Trabajo de Fin de Grado

Aplicación móvil de apoyo a la interpretación de textos para personas con Autismo

TITULACIÓN: Grado en Ingeniería Informática

AUTOR: Brian Palmés Gómez

TUTORIZADO POR:
Javier Jesús Sánchez Medina

Fecha Julio/2024

Agradecimientos

A mi madre que no hay un sólo día que no haga algo por mí.

A mi padre que aunque se hubiera reído diciendo "por fin"sé que le hacía más ilusión que a nadie .

A mis tres hermanos pequeños ya no tan pequeños.

A mi tío Francisco José por ser el hermano mayor de un hermano mayor.

A mi abuelo Manolo que creía que un día nublado también podía ser un gran día de playa.

Resumen

El Trabajo consiste en el desarrollo de una aplicación para móviles/ tabletas etc.. en entornos Android o IOS la cual ejecutándose en segundo plano tiene el propósito de leer los textos que aparecen en pantalla a vista del usuario y mediante el uso de una API enviarlos a un servidor que interpreta el lenguaje natural y devuelve un valor polar entre -1 y +1.

El objetivo de esta APP es facilitar la comprensión lectora del usuario que se encuentre en una condición médica o de aprendizaje. Por ejemplo un usuario diagnosticado con autismo.

Para ello la aplicación leerá el texto que aparece por pantalla mientras el usuario lo lee. Luego enviará el texto a una API en la cual un módulo con un modelo entrenado para la detección de sentimiento calculará el valor de polaridad del texto entre -1 y +1.

Este modelo será facilitado por el tutor del trabajo en R, y será implementado en el servidor (en el lenguaje que sea preciso) por el alumno.Usando este valor recibido se marcará el texto con colores en base a si tienen un significado positivo o negativo.

En concreto verde para valores cercanos al +1 y rojo para valores cercanos al -1. Esto será en tiempo real mientras el usuario se desplaza por el texto y se ejecutará en segundo plano.

Abstract

Summary of the TFT Mobile application to support the interpretation of texts for people with Autism.

The Project consists of the development of an application for mobiles/tablets, etc. in environments Android or IOS which running in the background has the purpose of reading the texts that appear in screen while the user reads that text then by using an API send them to a server that reads natural language and returns a polar value between -1 and +1.

The goal of this APP is to facilitate the reading comprehension of the user who is in a medical or learning condition. For example, a user diagnosed with autism.

For this APP: It will read the text that appears on the screen while the user reads it. It will send the text to an API in which a module with a model trained for the detection of sentiment will calculate the polarity value of the text, between -1 and +1.

This model will be provided by the tutor of the work in R, and will be implemented on the server (in the language that is necessary) by the student Using this received value the text will be marked with colors based on whether they have a meaning positive or negative.

Specifically green for values close to +1 and red for values close to -1. This will be in real time as the user scrolls through the text and will run in seconds flat.

Índice general

Índice de figuras

Índice de cuadros

Índice de Algoritmos

Capítulo 1

Introducción

> No podemos elegir si usamos los smartphones pero podemos elegir cómo los usamos
>
> Catherine Price autora de "How to Break Up with Your Phone"

Desde que en 2008 [20] se presentó el primer teléfono inteligente hemos visto cómo poco a poco estos pasaban a formar parte de nuestra vida diaria desde aplicaciones de mensajería[11] a aplicaciones de entretenimiento hasta leer el periódico en una aplicación móvil diseñada por su medio favorito.

Con el paso del tiempo no solo se ha convertido en nuestro principal medio de comunicación sino que en muchas ocasiones es una herramienta de trabajo[21] y una fuente de información ya sea leyendo un artículo, un libro o viendo un vídeo sobre un tema que nos interesa.

Sabiendo esto el objetivo de este proyecto de final de carrera es desarrollar una aplicación móvil que ejecutándose en segundo plano ayude a interpretar los textos leídos por el usuario.

Acompañando así a usuarios que les cuesta interpretar la carga emocional en los texto que lee por alguna condición médica o afección como puede ser el caso de las personas con autismo[28] .

Está aplicación usará los servicios de accesibilidad de Android[15] para acompañar al usuario mientras lee texto en internet u otras aplicaciones. Este texto será recogido por nuestra app y envíado a un servidor donde haremos uso de un modelo de procesamiento del lenguaje natural conocido como VADER(Valence Aware Dictionary for Sentiment Reasoning)[18] por sus siglas en inglés.

Este servidor tras la interpretación del texto nos devolverá un valor polar entre -1 y 1 donde -1 es su valor negativo y 1 el más positivo.

Este valor polar se mostrará por pantalla en una burbuja que acompañará al usuario

por las diferentes aplicaciones y le mostrara el valor polar e irá cambiando de color desde el rojo que representa el valor negativo hasta gradualmente el verde que representa una interpretación positiva de los sentimientos del texto.

1.1. Plan de Trabajo

Expondremos la carga de trabajo de la planificación original junto con las horas estimadas y las horas reales que ha llevado el proyecto así como señalar los cambios en la planificación y de las diferentes herramientas[35] .

En cuanto al reparto de horas la principal diferencia de las reales con las que se planearon originalmente es que en el desarrollo e implementación ha llevado más horas de las esperadas mientras que la evaluación y validación ha llevado menos tiempo[30].

Como cambios en las herramientas en principio se iba a utilizar flutter para diseñar esta aplicación debido a los problemas de instalación en el equipo al final decidimos utilizar React-Native[3].

1.2. Organización del Documento

En esta memoria se documentará a continuación el uso de herramientas y librerías para la interpretación de textos[17].

La motivación y objetivo de este proyecto así como una documentación detallada de la implementación técnica de la aplicación[24].

Bien tendrá capítulos dedicados a las competencias específicas a las aportaciones del trabajo y un capítulo sobre normativa y legislación de la Unión Europea[38] y el último capítulo con conclusiones personales y posibles trabajos futuros sobre esta misma aplicación.

1.3. Sobre el Autismo en dispositivos móviles

El autismo es una condición neurológica que afecta a la manera en la que una persona percive a las demás y en cómo socializa con ellos. Sus síntomas van desde dificultades en la comunicación verbal y no verbal, comportamientos repetitivos o un interés intenso por temas específicos y su severidad varía mucho entre los que padecen autismo[23].

Aunque no existe una cura ya que es una condición de por vida las intervenciones tempranas y un buen apoyo educativo pueden producir grandes mejorías en la calidad de vida de estas personas[6].

A la hora de utilizar dispositivos móviles las personas con autismo encuentran dificultades en la comunicación entre acción social e interpretación de la información emocional[Speaks].

El Trastorno del Espectro Autista (TEA) está caracterizado por una gran variación en las habilidades y comportamientos de los individuos es decir no todas las personas con autismo experimentan los mismos desafíos o síntomas pero hay algunas áreas comunes en las que presentan todos dificultades[for Disease Control and Prevention].

Uno de estos desafíos comunes es la sobrecarga sensorial que las personas que sufren autismo tienen a los estímulos sensoriales como sonidos fuertes, luces brillantes y cambios rápidos en una pantalla[4].

