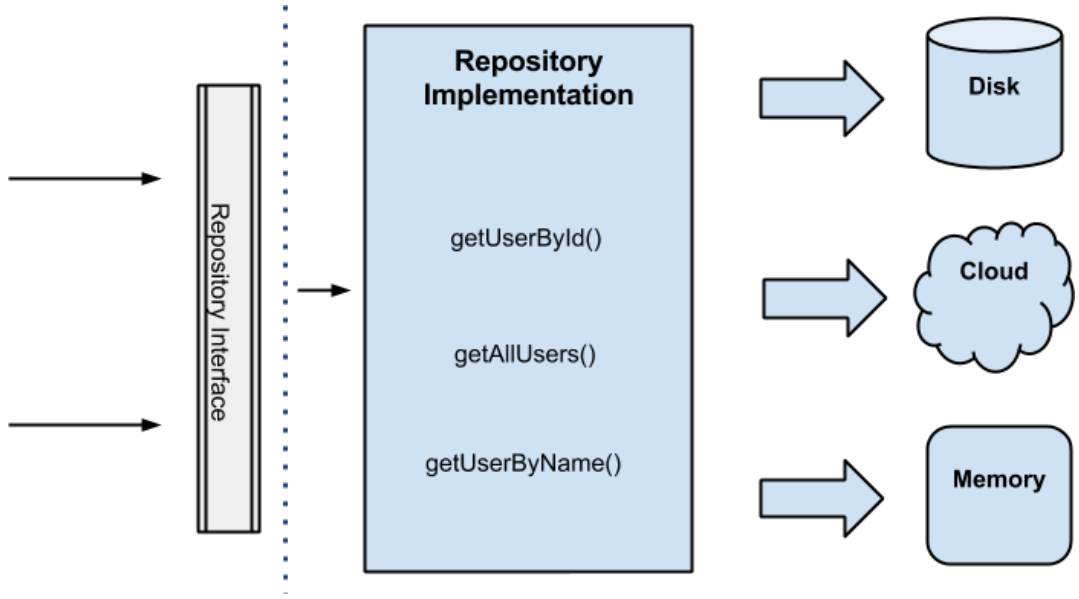

Figura 2.2: Capa de datos [4]

Capa de fuente de datos En este trabajo, la fuente de datos que se tiene es un API REST, sin embargo si se requiere acceder a información que se persista en el teléfono se pude agregar otra fuente de datos. Se utiliza retrofit para poder realizar la comunicación con el API REST.

La forma de utilizar retrofit es crear una interfaz con todos los métodos para recuperar o enviar información al API REST, en esta interfaz cada método tiene la URL a la cual se realizara la petición con alguno de los métodos que tiene HTTP, se tienen los parámetro que se envían y cada método nos regresa una llamada asincrona que se trabaja en la capa de repositorio. Esto se puede apreciar en el siguiente código.

```
1  public interface APIService {
2      // Crear proyectoModel
3      @POST("/proyectos")
4      Call<ProyectoData> createProyecto(@Body ProyectoData proyectoModel,
      @Header("Authorization") String token);
```

```
5     // Edicion de un proyectoModel
6     @PUT("/proyectos/{idProyecto}")
7     Call<ProyectoData> editProyecto(@Path("idProyecto") Integer idProyecto,
      @Body ProyectoData proyectoModel, @Header("Authorization") String token);
8     // Elimina un proyecto
9     @DELETE("/proyectos/{idProyecto}")
10    Call<ProyectoData> deleteProyecto(@Path("idProyecto") Integer idProyecto,
      @Header("Authorization") String key);
11    // Obtiene las traducciones asociadas a un proyecto
12    @GET("/proyectos/{idProyecto}/traducciones")
13    Call<List<TraduccionData>> getTraduccionesByProyecto(@Path("idProyecto")
      Integer idProyecto, @Header("Authorization") String key);
14    // Creacion de una traduccionModel
15    @POST("/traducciones")
16    Call<TraduccionData> createTraduccion(@Body TraduccionData traduccionData
      );
17    // Edicion de un traduccionModel
18    @PUT("/traducciones/{idTraducion}")
19    Call<TraduccionData> editTraduccion(@Path("idTraducion") Integer
      idTraducion, @Body TraduccionData traduccionData);
20    // Elimina una traduccion
21    @DELETE("/traducciones/{idTraducion}")
22    Call<TraduccionData> deleteTraduccion(@Path("idTraducion") Integer
      idTraducion, @Header("Authorization") String token);
23    // Manda a crear un usuarioData
24    @POST("/usuarios")
25    Call<UsuarioData> createUsuario(@Body UsuarioData usuarioData);
26    // Para hacer login
27    @POST("/users/login")
28    Call<UsuarioData> loginUsuario(@Body UsuarioData usuarioData);
29    // Para hacer la recuperacion de contra
30    @POST("/usuarios/recuperar")
31    Call<UsuarioData> recuperarUsuario(@Body UsuarioData usuarioData);
32    // Para editar usuarioData
33    @PUT("/users/{idUsuario}")
```

```
34      Call<UsuarioData> editUsuario(@Path("idUsuario") Integer id, @Body
        UsuarioData usuarioData, @Header("Authorization") String key);
35      // Obtiene los proyectos asociados a un usuario
36      @GET("/usuarios/{idUsuario}/proyectos")
37      Call<List<ProyectoData>> getProyectosByUsuario(@Path("idUsuario") Integer
        idUsuario, @Header("Authorization") String key);
38  }
```