Todo esto son características propias de la experiencia de utilizar dispositivos móviles o alguna aplicación que suelen encontrar la abrumadora o estresante. Esto es debido a que las interfaces de usuario suelen presentar mucha información de forma desordenada, cambian rápidamente o son muy diferentes de unas aplicaciones a otras y esto hace que las personas con autismo encuentren en los dispositivos móviles una experiencia compleja ya que al no seguir una estructura siempre predecible les causa una sensación de dificultad[Goodman].

Por otro lado también encuentran dificultades a la hora de interpretar y responder las señales emocionales. Las personas con autismo les cuesta reconocer y entender las emociones tanto la comunicación verbal como no verbal[1]. Esto en nuestro caso se extiende a la interpretación de textos escritos, por ejemplo en una aplicación donde al no encontrar señales claras de emoción verbal o no verbal les dificulta entender el tono o la intención detrás de un mensaje o de un texto por tanto en el entorno de un dispositivo móvil donde la comunicación escrita predomina esto les lleva constante malentendidos y a una disminución de la calidad de la información que reciben a través de estos dispositivos[26].

El objetivo al diseñar nuestra aplicación es ayudar a superar algunos de estos desafíos específicos y proporcional una herramienta útil y accesible para que las personas con autismo utilizando los servicios de accesibilidad de Android con los que nuestra aplicación captura el texto que el usuario lee en tiempo real y lo envía a un servidor donde ese texto será analizado con un modelo de procesamiento del lenguaje natural llamado VADER que evaluará la carga emocional del texto recibido y devolverá un valor polar que oscila entre -1 y +1[18].

El valor de ese texto pretende informar al usuario a través de una burbuja flotante en la parte superior de la pantalla que cambiará de color junto con el valor polar para indicar si el texto de una carga emocional negativa, más bien neutra o positiva.

La aplicación se ha diseñado con la intencionalidad de ser lo más intuitiva posible. La simplicidad de la interfaz siendo simplemente una burbuja que cambia de color con un valor polar pretende ayudar a los usuarios a encontrar en la información esencial al leer un texto y recibir información adicional de la carga emocional del texto a través del valor de la burbuja queriendo con esto mejorar la comprensión de la información y aumentar la confianza de independencia de las personas con autismo en la interpretación de textos en dispositivos móviles[Society].

Capítulo 2

Estado actual y objetivos iniciales

2.1. Estado Actual. Modelo de análisis de sentimientos

El análisis de sentimientos consiste en un análisis de texto que gracias a la evolución de los procesamiento de lenguaje natural (NPL) y en devolver la carga emocional ya sea positiva, negativa o neutra que contenga este texto debido a los avances de los últimos años en este campo han aparecido todo tipo de aplicaciones comerciales que han hecho que la industria del análisis de sentimientos crezca[27].

2.1.1. Librería Natural Language Toolkit (NTKL)

Haremos uso de la librería de lenguaje natural NKTL que significa por sus siglas en inglés Natural Language Toolkit es una librería ampliamente utilizada en el procesamiento natural del lenguaje en Python creada por Steven Bid y Edward Lopez[2].

Ampliamente utilizada en los ámbitos de investigación enseñanza y en el desarrollo de aplicaciones que utilizan el procesamiento de lenguaje natural debido a lo bien documentada que se encuentra y a su amplia gama de herramientas y recursos[22].

En concreto usaremos VADER(Valence Aware Dictionary and sEntiment Reasoner) qué es un método de sentimientos que usa un diccionario de palabras que tienen carga sentimental a la que adjudica unos valores positivos o negativos y el uso de reglas heurísticas con el objetivo de interpretar la carga sentimental en textos breves[18].

2.1.2. Diccionario de Vader.

El principal problema de VADER es que su diccionario originalmente se encuentra escrito en inglés y su versión en castellano es simplemente una traducción directa hecha por un

traductor automático al español directamente palabra por palabra del inglés por tanto esta traducción hereda esas cargas que definen el sentimiento de esas palabras en el diccionario por tanto la precisión del análisis de sentimientos de VADER en castellano baja mucho con respecto a el análisis en inglés.[18]

2.2. Objetivos Iniciales

2.2.1. Objetivos Iniciales

El objetivo de esta app es facilitar la comprensión lectora del usuario que se encuentra en una situación médica de aprendizaje por ejemplo un usuario diagnosticado con autismo. Para ello nuestra aplicación:

- Leerá el texto que aparece por pantalla mientras el usuario lo lee en tiempo real.

- Enviará este texto a una API en la cual un modelo entrenado para el análisis de sentimientos calculará la carga emocional del texto devolviendo un valor de polaridad entre el -1 y el +1 .Este modelo será facilitado por el tutor del trabajo en R, e implementado en un servidor al que se enviará el texto por el alumno.

–Usando este valor recibido se marcará el texto con colores en base a si tienen un significado positivo o negativo. En concreto verde para valores cercanos al +1 y rojo para valores cercanos al -1.

-Esto será en tiempo real mientras el usuario se desplaza por el texto y se ejecutará en segundo plano.

2.2.2. Cambios en los Objetivos Iniciales

Finalmente el modelo no será en R sino que utilizaremos la librería de análisis de sentimiento NKTL en concreto el modelo VADER.

Otro cambio será que el texto que lea el usuario no se marcará ni cambiará de color sino que se implementará una pequeña burbuja que aparecerá en la parte superior de la pantalla con un valor polar entre el -1 y +1 y rodeada por un color que irá del rojo al verde según la carga sentimental del texto que esté siendo leído por el usuario.

Capítulo 3

Competencias específicas y aportaciones del trabajo

3.1. Competencias específicas cubiertas

3.1.1. Competencia CP06

Esta competencia refiere a la capacidad para desarrollar y evaluar sistemas interactivos y de presentación de información compleja así como su aplicación a la resolución de problemas de diseño de interacción persona computadora.

Competencia que cumplimos para desarrollar nuestra aplicación usamos un servicio de accesibilidad de Android interactivo que recoge la información en forma de texto que lee el usuario y la envía vía API a un servidor donde se realizar el análisis de sentimiento

3.1.2. Competencia CP07

Hace referencia a la capacidad para conocer y desarrollar técnicas de aprendizaje computacional y diseñar e implementar aplicaciones y sistemas que las utilicen, incluyendo las dedicadas a extracción automática de información y conocimiento a partir de grandes volúmenes de datos .

Nuestro proyecto también cumple esta competencia ya que utiliza sistemas de análisis de sentimiento y se diseña una aplicación que extrae la información de forma automática que lee el usuario .

3.2. Aportaciones del trabajo

3.2.1. Motivación

La principal motivación en el desarrollo de esta aplicación es proporcionar una herramienta de apoyo a personas con una condición médica que dificulte la interpretación de textos con carga emocional por ejemplo un artículo de opinión de un periódico problema al que se enfrentan muchas personas con autismo.

3.2.2. Investigación

Porque este podría ser una primera app para acompañar en la interpretación de textos a personas con este tipo de afecciones médicas todavía queda mucho camino por recorrer.