Para poder hacer uso de esta interfaz se tiene que configurar bajo ciertas características especificas como lo son la URL a la cual hará peticiones, el logger que se utilizara para poder observar las peticiones que se realizan y brindar una retroalimentación a la hora de hacer pruebas y por ultimo el parser que se utilizara para trabajar y pasar de clases a datos que el API REST entienda y pueda utilizar, en este caso se utilizo el formato JSON. La definición de estas características se tiene en el siguiente código.

```
1  public class ServiceGenerator {
2      private static final String BASE_URL = "http://10.100.72.207:8000/";
3      private static Retrofit.Builder builder = new Retrofit.Builder().baseUrl(
       BASE_URL)
4              .addConverterFactory(GsonConverterFactory.create());
5
6      private static Retrofit retrofit = builder.build();
7      private static OkHttpClient.Builder httpClient = new OkHttpClient.Builder
       ();
8      private static HttpLoggingInterceptor loggingInterceptor = new
       HttpLoggingInterceptor();
9
10     public static <S> S createService(Class<S> serviceClass) {
11         if (!httpClient.interceptors().contains(loggingInterceptor)) {
12             loggingInterceptor.level(HttpLoggingInterceptor.Level.BODY);
13             httpClient.addInterceptor(loggingInterceptor);
14             builder.client(httpClient.build());
15             retrofit = builder.build();
16         }
17         return retrofit.create(serviceClass);
18     }
19 }
```

Finalmente, en esta capa se tienen clases Java que después se mapean a objetos JSON y viceversa, para realizar esto se crea un POJO con los atributos que se necesitan además de agregar anotaciones de retrofit para que el parser pude hacer la conversión necesaria. Un ejemplo de esto es en la siguiente clase de java.

```
1  public class UsuarioData {
2      @SerializedName("id")
3      private Integer id;
4      @SerializedName("nombre")
5      private String nombre;
6      @SerializedName("apellido")
7      private String apellidos;
8      @SerializedName("username")
9      private String email;
10     @SerializedName("password")
11     private String password;
12     @SerializedName("responseCode")
13     private Integer responseCode;
14     @SerializedName("keyAuth")
15     private String keyAuth;
16     @SerializedName("currentPassword")
17     private String currentPassword;
18
19     public String getEmail() {
20         return email;
21     }
22     public void setEmail(String email) {
23         this.email = email;
24     }
25     public String getPassword() {
26         return password;
27     }
28     public void setPassword(String password) {
29         this.password = password;
30     }
31     public Integer getId() {
```

```
32          return id;
33      }
34      public void setId(Integer id) {
35          this.id = id;
36      }
37      public String getNombre() {
38          return nombre;
39      }
40      public void setNombre(String nombre) {
41          this.nombre = nombre;
42      }
43      public String getApellidos() {
44          return apellidos;
45      }
46      public void setApellidos(String apellidos) {
47          this.apellidos = apellidos;
48      }
49      public Integer getResponseCode() {
50          return responseCode;
51      }
52      public void setResponseCode(Integer responseCode) {
53          this.responseCode = responseCode;
54      }
55      public String getKeyAuth() {
56          return keyAuth;
57      }
58      public void setKeyAuth(String keyAuth) {
59          this.keyAuth = keyAuth;
60      }
61      public String getCurrentPassword() {
62          return currentPassword;
63      }
64      public void setCurrentPassword(String currentPassword) {
65          this.currentPassword = currentPassword;
66      }
```

```
67 }
```

2.1.1.2. Capa de dominio

En esta capa es la intermediaria entre las otras dos capas que se tienen, es donde se encuentran los casos de uso también conocidos como interactors como se muestra en la figura 2.3 en ellos la lógica del negocio es ejecutada es por esto que es el núcleo de la aplicación.

Es importante mencionar que esta capa, al ser la encargada del negocio es donde se hacen validaciones en la información y dicha información se adapta para que sea trabajada en la capa de presentación o en la de datos

Además de contener los casos de uso en esta capa se encuentran las entidades y se hace uso de los repositorios para acceder a la información proporcionada por la capa de datos.

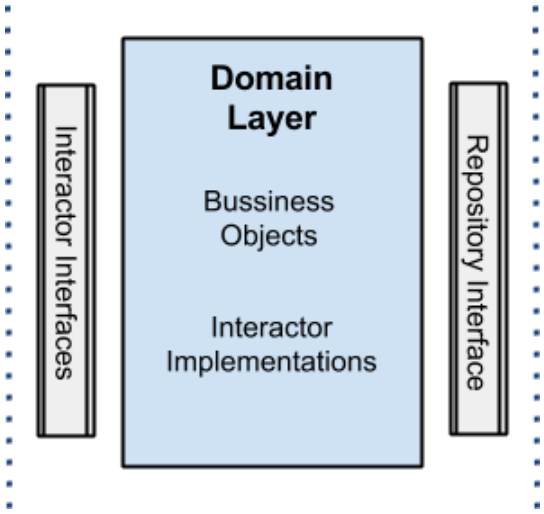

Figura 2.3: Capa de dominio [4]

Para tener un control sobre posibles errores en la capa de presentación o en la capa de datos se utilizan códigos de resultados al igual que una clase que contiene el resultado que se puede presentar, así como la información que se le regresa a la capa de presentación. Se hace uso de genéricos para poder reutilizar esta clase en toda la aplicación y no duplicar código. La clase es la siguiente.

```
1 public class BusinessResult<T> {
2     private Integer code = ResultCodes.ERROR;
3     private T result;
4     private List<T> results;
5
6     public Integer getCode() {
```

```
7           return code;
8       }
9
10      public void setCode(Integer code) {
11          this.code = code;
12      }
13
14      public T getResult() {
15          return result;
16      }
17
18      public void setResult(T result) {
19          this.result = result;
20      }
21
22      public List<T> getResults() {
23          return results;
24      }
25
26      public void setResults(List<T> results) {
27          this.results = results;
28      }
29  }
```