Desarrollo de esta app nos hemos enfrentado a diferentes desafíos como el hecho de que muchas páginas web tienen diferentes formas de mostrar la información y que las mayorías de las aplicaciones en las que ayudamos al usuario a interpretar los textos usan text viewers una librería de Android que te permite desplegar un navegador dentro de la propia aplicación[Google].

Esto dificulta mucho la interpretación de textos ya que la librería de Android y de accesibilidad recorre la app siguiendo una estructura y extrayendo los textos de esta misma pero al ser un navegador esa estructura no siempre es igual debido a que cada diseñador web y cada empresa diseña su web de diferente manera[29].

Además de que esta solo se encuentra disponible para Android y quizás algún compañero haciendo un trabajo de final de grado similar a este podría expandir a entornos iOS o hacer un tratamiento más preciso de los textos cuando la aplicación se encuentra anuncios dentro de la aplicación o dentro de la web que está leyendo[Inc.].

También el modelo de interpretación del lenguaje natural que hemos utilizado es mucho más preciso en inglés que en castellano, quizá con el tiempo vayan apareciendo modelos que también sean precisos en castellano.

3.2.3. Social

Esta aplicación puede servir para dar visibilidad a los problemas que tienen la gente que sufre afecciones como el autismo para la interpretación de textos y la importancia de la accesibilidad en las aplicaciones y páginas web.

Como decíamos anteriormente muchas aplicaciones móviles de tipo prensa simplemente cargan un navegador dentro de su app mostrando así su web lo que dificulta mucho la librería de accesibilidad la interpretación de textos quizás con el tiempo se coja conciencia de esto y especialmente las aplicaciones de prensa empiezan a tener esto en cuenta.

3.2.4. Contra el Ludismo

También esta aplicación puede ayudar a que la sociedad vea con otros ojos a la inteligencia artificial ya que debido a los avances de esta última en estos últimos años hemos visto nacer cierta corriente ludita que rechaza el uso de inteligencia artificial en nuestro día a día[25].

Aunque es verdad que la inteligencia artificial se puede utilizar con malos propósitos también puede ser una herramienta muy valiosa ayudando a progresar como puede ser en el campo de la medicina o en el caso de esta aplicación un uso práctico para personas que pueden necesitar una herramienta de apoyo a la hora de leer textos[16].

Capítulo 4

Herramientas

4.1. React-Native

4.1.1. Desarrollo móvil

Es un framework de código abierto creado por Facebook que se utiliza para desarrollar aplicaciones en iOS y Android compartiendo gran parte del código e integrarlo con características nativas de estas plataformas .

Permite construir aplicaciones móviles utilizando Javascript y React que es un framework que se usa en el diseño de interfaces de usuario en aplicaciones web. Además React-Native facilita la creación de aplicaciones móviles nativas para iOS y Android que comparten una gran parte del código reduciendo así el tiempo y costo de desarrollo .

Ilustración 4.1: React-Native

4.1.2. Características

Sus principales características encontramos

Los componentes que usamos en React-Native se traducen a componentes nativos en iOS y Android ya que reutiliza mucho código de forma que conseguimos un rendimiento cercano al nativo y una mejor experiencia para el usuario.

El Hot Reloading es una característica que permite a los desarrolladores ver los cambios en tiempo real a medida que diseñan la interfaz .

Está muy bien documentado y cuenta con una amplia gama de librerías y componentes de terceros lo que facilita la implementación de funcionalidades además posee una de las mayores comunidades de desarrolladores cuando hablamos de desarrollo de aplicaciones móviles.

4.1.3. Comparación con Flutter

Flutter es otro framework de código abierto pensado para desarrollar aplicaciones móviles multiplataforma.

Y aunque se barajó como primera opción se acabó optando por React-Native debido a React-Native usa JavaScript que es un lenguaje más conocido y usado mientras que Flutter usa Dart un lenguaje aunque pensado para el desarrollo móvil es menos usado y conocido. La comunidad de React-Native también es mayor lo que a la hora de desarrollar un proyecto siempre se tiene en cuenta ya que suele haber más recursos y soluciones por parte de otros usuarios.

4.2. Kotlin

4.2.1. Desarrollo móvil

Es uno de los mejores lenguajes de programación para desarrollar en Android tiene una síntesis concisa y clara un gran rendimiento y una curva de aprendizaje relativamente sencilla.

Fue desarrollado por JetBrains y se ejecuta en la máquina virtual de Java .Es el lenguaje oficial de Google para desarrollar aplicaciones Android lo que ha aumentado su uso para desarrollar aplicaciones móviles.

4.2.2. Comparación con Java

Aunque Java ha sido siempre el lenguaje de las aplicaciones Android nos decantamos por Kotlin ya que es completamente interoperable con Java pero hacen el desarrollo más rápido y sencillo ya que es un lenguaje más moderno por ejemplo tiene la nulabilidad controlada y

Ilustración 4.2: Kotlin

Ilustración 4.3: Visual Studio Code

es tan parecido a Java que la curva de aprendizaje para alguien que haya trabajado con Java es muy suave.

4.3. Visual Studio Code

Cómo editor de texto se ha usado principalmente Visual Studio Code que es uno de los editores de texto más usados por su facilidad para la depuración de código, integración con Git y finalización inteligente de código que facilita y agiliza la programción.

Desarrollado por Microsoft es de uso gratuito y de código abierto.

4.4. Android Studio

Android Studio es un IDE pensado para la plataforma Android y es el IDE oficial para el desarrollo de aplicaciones para la plataforma Android es gratuito y de código abierto.Usa como base el software IntelliJ IDEA desarrollado por JetBrains.

En nuestro caso hemos usado principalmente Visual Studio Code pero las herramientas de android Studio para encontrar errores como el LogCat o la facilidad para instalar Android en el equipo vía Android Studio así que aunque no lo hemos usado como un IDE al uso sí hemos usado sus herramientas.

4.4.1. Comparación con IntelliJ IDEA

Aunque IntellJ IDEA es más flexible y puede usar muchos lenguajes de programación nos decantamos por Android Studio porque está específicamente pensado para para el desarrollo

Ilustración 4.4: Android Studio

de Android y tiene herramientas que nos han resultado de gran utilidad como el LogCat.

4.5. Replit

Replit es un IDE que se puede usar de forma online para desarrollar aplicaciones y escribir código en el navegador en nuestro caso lo hemos usado para simular el servidor de forma que el programa que recibe el texto desde la aplicación Android se ejecute en esta plataforma y devuelva el valor polar como respuesta simulando así la parte del servidor de este proyecto.

4.5.1. Comparación con Glitch

Glitch es otra plataforma que permite ejecutar aplicaciones en el navegador y aunque es considerada más sencilla de usar nos decantamos por Replit porque maneja mayor número de lenguajes y puedes hacer proyectos más complejos.

Capítulo 5

Normativa y legislación.

5.1. Reglamento General de Protección de Datos (GDPR)

"El Reglamento General de Protección de Datos (GDPR) garantiza la protección de los datos personales y la privacidad de los individuos dentro de la Unión Europea (UE) y el Espacio Económico Europeo (EEE)"[41].

Es el reglamento que se aplica a cualquier aplicación o sistema que maneje datos personales en la Unión Europea y establece las obligaciones sobre cómo deben ser gestionados y protegidos los datos de los usuarios.

Este reglamento requiere que cualquier dato personal recogido cumpla las disposiciones del GDRP y exige obtener el consentimiento explícito de los usuarios para recopilar y procesar sus datos personales también obliga a informa a los usuario sobre el uso que se da a sus datos y el derecho a borrar esa información si así lo desean.

En cuanto a nuestra aplicación aunque lee los datos del usuario lo hace usando servicios de accesibilidad previamente aprobados por el usuario y en este caso no se almacena ningún dato aunque el texto se envía al servidor allí los datos ni se alamacenan ni se usan para nada una vez son pasados por el modelo de análisis de sentimientos así que no se nos aplicaría el tener que gestionar los datos de los usuarios porque no los usamos ni los alamacenamos.

5.2. Directiva de Accesibilidad Web (WAD)

"La Directiva (UE) 2016/2102 del Parlamento Europeo y del Consejo sobre la accesibilidad de los sitios web y aplicaciones móviles de los organismos del sector público exige que los sitios web y las aplicaciones móviles del sector público cumplan con ciertos estándares de accesibilidad"[40].

Esta directiva intenta garantizar que las aplicaciones móviles se diseñen intentando ser

accesibles para todos los usuarios independientemente de sus capacidades físicas o tecnológicas.

En el caso de nuestra aplicación, que cuyo objetivo es hacer la lectura más accesible a las personas con dificultades para interpretar textos la cumplimos.

5.3. Ley de Igualdad de la Unión Europea

"La Ley de Igualdad de la Unión Europea (Directive 2000/78/EC) establece un marco general para la igualdad de trato en el empleo y la ocupación, protegiendo a las personas con discapacidades y asegurando que tengan igualdad de oportunidades"[7].

En este caso al igual que con la WAD nuestra aplicación colabora activamente con el espíritu de esta ley ya que se basa en mejorar los servicios de accesibilidad.

5.4. Reglamento ePrivacy

El Reglamento ePrivacy, también conocido como la Regulación de Privacidad Electrónica, es una propuesta de la Comisión Europea para reforzar la privacidad en las comunicaciones electrónicas, complementando al GDPR"[9].

Esta ley solo está propuesta pero promete ser importante en cuanto a la gestión de la privacidad en las comunicaciones electrónicas. Para nuestra aplicación solo queda estar pendientes de a qué cambios podría obligarnos esta ley a medida que avanza por el proceso legislativo europeo.

Capítulo 6

Metodología

En este trabajo acordamos una metodología SCRUM para el desarrollo del proyecto[33]. Scrum es un marco de trabajo dentro de las metodologías ágiles para el desarrollo de software y aplicado también a otras industrias[32].

Con Scrum el trabajo de los equipos queda dividido en objetivos más pequeños que se van cumplimentando a medida que avanza el proceso de desarrollo en unos periodos de tiempo iterativos conocidos como sprints[31].

La idea es que cada sprint no dure más de un mes y generalmente suelen ser periodos de 15 días. Cada equipo de Scrum establece unos objetivos en un tiempo delimitado y tienen una reunión de 15 minutos a diario generalmente al final de cada sprint el equipo scrum tiene dos reuniones adicionales una para enseñar el valor del trabajo realizado los accionistas y otra de carácter interno sobre el propio sprint.

Una ventaja de la metodología SCRUM para desarrollar un producto es que principalmente lleva la toma de decisiones al nivel operacional a diferencia de otros metodologías secuenciales para desarrollar productos scrum es iterativa e incremental de esta manera permite a los equipos tener flexibilidad y feedback de los compañeros[33].

6.1. Equipo SCRUM

Generalmente un equipo SCRUM tiene al menos tres roles por un lado está el ”Product Owner”, luego los desarrolladores y finalmente el ”SCRUM Master”[31].

6.1.1. Product Owner

Generalmente cada equipo de SCRUM tiene un Product Owner este rol se encarga de la parte del negocio del producto y en un ambiente corporativo la mayoría del tiempo es el que responde ante los accionistas y el que tiene la responsabilidad de entregar el resultado a nivel

del negocio. Es el responsable de generar el máximo valor para estos aunque no suele estar implicados en la solución es técnicas de un equipo si tienen una gran implicación en crear consensos entre los miembros de los equipos.

6.1.2. Desarrolladores

En un marco SCRUM los desarrolladores son los que desempeñan un rol de desarrollo, soporte investigación, arquitectos, programadores, diseñadores o otros miembros del equipo que participan activamente sin un rol de gestión en el proceso.

6.1.3. SCRUM Master

Un proyecto que aplica la Metodología SCRUM está dirigido por un Scrum Master este rol principalmente dirige y enseña a los jefes de equipo sobre la teoría de la metodología SCRUM y cómo llevarla a la práctica en el proyecto que están desarrollando. Sus principales tareas incluyen la resolución de problemas tanto diarios como otros niveles comunicación entre equipos selección de objetivos para estos equipos y planificación[37] .

6.2. Características de SCRUM

Entre las principales características de SCRUM se encuentran un desarrollo incremental del proyecto limitado en bloques temporales cortos y fijos.

Esta metodología se centra en aportar el máximo valor para el cliente los equipos trabajan todos los días mediante pequeñas reuniones y se realizan las adaptaciones necesarias.

Estos equipos tienen poder de decisión y de autogestionarse para ir adaptándose a los diferentes desafíos que presentan los requisitos en cada iteración se muestra el cliente o los accionistas el resultado obtenido para recibir una valoración de estos[31].

6.3. Flujo de trabajo

6.3.1. Sprint

Es un período de tiempo finito predefinido previamente en el que se lleva a cabo en determinadas tareas que contribuyen al desarrollo final del producto. Al final de cada sprint los equipos presentan los progresos conseguidos deben presentar un producto que se podría presentar a un cliente o a los accionistas.

Generalmente el tiempo de un sprint dependiendo de la empresa o del proyecto varía de una semana a cuatro aunque se pueden hacer pequeños ajustes de tiempo si fuera necesario y si el proyecto lo requiere .

6.3.2. Scrum Diario

Donde los miembros del equipo se mantienen utilizados unos a otros sobre el estado en el que se encuentra el trabajo desde la última reunión qué problemas encuentran y qué soluciones proponen.

Estas reuniones suelen ser cortas entre 15 y 20 minutos y asisten los diferentes roles y equipos del proyecto .

6.3.3. Análisis del Sprint

Cuando termina un sprint se llevan a cabo tareas de revisión y análisis del sprint completado en ellas los miembros de los equipos hacen un análisis sobre el sprint que ha finalizado el propósito de este sprint es analizar y mejorar los errores que se hayan podido cometer y qué cosas se podrían mejorar y así en cada iteración conseguir un estado de mejora continua de la implementación de la metodología SCRUM.

6.4. Aplicado a nuestro proyecto

Para la realización de nuestra aplicación hemos utilizado la metodología Scrum pero hay que señalar ciertos detalles.

Primero el sistema de roles no tiene sentido ninguno aplicarlo ya que esto es un proyecto de final de carrera llevado de manera individual donde como alumno asumo las tareas de los 3 roles mencionados anteriormente.