La forma en la que se utiliza esta clase en un caso de uso se presenta en el siguiente código que permite iniciar sesión.

```
1  public class UserInteractorImpl implements UserInteractor {
2      public static final String TAG = UserInteractorImpl.class.
       getCanonicalName();
3      private UserRepository repository;
4
5      public UserInteractorImpl() {
6          repository = new UserRepositoryImpl();
7      }
8
9      @Override
```

```
10      public MutableLiveData<BusinessResult<UsuarioModel>> logIn(UsuarioModel
    usuarioModel) {
11          BusinessResult<UsuarioModel> resultado = new BusinessResult<>();
12          MutableLiveData<BusinessResult<UsuarioModel>> mutableLiveData = new
    MutableLiveData<>();
13          usuarioModel.setValidPassword(RN002.isPasswordValid(usuarioModel.
    getPassword()));
14          usuarioModel.setValidEmail(RN002.isEmailValid(usuarioModel.getEmail()
    ));
15          if (usuarioModel.getValidEmail() && usuarioModel.getValidPassword())
    {
16              UsuarioData usuarioData = new UsuarioData();
17              usuarioData.setEmail(usuarioModel.getEmail());
18              usuarioData.setPassword(usuarioModel.getPassword());
19              mutableLiveData = repository.login(usuarioData);
20          } else {
21              resultado.setCode(ResultCodes.RN002);
22              resultado.setResult(usuarioModel);
23              mutableLiveData.setValue(resultado);
24          }
25
26          return mutableLiveData;
27      }
28  }
```

A su vez el caso de uso utiliza sus propios clases de java para presentar información al usuario en la capa de presentación así como controlar posibles errores en la información que ingrese el usuario los campos de los formularios, un ejemplo de este tipo de clases es el siguiente.

```
1  public class UsuarioModel {
2      private Integer id;
3      private String email;
4      private String password;
5      private String keyAuth;
6      private String name;
7      private String secondPassword;
8      private String lastname;
```

```
9     private String currentPassword;
10    private Boolean validLastName = false;
11    private Boolean validName = false;
12    private Boolean validSecondPassword = false;
13    private Boolean validEmail = false;
14    private Boolean validPassword = false;
15    private Boolean validCurrentPassword = false;
16
17    public String getEmail() {
18        return email;
19    }
20
21    public void setEmail(String email) {
22        this.email = email;
23    }
24
25    public String getPassword() {
26        return password;
27    }
28
29    public void setPassword(String password) {
30        this.password = password;
31    }
32
33    public Boolean getValidEmail() {
34        return validEmail;
35    }
36
37    public void setValidEmail(Boolean validEmail) {
38        this.validEmail = validEmail;
39    }
40
41    public Boolean getValidPassword() {
42        return validPassword;
43    }
```

```
44
45      public void setValidPassword(Boolean validPassword) {
46          this.validPassword = validPassword;
47      }
48
49      public Integer getId() {
50          return id;
51      }
52
53      public void setId(Integer id) {
54          this.id = id;
55      }
56
57      public String getKeyAuth() {
58          return keyAuth;
59      }
60
61      public void setKeyAuth(String keyAuth) {
62          this.keyAuth = keyAuth;
63      }
64
65      public String getName() {
66          return name;
67      }
68
69      public void setName(String name) {
70          this.name = name;
71      }
72
73      public String getSecondPassword() {
74          return secondPassword;
75      }
76
77      public void setSecondPassword(String secondPassword) {
78          this.secondPassword = secondPassword;
```

```
79      }
80
81      public String getLastname() {
82          return lastname;
83      }
84
85      public void setLastname(String lastname) {
86          this.lastname = lastname;
87      }
88
89      public Boolean getValidLastName() {
90          return validLastName;
91      }
92
93      public void setValidLastName(Boolean validLastName) {
94          this.validLastName = validLastName;
95      }
96
97      public Boolean getValidName() {
98          return validName;
99      }
100
101     public void setValidName(Boolean validName) {
102         this.validName = validName;
103     }
104
105     public Boolean getValidSecondPassword() {
106         return validSecondPassword;
107     }
108
109     public void setValidSecondPassword(Boolean validSecondPassword) {
110         this.validSecondPassword = validSecondPassword;
111     }
112
113     public String getCurrentPassword() {
```

```
114         return currentPassword;
115     }
116
117     public void setCurrentPassword(String currentPassword) {
118         this.currentPassword = currentPassword;
119     }
120
121     public Boolean getValidCurrentPassword() {
122         return validCurrentPassword;
123     }
124
125     public void setValidCurrentPassword(Boolean validCurrentPassword) {
126         this.validCurrentPassword = validCurrentPassword;
127     }
128 }
```

2.1.1.3. Capa de presentación

En esta capa como se muestra en la figura 2.4 se trabaja con la lógica relacionada a las interfaces que se tienen en la aplicación, es decir a actividades, fragmentos y archivos XML.

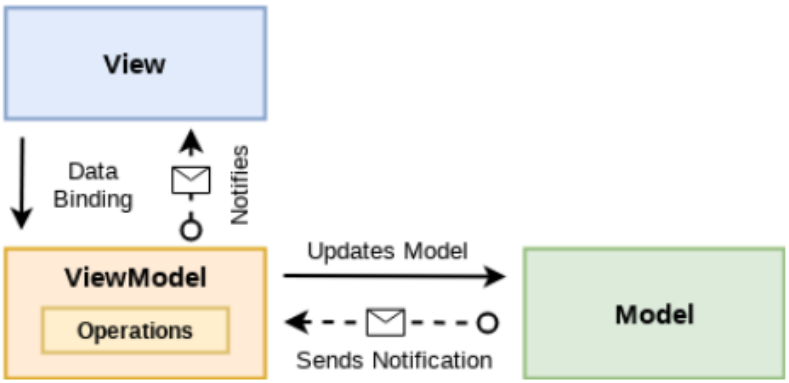

Figura 2.4: Capa de presentación [5]

En esta capa se pueden trabajar con patrones como MVC y MVP pero en este caso se utiliza el patrón MVVM cada uno con una función en particular. [5]

- **Modelo** Se encarga de representar la información que sera presentada en la vista.

- **Vista** Compuesta en este caso por las actividades y fragmentos de la aplicación, su tarea es mostrar la información, hacen uso de los viewmodels para poder realizar cambios en la interfaz.
- **ViewModel** El ViewModel sera el encargado de ejecutar los casos de uso o interactors con el objetivo de actualizar la vista de acuerdo a la información que presente el modelo.

2.2. Web

Esta sección tiene como objetivo presentar las principales características en el desarrollo de la aplicación Web.

2.2.1. Arquitectura de la aplicación

Para el desarrollo de la aplicación se implemento el patrón de diseño Modelo Vista Template (MTV), que como se menciono previamente, es el patrón que Django utiliza.

2.2.1.1. Modelo

En esta capa se maneja todo el acceso a los datos de la aplicación. Django provee de un ORM que permite controlar una base de datos (PostgreSQL para este proyecto). De este modo, todas las tablas que componen la base de datos, están declaradas en esta capa.

2.2.1.2. Vista

En esta capa se maneja la lógica del negocio. Se implementan las validaciones necesarias y se decide que datos deben de ser mostrados al usuario sin indicar como deben de ser presentados a diferencia del tradicional MVC. Esta capa debe de ser vista como un puente entre el Modelo y los Templates.

2.2.1.3. Template

Esta es la capa de presentación. En ella se maneja la forma en la que serán mostrados los datos de la aplicación.

2.3. Conjunto de entrenamiento: CROHME

El conjunto de datos de entrenamiento para el desarrollo de la red es el denominado **CROHME** por sus siglas en inglés: **Competition on Recognition of Online Handwritten Mathematical Expressions** publicado por los organizadores de la competencia internacional CROHME; para el caso del conjunto de entrenamiento fue posible reunir 7169 elementos que de acuerdo a investigación previa [15], son relativamente pocos para esperar una buena precisión, los elementos en en el conjunto son archivos de tipo INKML.

2.3.1. Formato del conjunto de datos

Como ya se mencionó, los elementos del conjunto son archivos de tipo INKML (Ink Markup Language) y que principalmente se compone de tres partes:

- Ink: Un conjunto de trazos definidos por puntos.
- Nivel de símbolo Ground-Truth: La segmentación e información de etiqueta de cada símbolo de la expresión.
- Ground-truth: La estructura MATHML de la expresión.

La información de Ground-Truth tanto de nivel de símbolo como de la expresión matemática fueron ingresadas manualmente por los colaboradores de la generación del conjunto, además, alguna información general es agregada en el archivo:

- Los canales (en este caso X, Y)
- La información del escritor (Identificación, entrega, edad, Mano dominante, género, etc), si está disponible.
- Ground-Truth en LaTeX para fácilmente renderizarlo.

A continuación se muestra un ejemplo de un archivo del dataset que representa la expresión $2 ^ {-1 } $:
2^{-1} renderizado.

```
<ink xmlns="http://www.w3.org/2003/InkML">
<traceFormat>
<channel name="X" type="decimal"/>
<channel name="Y" type="decimal"/>
</traceFormat>
<annotation type="truth">$2^{-1}$</annotation>
```

```
<annotation type="UI">2011_IVC_CIEL_F696_E6</annotation>
<annotation type="copyright">LUNAM/IRCCyN</annotation>
<annotation type="writer">CIEL696</annotation>
<annotationXML type="truth" encoding="Content-MathML">
  <math xmlns='http://www.w3.org/1998/Math/MathML'>
    <msup>
      <mn xml:id="2_1">2</mn>
      <mrow>
        <mo xml:id="-_1">-</mo>
        <mn xml:id="1_1">1</mn>
      </mrow>
    </msup>
  </math>
</annotationXML>
<trace id="0">
3849 2989, 3849 2989, 3847 2979, 3853 2964, 3868 2949, 3875 2949, 3887 2962,
    3887 2974, 3880 2991, 3862 3011, 3822 3042, 3827 3052, 3839 3061, 3898
    3049
</trace>
...
<trace id="2">
4009 2905, 4009 2905, 4027 2900, 4049 2887, 4040 2947, 4040 2958
</trace>
<traceGroup xml:id="3">
  ...
</traceGroup>
</ink>
```

Sin embargo, para el propósito del trabajo terminal, el conjunto de datos no es útil en este formato, por lo que se tenía que transformar la información de los trazos en imágenes.