Pero si aplicaremos el desarrollo iterativo y la mecánica de los sprints.

6.4.1. Sprint 1 Captura de texto

En este primer sprint nos marcamos como objetivo conseguir que la aplicación leyese el texto que el usuario veía por pantalla meintras navegaba por internet o de otras aplicaciones siempre en segundo plano.

También una importante fase de toma de desiciones como cambiarnos de la idea original de utilizar el Framework de Flutter para usar React Native que resultó más facil de instalar en nuestro equipo.

Luego se decidió usar la librería de Android android.view.accessibility que nos permite capturar los diferentes eventos de accesibilidad dentro de lo que ve el usuario.Después se entrará en detalle con esta librería.

Pero principalmente se toma esta decisión ya que los sistemas operativos de los móviles tienen mucho cuidado por razones de privacidad en no permitir que otras aplicaciones espien lo que está leyendo el usuario actual.

Como el objetivo de esta aplicación es mejorar la accesibilidad de las personas con problemas a la hora de interpretar la carga sentimental en los textos esta librería resulta perfecta para la implementación de nuestra aplicación pero respetando al mismo tiempo la privacidad del usuario.

6.4.2. Sprint 2 Envío de texto a un servidor.

Una vez capturado el texto correctamente usando la libreria de HTTP OkHttp que se usa para el manejo de solicitudes HTTP en aplicaciones Android.

Este texto se envía a un servidor donde usaremos la librería de analisis de sentimientos NKTL en concreto su modelo VADER este programa diseñado y ejecutado en el servidor leerá el texto envíado y devolverá un valor polar entre -1 y +1 que será enviado de vuelta a la aplicación.

6.4.3. Sprint 3 Mostrar por pantalla el valor polar.

En este último sprint se diseña la burbuja que mostrará el valor polar y su implementación y se realizarán pruebas con diferentes tipos de web de prensa y articulos con diferentes cargas sentimentales.

Capítulo 7

Desarrollo

En este capitulo abordaremos en detalle las tecnologías y librerías utilizadas así como la toma de decisiones de diseño e implementación de esta aplicación.

Se desarrolla una aplicación para Android que se ejecutará en segundo plano recogiendo el texto que lee el usuario en tiempo real y enviando a un servidor que respondera con un valor polar entre +1 y -1 indicando la carga sentimental del texto que está leyendo el usuario[14].

Este texto se mostrara en una burbuja a modo de interfaz que el usuario podrá desplazar por la pantalla y que cambiará de color según el valor polar devuelto.

Para recoger el texto con éxito se ha recurrido a una librería propia de Android y pensada para la accessibilidad ya que de cualquier otra manera el sistema operativo no te permite leer lo que ve el usuario por temas de privacidad[8].

Esta librería de Accesibilidad es la que permitirá el desarrollo y correcto funcionamiento de nuestra aplicación y se darán los detalles técnicos más adelante en este capítulo.

7.1. Android View Accessibility

La librería de accesibilidad de Android es una herramienta oficial proporcinada por Android para facilitar la tarea a los desarrolladores de crear aplicaciones más inclusivas o para personas con necesidades específicas con el objetivo de que estos usuarios tengan un acceso a las aplicaciones sin las barreras que les impone su condición médica[5].

Esta librería entre otras cosas proporciona un conjunto de APIs que permiten por ejemplo el diseño de interfaces más accesibles, tecnologías de asistencia por voz y como en nuestro caso el desarrollo de aplicaciones que necesitan permiso del usuario para que el sistema operativo no impida la lectura de texto que ve el usuario por pantalla[39].

7.1.1. Servicios de Accesibilidad

La librería de Accesibilidad utiliza servicios para interactuar con el dispositivo Android. Los servicios cumplen funciones como esperar por eventos de accesibilidad u obtener información detallada de la interfaz de usuario que ve el usuario así como realizar ciertas acciones por él[8].

Por ejemplo leer en voz alta los textos, navegación por texto o como en este caso información detallada sobre el texto que ve el usuario por pantalla.

7.1.2. Eventos de Accesibilidad

La librería trabaja con un sistema de eventos que se usan para notificar a los servicios de Accesibilidad de Android cuando ocurre un cambio respecto a algún elemento que el servicio de Accesibilidad de Android está monitorizando.

Los desarrolladores pueden configurar a placer estos eventos definiendo mejor el uso de su aplicación y a qué elementos debe prestar atención el servicio así como qué acciones del usuario debe vigilar de manera que el servicio pueda reaccionar en consecuencia[5].

7.1.3. Nodos de Accesibilidad

La librería entiende como nodos de accesibilidad a los diferentes elementos de la interfaz de usuario y contienen toda la información sobre ese elemento de la vista (Android.View) estructurar la interfaz de esta manera permite a los servicios y los eventos reconocer la diferentes interacciones así como los cambios que se producen en la vista para tratarlos correctamente.

7.2. Captura de texto con Accesibilidad

Para nuestro proyecto el principal uso de la librería de Accesibilidad de Android se centra en el tratamiento del texto que ve el usuario por pantalla en tiempo real es decir mientras se mueve por ejemplo por una pagina web[8].

7.3. Estructura del proyecto.

Para desarrollar este este proyecto hemos usado el framework React-Native comúnmente utilizado para desarrollar aplicaciones móviles multiplataforma.

La estructura de los proyectos que usan React-Native se basa en componentes y estilos en JavaScript de manera que se hace más sencillo el desarrollo del proyecto y el mantenimiento del código.

Es fácil de configurar, es robusto y multiplataforma que te permite crear aplicaciones desde una misma base del código.

Una vez finalizada la instalación inicial con el comando en el directorio donde se encuentra tu proyecto:

```
react-native init Sarcasmo
```

Encontrarás el proyecto base sobre el que empezar con una estructura en la que destacan:

```
node_modules/: Donde estan instaladas las dependencias del proyecto

ios/: Carpeta con los archivos especificos para desarrolla la aplicacion para iOS

android/: carpeta con los archivos nativos de android que seran de gran importancia en nuestro proyecto

App.js : es el archivo principal de un proyecto React-Native y se encuentra en la raiz

index.js : archivo que registra el componente raiz

package.json : importante archivo donde se define la informacion sobre el proyecto y especifica las dependencias
```