2.3.2. Conversión a imagen

Con la información de los trazos es posible generar una imagen con los puntos de los trazos en negro y fondo blanco, el primer reto fue identificar las etiquetas que contenían a los trazos, para ello se utilizó la biblioteca de Python xml.etree y una función de terceros para poder utilizar dichos trazos posteriormente:

```
traces_data = []


    tree = ET.parse(inkml_file_abs_path)
    root = tree.getroot()
    doc_namespace = "{http://www.w3.org/2003/InkML}"

    'Stores traces_all with their corresponding id'
    traces_all = [{'id': trace_tag.get('id'),
                   'coords': [[round(float(axis_coord)) if float(axis_coord).is_integer() else round(float(axis_coord) * 10000) \
                               for axis_coord in coord[1:].split(' ')] if coord.startswith(' ') \
                              else [round(float(axis_coord)) if float(axis_coord).is_integer() else round(float(axis_coord) * 10000) \
                               for axis_coord in coord.split(' ')] \
                             for coord in (trace_tag.text).replace('\n', '').split(',')]} \
                             for trace_tag in root.findall(doc_namespace + 'trace')]
```

Una vez obtenidos los trazos y con ayuda de matplotlib fueron separados como puntos x,y y utilizados en la función plot de matplotlib para posteriormente guardarlo como imagen.

```
def inkml2img(input_path, output_path):
    traces = get_traces_data(input_path)
    plt.gca().invert_yaxis()
    plt.gca().set_aspect('equal', adjustable='box')
    plt.axes().get_xaxis().set_visible(False)
    plt.axes().get_yaxis().set_visible(False)
    plt.axes().spines['top'].set_visible(False)
    plt.axes().spines['right'].set_visible(False)
    plt.axes().spines['bottom'].set_visible(False)
    plt.axes().spines['left'].set_visible(False)
    for elem in traces:
        ls = elem['trace_group']
        for subls in ls:
            data = np.array(subls)
            x,y=zip(*data)
```

```
            plt.plot(x,y,linewidth=2,c='black')
    plt.savefig(output_path, bbox_inches='tight', dpi=100)
    plt.gcf().clear()
```

Esto tenía que realizarse por cada uno de los elementos del conjunto de entrenamiento, además de también extraer la expresión matemática encerrada entre las etiquetas <**annotation**></**annotation**> con el atributo **type**, para ello nuevamente se utilizó la biblioteca de python xml.etree para acceder a los nodos del árbol directamente:

```
def inkml2tag(self, inkml_path):
    tree = ET.parse(inkml_path)
    root = tree.getroot()
    prefix = "{http://www.w3.org/2003/InkML}"
    GT_tag = [GT for GT in root.findall(prefix + 'annotation') if GT.attrib
   == {'type': 'truth'}]
    if GT_tag is None or len(GT_tag) == 0:
      return ""
    if GT_tag[0] is None or GT_tag[0].text is None:
      return ""
    return GT_tag[0].text
```

Con estos subscripts fue posible desarrollar una biblioteca que permite acceder a la carpeta con el conjunto de entrenamiento en formato inkml y guardarlos como imagen en otra carpeta junto con un archivo CSV conteniendo la ruta relativa de la imagen y la expresión en LaTeX correspondiente separados por coma.

```
converted_expressions/200924-1312-148.i.png, \left ( { a + 7 } \right )
converted_expressions/200926-1550-27.i.png, \mbox { G }
converted_expressions/TrainData2_0_sub_71.i.png,\frac{\sin B + \sin C}{\cos B
    + \cos C}
converted_expressions/200923-1251-164.i.png, { \mbox { z } } ^ { \phi }
converted_expressions/200923-1553-186.i.png, \sqrt { { \mbox { N } - \mbox {
   P } } }
converted_expressions/formulaire012-equation003.i.png,$4 \pi d^2$
converted_expressions/formulaire012-equation036.i.png,$a_ns_3 + a_{n-1}s_2 +
   a_{n-2}s_1 + 3a_{n-3} = 0$
converted_expressions/79_carlos.i.png,$p = \sqrt{a^2 + b^2 - 2ab \cos A}$
converted_expressions/200923-1553-313.i.png, \left ( \sum { \mbox { S } } \
   right )
```

```
converted_expressions/formulaire010-equation073.i.png,$x^7-x^6-x^4-x^2-1$
```

2.3.3. Generador de secuencia de tokens

El archivo CSV generado con lo descrito anteriormente no es suficiente para cargarlo en TensorFlow, la expresión matemática en LaTeX debía ser expresada como una secuencia numérica, por lo que se necesitaba identificar a cada símbolo con un número único y conformar a la secuencia, de modo que la estructura del archivo CSV pasaría de tener la forma **RUTA_IMAGEN**, **EXPRESION_LATEX** a tener la forma **RUTA_IMAGEN**, **SECUENCIA_NUMÉRICA**, teniendo así una nueva representación del conjunto de entrenamiento conformada por una carpeta con las imágenes y un archivo CSV previamente descrito para poder cargarse en TensorFlow.

Para esto se desarrolló gracias a lex en Python un script para especificar los tokens tomando como base los símbolos especificados en la sección **??**.

```
tokens = [
    #'corta',
    #'larga',
    'alpha',
    'pi',
    'beta',
    'gamma',
    'lambda',
    'sigma',
    'mu',
```

```
  def tokenizeDataset(self, file):

    fileTokenized = open("tokenized_test.csv", "w")

    myMap = dict()
    for line in file:

      self.lexer.input(",".join(line.split(",")[1:]))
      #print(line.split(",")[1])
      #print(repr(line.split(",")[1]))
      token = self.lexer.token()
      arr = []
      while token is not None:
        print(token)
        myMap[token.type]= token.value  #fileMap.write(token.type+","+token.
    value+"\n")
        arr.append(Rules.tokens.index(token.type) + 1 ) #+1 means shift one
    due to 0 is reserved
        token = self.lexer.token()
      fileTokenized.write(line.split(",")[0] + "," + toString(arr) + "\n")
    fileTokenized.close()
```

```
converted_expressions/200924-1312-148.i.png,64 33 41 74 47 58 42 65 34
converted_expressions/200926-1550-27.i.png,41 115 42
converted_expressions/TrainData2_0_sub_71.i.png,12 41 14 100 47 14 99 42 41 16 100 47 16 99 42
converted_expressions/200923-1251-164.i.png,41 41 78 42 42 62 41 9 42
converted_expressions/200923-1553-186.i.png,13 41 41 41 107 42 46 41 109 42 42 42
converted_expressions/formulaire012-equation003.i.png,55 2 79 62 53
converted_expressions/formulaire012-equation036.i.png,74 61 75 95 61 54 47 74 61 41 75 46 52 42 95 61 53 47 74 61 41 75 46 53 42 95 61 52 47 54 74 61 41 75 46 54 42 48 51
converted_expressions/79_carlos.i.png,86 48 13 41 74 62 53 47 77 62 53 46 53 74 77 16 102 42
converted_expressions/200923-1553-313.i.png,64 33 18 41 41 108 42 42 65 34
converted_expressions/formulaire010-equation073.i.png,73 62 58 46 73 62 57 46 73 62 55 46 73 62 53 46 52
```