7.4. Clase CapturaAccessibilityService.kt

En esta sección analizaremos de forma detallada la clase CapturaAccessibilityService.kt que es la que controla todo el funcionamiento de los servicios de Accesibilidad de Android por tanto es una clase fundamental para el funcionamiento de esta aplicación.

```
package com.sarcasmo

import android.accessibilityservice.AccessibilityService
import android.content.Context
import android.graphics.PixelFormat
import android.graphics.Point
import android.graphics.Rect
import android.os.Handler
import android.os.Looper
import android.util.Log
import android.view.*
import android.view.accessibility.AccessibilityEvent
import android.view.accessibility.AccessibilityNodeInfo
import android.widget.FrameLayout
import android.widget.TextView
import okhttp3.*
import okhttp3.MediaType.Companion.toMediaTypeOrNull
import org.json.JSONObject
import java.io.IOException
import com.facebook.react.bridge.ReactApplicationContext
import com.facebook.react.modules.core.DeviceEventManagerModule
```

```
class CapturaAccessibilityService : AccessibilityService() {

    companion object {
        private const val TAG = "CapturaAccessibilityService"
        private const val API_URL = "https://b818c21e-0a56-455b-afa9-ace0493a0696-00-
␣␣␣␣␣␣␣␣1r1ou3r2wrszv.worf.replit.dev/analyze"
    }

    private lateinit var windowManager: WindowManager
    private lateinit var floatingBubble: FrameLayout
    private lateinit var bubbleTextView: TextView
    private val handler = Handler(Looper.getMainLooper())

    private var initialX = 0
    private var initialY = 0
    private var initialTouchX = 0f
    private var initialTouchY = 0f

    override fun onCreate() {
        super.onCreate()
        windowManager = getSystemService(Context.WINDOW_SERVICE) as WindowManager
        initializeFloatingBubble()
    }

    private fun initializeFloatingBubble() {
        floatingBubble = LayoutInflater.from(this).inflate(R.layout.floating_bubble, null) as FrameLayout
        bubbleTextView = floatingBubble.findViewById(R.id.bubble_text)

        val params = WindowManager.LayoutParams(
            WindowManager.LayoutParams.WRAP_CONTENT,
            WindowManager.LayoutParams.WRAP_CONTENT,
            WindowManager.LayoutParams.TYPE_APPLICATION_OVERLAY,
            WindowManager.LayoutParams.FLAG_NOT_FOCUSABLE or
            WindowManager.LayoutParams.FLAG_NOT_TOUCH_MODAL or
            WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS,
            PixelFormat.TRANSLUCENT
        )
        params.gravity = Gravity.TOP or Gravity.START
        params.x = 0
        params.y = 100

        windowManager.addView(floatingBubble, params)

        floatingBubble.setOnTouchListener { _, event ->
            when (event.action) {
                MotionEvent.ACTION_DOWN -> {
                    initialX = params.x
                    initialY = params.y
                    initialTouchX = event.rawX
                    initialTouchY = event.rawY
                    true
                }
                MotionEvent.ACTION_MOVE -> {
                    params.x = initialX + (event.rawX - initialTouchX).toInt()
                    params.y = initialY + (event.rawY - initialTouchY).toInt()
                    windowManager.updateViewLayout(floatingBubble, params)
                    true
                }
                else -> false
```

```
        }
    }
}

override fun onAccessibilityEvent(event: AccessibilityEvent) {
    if (event.eventType == AccessibilityEvent.TYPE_WINDOW_CONTENT_CHANGED ||
        event.eventType == AccessibilityEvent.TYPE_VIEW_SCROLLED ||
        event.eventType == AccessibilityEvent.TYPE_VIEW_FOCUSED) {
        val rootNode = rootInActiveWindow
        if (rootNode != null) {
            val visibleText = StringBuilder()
            collectVisibleText(rootNode, visibleText)
            if (visibleText.isNotEmpty()) {
                val text = visibleText.toString()
                Log.d(TAG, "Captured␣Text:␣$text")
                analyzeText(text)
            }
        }
    }
}

private fun collectVisibleText(node: AccessibilityNodeInfo, visibleText: StringBuilder) {
    if (node.isVisibleToUser && node.text != null && isNodeWithinScreenBounds(node)) {
        visibleText.append(node.text).append("\n")
    }
    for (i in 0 until node.childCount) {
        val childNode = node.getChild(i)
        if (childNode != null) {
            collectVisibleText(childNode, visibleText)
            childNode.recycle()
        }
    }
}

private fun isNodeWithinScreenBounds(node: AccessibilityNodeInfo): Boolean {
    val screenRect = Rect()
    node.getBoundsInScreen(screenRect)

    val screenSize = Point()
    windowManager.defaultDisplay.getSize(screenSize)

    return screenRect.intersect(0, 0, screenSize.x, screenSize.y)
}

override fun onInterrupt() {
    Log.e(TAG, "Service␣Interrupted")
}

private fun analyzeText(text: String) {
    val sanitizedText = text.replace("[\\u0000-\\u001F]".toRegex(), "")
    val client = OkHttpClient()
    val mediaType = "application/json".toMediaTypeOrNull()
    val body = RequestBody.create(mediaType, "{\"text\":\"$sanitizedText\"}")
    val request = Request.Builder()
        .url(API_URL)
        .post(body)
        .build()

    client.newCall(request).enqueue(object : Callback {
```

```
            override fun onFailure(call: Call, e: IOException) {
                Log.e(TAG, "API␣Request␣Failed", e)
            }

            override fun onResponse(call: Call, response: Response) {
                if (response.isSuccessful) {
                    val responseData = response.body?.string()
                    Log.d(TAG, "Text␣Analysis␣Response:␣$responseData")
                    responseData?.let {
                        val jsonResponse = JSONObject(it)
                        val compound = jsonResponse.getDouble("compound")
                        updateBubble(compound)
                        sendEventToReactNative("TextAnalysisEvent", it)
                    }
                }
            }
        })
    }

    private fun updateBubble(compound: Double) {
        handler.post {
            val green = (255 * ((compound + 1) / 2)).toInt()
            val red = (255 * (1 - (compound + 1) / 2)).toInt()
            val color = 0xFF000000.toInt() or (red shl 16) or (green shl 8)

            bubbleTextView.setBackgroundColor(color)
            bubbleTextView.text = String.format("%.2f", compound)
        }
    }

    private fun sendEventToReactNative(eventName: String, eventData: String) {
        val reactContext = (applicationContext as MainApplication).reactNativeHost.reactInstanceManager.currentReactContex
        reactContext?.getJSModule(DeviceEventManagerModule.RCTDeviceEventEmitter::class.java)
            ?.emit(eventName, eventData)
    }
}
```

Una vez mostrado el código ahora lo explicaremos en detalle. Esta clase controla todo el servicio de Accesibilidad de Android

```
class CapturaAccessibilityService : AccessibilityService() {
```

Aquí se define el nombre de la clase extendiendo de AccessibilityService de manera que la clase se comporte como un servicio de Accesibilidad de Android.

A continuación se definen una series de variables que utilizaremos para controlar el servicio de Accesibilidad.

Empezando por las variables tipo companion object que son variables estáticas una variable TAG para utilizar luego en los logs de la consola durante la solución de problemas y otra API-URL que recoge la dirección del servidor donde se enviará el texto para ser tratado.

```
    companion object {
        private const val TAG = "CapturaAccessibilityService"
        private const val API_URL = "https://b818c21e-0a56-455b-afa9-ace0493a0696-00-
␣␣␣␣␣␣␣␣1r1ou3r2wrszv.worf.replit.dev/analyze"
    }
```

```
    private lateinit var windowManager: WindowManager
    private lateinit var floatingBubble: FrameLayout
    private lateinit var bubbleTextView: TextView
    private val handler = Handler(Looper.getMainLooper())

    private var initialX = 0
    private var initialY = 0
    private var initialTouchX = 0f
    private var initialTouchY = 0f
```

A continuación se definen variables una tipo WindowManager que es una clase de Android que se usa para el manejo de las ventanas que muestra la pantalla del dispositivo esta clase contiene la infomación necesaria para añadir eliminar y modificar ventanas.