Es importante destacar que se debe también guardar este mapeo único y no alterar el orden, por lo que de requerir agregar nuevos símbolos al conjunto es necesario hacerlo al final de los ya existentes, ya que la red entrenada devuelve secuencias numéricas que deben mapearse para tener la correspondiente expresión en LaTeX.

Capítulo 3

Pruebas del sistema

En este capítulo se detallan las diversas pruebas realizadas a los módulos desarrollados con el fin de verificar y validar su correcto funcionamiento.

3.1. Pruebas unitarias

3.1.1. Aplicación android

Para realizar las pruebas en android se utilizo la biblioteca JUnit la cual es utilizada para realizar pruebas unitarias en Java.

JUnit se utilizo sobre aquellas clases que se utilizaron para trabajar partes en especifico de la lógica del negocio que no involucraban utilitarias de android, es decir, que solamente utilizan Java.

3.1.1.1. Pruebas sobre la clase DateFormater

La clase DateFormater se utiliza en la aplicación para obtener fechas y presentarlas en un formato correcto en pantalla, así como pasar de una cadena de texto a un objeto de tipo Date, por lo cual las as pruebas realizadas sobre esta clase fueron tres.

1. En la primera se valida que la conversión de una cadena de texto a un objeto de tipo Date se haga correctamente siempre y cuando se trate de una cadena valida.

2. En la segunda prueba se valida que se pueda pasar de un tipo date a una cadena de texto y que el valor que se obtenga sea el mismo.

3. En la ultima prueba se valida que se lance una excepción si el valor a convertir a una cadena de texto no es valido.

El código utilizado en cada una de las pruebas es el siguiente.

```java
package com.equipo.superttapp;

import com.equipo.superttapp.util.DateFormater;

import org.junit.Assert;
import org.junit.Test;

import java.util.Calendar;
import java.util.Date;

public class DateFormaterTest {
    @Test
    public void convertStringToDate_validString_shouldParseToDate() {
        Calendar cal = Calendar.getInstance();
        cal.set(2020, 0, 1, 0, 0, 0);
        cal.setTimeInMillis(1577858400000L);
        Date date = cal.getTime();
        Assert.assertEquals("Probando que se obtenga un objeto date con la fecha en string", date.getTime(), DateFormater.convertStringToDate("01/01/2020").getTime());
    }

    @Test
    public void convertDateToString_validDate_shouldParseToString() {
        Calendar cal = Calendar.getInstance();
        cal.set(2020, 0, 1);
        Date date = cal.getTime();
        Assert.assertEquals("Probando que no se obtenga null despues de realizar un cast", "01/01/2020", DateFormater.convertDateToString(date));
    }

    @Test(expected = NullPointerException.class)
    public void convertDateToString_NoValidDate_shouldThrowException() {
        DateFormater.convertDateToString(null);
    }
```

```
33 }
```

En el código anterior, cada uno de los métodos anotados con @Test es una prueba a ejecutar, la forma en la que se valida que la ejecución sea correcta es con la linea de código correspondiente a Assert.assertEquals, la cual compara dos valores y los compara, en caso de ser iguales la prueba es exitosa en caso contrario la prueba ha fallado.

En la ultima prueba la anotación @Test tiene como parámetro una excepción a esperar, si la excepción se dispara la prueba es correcta, por otro lado, si no se da el caso la prueba ha fallado.

La correcta ejecución de las pruebas se puede apreciar en la figura **??**

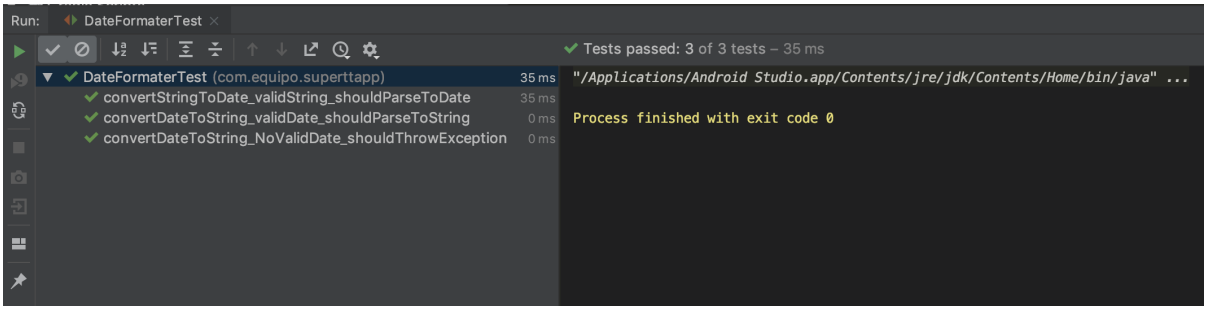

Figura 3.1: Resultados de las pruebas de la clase DateFormater

3.1.1.2. Pruebas sobre la clase RNN002

La clase RNN002 se utiliza en la aplicación para validar el formato de diversos campos que se tienen en los diferentes formularios que incluye la aplicación, es por esto que las pruebas que se realizaron en este caso fueron las siguientes.