En nuestro caso la usaremos para manejar la burbuja flotante donde se volcará el valor polar devuelto por el servidor una vez leído el texto.

Después se declaran otras tres variables una floatingBubble y otra bubbleTextView estás clases son el contenedor de la burbuja y el texto que irá dentro de esa burbuja y están definidas en otros ficheros que se verán más adelante.

Y declaramos también una clase tipo Handler que se usa para asegurarnos que el código se ejecutarán en el hilo principal de ejecución.

Se declaran varias variables que serán utilizadas para maenjar la posición de la burbuja y la posición en la que el usuario la toca.

Se declara un método onCreate() que inicializa el servicio WindowManager e invoca al método initializeFloatingBubble().

```
override fun onCreate() {
        super.onCreate()
        windowManager = getSystemService(Context.WINDOW_SERVICE) as WindowManager
        initializeFloatingBubble()
    }
```

7.4.1. Función initializeFloatingBubble()

Esta función privada configura y controla la burbuja flotante donde se mostrará el valor polar devuelto por el servidor permitiendo que el usuario la mueva libremente por la pantalla para ubicarla donde le resulte más cómodo.

Primero se crea una vista a partir del archivo XML y lo guarda en floatingBubble como un FrameLayout y tambien asignamos a la variable bubbleTextView el texto que se encuentra dentro floatingBubble.

A continuación creamos una variable params a la que asignaremos un conjunto de parámetros necesarios para manejar la burbuja.

Estos parámetros definen el ancho y altura de la burbuja, también definen que la burbuja será una sobreposición mostrandola sobre otras aplicaciones y que los elementos táctiles pasen a otras ventanas por debajo de la burbuja.

Luego se establece la posisición inicial de la burbubuja arriba a ala izquierda de la pantalla y se añade con windowManager.addView la burbuja flotante con los parámetros establecidos anteriormente.

```
private fun initializeFloatingBubble() {
        floatingBubble = LayoutInflater.from(this).inflate(R.layout.floating_bubble, null) as FrameLayout
        bubbleTextView = floatingBubble.findViewById(R.id.bubble_text)

        val params = WindowManager.LayoutParams(
            WindowManager.LayoutParams.WRAP_CONTENT,
            WindowManager.LayoutParams.WRAP_CONTENT,
            WindowManager.LayoutParams.TYPE_APPLICATION_OVERLAY,
            WindowManager.LayoutParams.FLAG_NOT_FOCUSABLE or
            WindowManager.LayoutParams.FLAG_NOT_TOUCH_MODAL or
            WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS,
            PixelFormat.TRANSLUCENT
        )
        params.gravity = Gravity.TOP or Gravity.START
        params.x = 0
        params.y = 100

        windowManager.addView(floatingBubble, params)

        floatingBubble.setOnTouchListener { _, event ->
            when (event.action) {
                MotionEvent.ACTION_DOWN -> {
                    initialX = params.x
                    initialY = params.y
                    initialTouchX = event.rawX
                    initialTouchY = event.rawY
                    true
                }
                MotionEvent.ACTION_MOVE -> {
                    params.x = initialX + (event.rawX - initialTouchX).toInt()
                    params.y = initialY + (event.rawY - initialTouchY).toInt()
                    windowManager.updateViewLayout(floatingBubble, params)
                    true
                }
                else -> false
            }
        }
    }
```

En la última parte se usa un listener para gestionar los eventos táctiles de la burbuja cuando se detecta un primer toque sobre la burbuja y cuando se detecta movimiento tras el toque es decir se arrastra el toque sobre la burbuja.

7.4.2. Función onAccessibilityEvent()

En esta función se implementa el principal servicio de Accesibilidad de Android se sobrescribirá la función original Android onAccessibilityEvent() para adaptarla a las necesidades de la aplicación.

Se escuchan una serie de eventos que se recogen como event.eventType estos indican cambios en el contenido de la ventana actual ya sean el contenido de la ventana si se ha hecho scroll en la pantalla de forma que estos eventos reaccionen a cambios en la pantalla del usuario para notificar al servicio de Accesibilidad De Android de estos.

Luego se almacena el nodo raíz de la ventana actual en una variable y se comprueba que este no es nulo en ese caso utilizamos el método privado collectVisibleText() al que pasamos el nodo raíz y una variable tipo StringBuilder dónde almacenaremos el texto recogido por el nodo.

Este método privado collectVisibleText() coge el nodo raíz pasado por parámetro y comprueba si el nodo es visible para el usuario y si está dentro de los límites de la pantalla ya que se quiere que el análisis de texto sea en tiempo real y del texto que ve el usuario por pantalla en ese momento preciso. Si se cumplen estas condiciones se añade el texto de este nodo al StringBuilder pasado por parámetro.

A continuación se recorrerá el arbol de nodos hijos del nodo actual de manera que si el nodo actual tiene nodos hijo se invocará a esta misma función de manera se recorren todos los nodos hijos y se van añadiendo a la variable StringBuilder el texto de estos nodos si lo tuviese y si este se encuentra en los límites de la pantalla.

Para comprobar que el nodo se encuentra en pantalla se implementa una función isNodeWithinScreenBounds() que devolverá un valor true si los límites del nodo se encuentran dentro del tamaño de la pantalla.

```
override fun onAccessibilityEvent(event: AccessibilityEvent) {
        if (event.eventType == AccessibilityEvent.TYPE_WINDOW_CONTENT_CHANGED ||
            event.eventType == AccessibilityEvent.TYPE_VIEW_SCROLLED ||
            event.eventType == AccessibilityEvent.TYPE_VIEW_FOCUSED) {
            val rootNode = rootInActiveWindow
            if (rootNode != null) {
                val visibleText = StringBuilder()
                collectVisibleText(rootNode, visibleText)
                if (visibleText.isNotEmpty()) {
                    val text = visibleText.toString()
                    Log.d(TAG, "Captured␣Text:␣$text")
                    analyzeText(text)
                }
            }
        }
    }

    private fun collectVisibleText(node: AccessibilityNodeInfo, visibleText: StringBuilder) {
        if (node.isVisibleToUser && node.text != null && isNodeWithinScreenBounds(node)) {
            visibleText.append(node.text).append("\n")
        }
        for (i in 0 until node.childCount) {
```

```
            val childNode = node.getChild(i)
            if (childNode != null) {
                collectVisibleText(childNode, visibleText)
                childNode.recycle()
            }
        }
    }

    private fun isNodeWithinScreenBounds(node: AccessibilityNodeInfo): Boolean {
        val screenRect = Rect()
        node.getBoundsInScreen(screenRect)

        val screenSize = Point()
        windowManager.defaultDisplay.getSize(screenSize)

        return screenRect.intersect(0, 0, screenSize.x, screenSize.y)
    }
```

Por último se pasa el texto recogido de cada nodo por la funcion analyzeText() en esta funcion el texto será corregido de los caracteres nulos no imprimibles que son los caracteres de control de Unicode y con la expresión regular representamos el rango de caracteres de control y los eliminamos del texto sustituyéndolos por una cadena vacía ya que si no estos caracteres de control crean problemas al ser enviados al servidor.