1. En la primera prueba se verifica que para un email valido, el resultado sea true.
2. En la siguiente prueba, para un email no valido, el resultado debe de ser false.
3. En la siguiente prueba, para una contraseña valida, el resultado debe de ser true.
4. En la siguiente prueba, para una contraseña no valida, el resultado debe de ser false.
5. En la siguiente prueba, para dos contraseñas validas, el resultado debe de ser true.
6. En la siguiente prueba, para dos contraseñas, una de ellas no valida, el resultado debe de ser false.
7. En la siguiente prueba, para un nombre valido, el resultado debe de ser true.
8. En la siguiente prueba, para un nombre no valido, el resultado debe de ser false.
9. En la siguiente prueba, para un apellido valido, el resultado debe de ser true.

10. En la siguiente prueba, para un apellido no valido, el resultado debe de ser false.
11. En la siguiente prueba, para un nombre de proyecto valido, el resultado debe de ser true.
12. En la ultima prueba, para un nombre de proyecto no valido, el resultado debe de ser false.

El código utilizado en cada una de las pruebas es el siguiente.

```
1  package com.equipo.superttapp;
2
3  import com.equipo.superttapp.util.RN002;
4
5  import org.junit.Assert;
6  import org.junit.Test;
7
8  public class RN002Test {
9      @Test
10     public void isEmailValid_validEmail_shouldReturnTrue() {
11         Assert.assertTrue("Prrobando que se regrese true en una email valido"
       , RN002.isEmailValid("carlostonatihu@gmail.com"));
12     }
13
14     @Test
15     public void isEmailValid_invalidEmail_shouldReturnFalse() {
16         Assert.assertFalse("Prrobando que se regrese false en una email no
       valido", RN002.isEmailValid("carlostonatihumail.com"));
17     }
18
19     @Test
20     public void isPassordValid_validPassword_shouldReturnTrue() {
21         Assert.assertTrue("Debe retornar true ya que la contrasena es valida"
       , RN002.isPasswordValid("madremiawilly"));
22     }
23
24     @Test
25     public void isPassordValid_invalidPassword_shouldReturnFalse() {
26         Assert.assertFalse("Debe retornar false ya que la contrasena es
       invalida", RN002.isPasswordValid(""));
```

```
27      }
28
29      @Test
30      public void isSecondPassordValid_validPassword_shouldReturnTrue() {
31          Assert.assertTrue("Debe retornar true ya que las dos contrasenas son
        validas", RN002.isSecondPasswordValid("madremiawilly", "madremiawilly"));
32      }
33
34      @Test
35      public void isSecondPassordValid_invalidPassword_shouldReturnFalse() {
36          Assert.assertFalse("Debe retornar true ya que las dos contrasenas son
         validas", RN002.isSecondPasswordValid("madremiawilly", null));
37      }
38
39      @Test
40      public void isNameValid_validName_shouldReturnTrue() {
41          Assert.assertTrue("Debe retornar true ya que el nombre es valido",
        RN002.isNameValid("Carlos Tonatihu"));
42      }
43
44      @Test
45      public void isNameValid_invalidName_shouldReturnFalse() {
46          Assert.assertFalse("Debe retornar false ya que el nombre no es valido
        ", RN002.isNameValid(" "));
47      }
48
49      @Test
50      public void isLastNameValid_validLastName_shouldReturnTrue() {
51          Assert.assertTrue("Debe retornar true ya que el apellido es valido",
        RN002.isLastnameValid("Barrera"));
52      }
53
54      @Test
55      public void isLastNameValid_invalidLastName_shouldReturnFalse() {
56          Assert.assertFalse("Debe retornar false ya que el apellido no es
```

```
    valido", RN002.isNameValid(null));
57   }
58
59   @Test
60   public void isProyectoNombreValid_validProyectoNombre_shouldReturnTrue()
    {
61       Assert.assertTrue("Debe retornar true ya que el nombre del proyecto
    es valido", RN002.isProyectoNombreValid("Proyecto 1"));
62   }
63
64   @Test
65   public void isProyectoNombreValid_invalidProyectoNombre_shouldReturnFalse
    () {
66       Assert.assertTrue("Debe retornar false ya que el nombre del proyecto
    no es valido", RN002.isProyectoNombreValid("1"));
67   }
68 }
```

En el caso de estas pruebas presentadas en el código anterior, se utilizaron los métodos Assert.assertTrue y Assert.assertFalse para verificar que las pruebas fueran correctas. En el caso del primero de estos métodos la prueba es exitosa si el valor que retorna el método a probar es falso, mientras que para el segundo método si el valor de retorno es falso la prueba es correcta.

La ejecución de las pruebas se muestra en la figura 3.2.

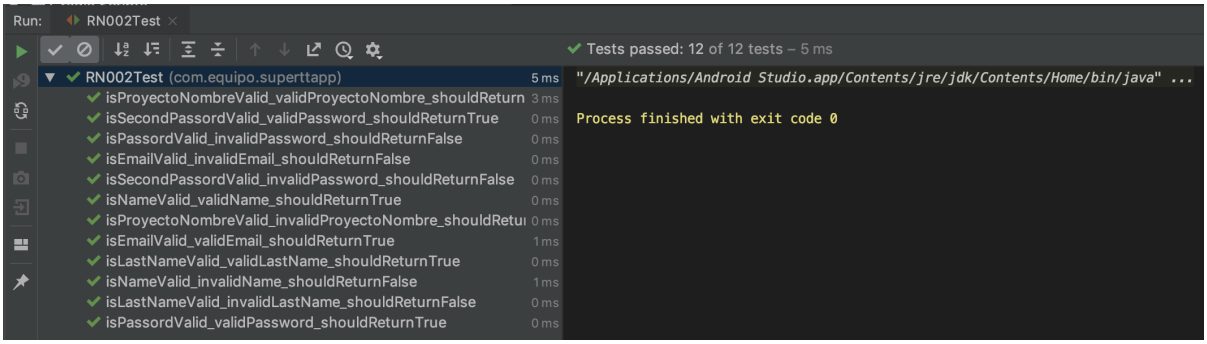

Figura 3.2: Resultados de las pruebas de la clase RNN002

3.2. Pruebas de integración

3.3. Pruebas de requerimientos funcionales

Bibliografía

[1] R. C. Martin, "The Clean Architecture." https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html, 2012. [Consultado: 2019-09-21].

[2] K. Singhal, "CROHME Data Extractor." https://github.com/karan1149/crohme-data-extractor, 2017. [Consultado: 2018-11-01].

[3] M. S. de Lorenzo, "Clean Architecture Guide (with tested examples): Data Flow != Dependency Rule." https://proandroiddev.com/clean-architecture-data-flow-dependency-rule-615ffdd79e29, 2018. [Consultado: 2019-09-21]. III, 7

[4] F. Cejas, "Architecting Android...The clean way?." https://fernandocejas.com/2014/09/03/architecting-android-the-clean-way/, 2014. [Consultado: 2019-09-21]. III, III, 9, 14

[5] F. Cejas, "Architecting Android...Reloaded." https://fernandocejas.com/2018/05/07/architecting-android-reloaded/, 2018. [Consultado: 2019-09-21]. III, 20

[6] C. Larman, *Agile and Iterative Development: A Manager"s Guide*. 01 2004. 5

[7] I. Sommerville, *Ingeniería del software*. 2005. 5

[8] A. Seitz, S, "Computer vision." https://courses.cs.washington.edu/courses/cse576/, 2000. [CSE576: Computer Vision].

[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[10] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*. 2001.

[11] R. H. Anderson, "Syntax directed recognition of hand-printed two-dimensional mathematics," *Harvard University*.

[12] E. Tapia and R. Rojas, "Recognition of on-line handwritten mathematical expressions using a minimum spanning tree construction and symbol dominance," *Freie Universit at Berlin, Institut fur Informatik*.

[13] G. Genthial and R. Sauvestre, "Image to latex," *Stanford University*.

[14] Y. Deng, A. Kanervisto, J. Ling, and A. M. Rush, "Image-to-markup generation with coarse-to-fine attention," *34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017*, 2017.

[15] J. Zhanga, J. Dua, S. Zhanga, D. Liub, Y. Hub, J. Hub, S. Weib, and L. Daia, "Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition," *Pattern Recognition*, p. 196–206, 2017. 22

[16] S. Luján-Mora, "Programación en internet: clientes web," 2001-10-08.

[17] A. Holovaty and J. Kaplan-Moss, "El libro de Django 1.0." `https://uniwebsidad.com/libros/django-1-0/capitulo-5/el-patron-de-diseno-mtv`.

[18] M. Meng, S. Steinhardt, and A. Schubert, "Application programming interface documentation: What do software developers want?," *Journal of Technical Writing and Communication*, vol. 48, p. 295–330, 07 2018.

[19] E. Sundvall, M. Nyström, D. Karlsson, M. Eneling, R. Chen, and H. Orman, "Applying representational state transfer (rest) architecture to archetype-based electronic health record systems," *BMC medical informatics and decision making*, vol. 13, p. 57, 05 2013.

[20] "Identificación y autentificación." `https://www.ibm.com/support/knowledgecenter/es/SSFKSJ_7.5.0/com.ibm.mq.sec.doc/q009740_.htm`.

[21] C. Salazar, "Los principios de SOLID de la programación orientada a objetos." `https://www.codesolt.com/tutoriales/fundamentos/solid/`, 2018. [Consultado: 2019-09-21].

[22] A. Murthy, "Exploring S.O.L.I.D Principle in Android." `https://proandroiddev.com/exploring-s-o-l-i-d-principle-in-android-a90947f57cf0`, 2018. [Consultado: 2019-09-21].

[23] "Mahphix." `https://mathpix.com/about`.

[24] "MyScript Nebo." `https://www.nebo.app/es/`.

[25] F. Álvaro, J.-A. Sánchez, and J.-M. Benedí, "An integrated grammar-based approach for mathematical expression recognition," *Pattern Recognition*, vol. 51, pp. 135 – 147, 2016.

[26] "IDEAL Math Writer." `https://ideal-math-writer.soft112.com/`.

[27] D. Development, "10 Major Differences Between Android and iOS App Development." http://ddi-dev.com/blog/programming/10-differences-between-android-and-ios-app-development/, 2018. [Consultado: 2019-09-21].

[28] "System Properties Comparison MySQL vs. Oracle vs. PostgreSQL." https://db-engines.com/en/system/MySQL%3BOracle%3BPostgreSQL, 2019.

[29] A. Nesmiyanova, "Ruby on Rails vs Django vs Laravel: The Ultimate Comparison of Popular Web Frameworks." https://steelkiwi.com/blog/ruby-django-laravel-frameworks-comparison, 2019.

[30] H. Mouchere, "CROHME: Competition on Recognition of Online Handwritten Mathematical Expressions." http://www.iapr-tc11.org/mediawiki/index.php/CROHME:_Competition_on_Recognition_of_Online_Handwritten_Mathematical_Expressions, 2014. [Consultado: 2018-11-01].