Luego creamos una instancia de OkHttpClient y otra variable que indica que el cuerpo de la solicitud será de tipo JSON y se define el cuerpo de la solicitud en una variable que almacenará el RequestBody que enviará el texto corregido sin caracteres de control.

Haciendo uso de la función request.Builder() construimos el objeto request este objeto contiene la solicitud tipo post de HTTP con el cuerpo de la solicitud y se envía a la dirección del servidor almacenada en la constante.

La variable client que es un objeto tipo OkHttpClient() haciendo uso de la función newCall() a la que pasará como parámetro el cuerpo de la petición HTTP previamente definida.

```
private fun analyzeText(text: String) {
        val sanitizedText = text.replace("[\\u0000-\\u001F]".toRegex(), "")
        val client = OkHttpClient()
        val mediaType = "application/json".toMediaTypeOrNull()
        val body = RequestBody.create(mediaType, "{\"text\":\"$sanitizedText\"}")
        val request = Request.Builder()
            .url(API_URL)
            .post(body)
            .build()

        client.newCall(request).enqueue(object : Callback {
            override fun onFailure(call: Call, e: IOException) {
                Log.e(TAG, "API␣Request␣Failed", e)
            }

            override fun onResponse(call: Call, response: Response) {
                if (response.isSuccessful) {
                    val responseData = response.body?.string()
                    Log.d(TAG, "Text␣Analysis␣Response:␣$responseData")
                    responseData?.let {
                        val jsonResponse = JSONObject(it)
```

```
                    val compound = jsonResponse.getDouble("compound")
                    updateBubble(compound)
                    sendEventToReactNative("TextAnalysisEvent", it)
                }
            }
        }
    })
}
```

De cara a un control de errores se sobrescribe el metodo onFailure() y para recoger la respuesta del servidor se sobrescribe el método onResponse() que es llamado cuando la solicitud HTTP se completa con éxito.Se almacena la respuesta en una variable en caso de ser recibir la respuesta de manera correcta.En caso de no ser una respuesta nula se extrae el valor compound del objeto JSON recibido por el servidor.

Por ver de esta clase quedan las dos últimas funciones privadas una funcion updateBubble() a la que se le pasa como argumento el valor compound tipo double y se fuerza con el handler a que la actualización de la burbuja se haga en el hilo principal para conseguir el menor retardo posible y se implementa el calculo del color de la burbuja que recordemos cambiara con el valor polar recibido en compound a más rojo cuando el valor sea cercano a -1 y a más verde cuando sea cercano a +1.

```
private fun updateBubble(compound: Double) {
        handler.post {
            val green = (255 * ((compound + 1) / 2)).toInt()
            val red = (255 * (1 - (compound + 1) / 2)).toInt()
            val color = 0xFF000000.toInt() or (red shl 16) or (green shl 8)

            bubbleTextView.setBackgroundColor(color)
            bubbleTextView.text = String.format("%.2f", compound)
        }
    }

    private fun sendEventToReactNative(eventName: String, eventData: String) {
        val reactContext = (applicationContext as
        MainApplication).reactNativeHost.reactInstanceManager.currentReactContext
        reactContext?.getJSModule(DeviceEventManagerModule.RCTDeviceEventEmitter::class.java)
            ?.emit(eventName, eventData)
    }
```

Con la función sendEventToReactNative() podemos actualizar el valor de la burbuja flotante y el cambio de color y enviarlo a la parte React-Native de la aplicación.

Con esto concluímos el análisis de la clase CapturaAccessibilityService que es la clase en cargada de manejar todo el servicio de Accesibilidad de recoger el texto recorriendo los nodos de las ventanas de Android y enviarlo al servidor así como de recibir la respuesta gestionar el retorno del valor compound y sincronizarse con la parte React-Native del proyecto.

El resto de clases utilizadas tienen un papel menos importante y su código se pondrá en el manual de usuario y tambien se podrá consultar en el repositorio pero se considera que esta clase es la más importante para entender el desarrollo del proyecto.

7.5. Parte del Servidor. Análisis de Sentimientos.

En esta sección se muestra y explica la implementación de la parte del servidor, se comentaba al principio de esta memoria que nuestro servidor sería una pagina web llamada Replit.com pensada precisamente para ejecutar código en un navegador será en esta web donde ejecutaremos nuestra clase main.py implementada en lenguaje Python.

Se encarga de gestionar la solicitudes HTTP enviadas desde la aplicación en tiempo real y devolver el valor compound a la aplicación después de pasar el texto recibido por la clase SentimentIntensityAnalyzer() que es la clase principal de vaderSentiment una librería de análisis de sentimientos de textos que determina la polaridad positiva negativa o neutra de un texto siendo más precisa en textos cortos.

Primero se importan las librerías mecesarias para manejar solicitudes HTTP (request), crea la aplicación web (Flask) y para convertir datos en formato JSON (jsonfy).

Inicializamos CORS que es un mecanismo que permite que la infromación de una web sea solicitada en otro dominio que no sea de la web original es decir permite el intercambio entra aplicaciones web alojadas en dominios diferentes.

Habilitamos la recepción de solicitudes POST por la ruta /analyze y definimos una función analyze() que se lanza cuando recibe una solicitud en la ruta de /analyze. Se guarda en la variable data los datos del JSON recibidos en el POST luego se comprueba el si hay texto comprobando que text no sea nula.

Luego haciendo uso de la clase analyzer y la función polarity scores se calcula la carga sentimental del texto recibido y se devuelve el valor compound a la aplicación.

```
from flask import Flask, request, jsonify
from flask_cors import CORS
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

app = Flask(__name__)
CORS(app)
analyzer = SentimentIntensityAnalyzer()


@app.route('/analyze', methods=['POST'])
def analyze():
    data = request.get_json()
    if 'text' not in data:
        return jsonify({'error': 'No text provided'}), 400

    text = data['text']
    vs = analyzer.polarity_scores(text)

    # Imprimir el texto en la consola
    print(f'Texto: {text}')
    print(f'Resultados del analisis: {vs}')

    return jsonify({'compound': vs['compound']})
```

```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Capítulo 8

Diseño

Aunque esta aplicación se ejecutará principalmente en segundo plano si tiene algunos elementos gráficos como la pantalla principal de la aplicación donde podremos abrir las opciones de accesibilidad y la ya mencionada burbuja flotante que mostrará por pantalla el valor polar recibido desde el servidor para mostrar el valor polar que representa la carga sentimental del texto leído por el usuario.

Nuestra aplicación es importante que la burbuja sea fácil de mover por la pantalla en caso de que dificulta la lectura que su tamaño no fuera demasiado grande o molesto pero sí lo suficientemente grande para leer el texto e identificar fácilmente el color que toma según el valor polar recibido.

8.1. Pantalla Principal

La pantalla principal consiste en una sencilla interfaz que nos permite pulsando un botón acceder a las opciones de accesibilidad de nuestro dispositivo Android también al abrir esta pantalla principal se reiniciará la burbuja en caso de que haya habido algún fallo y se haya perdido o cerrado accidentalmente.

8.2. Burbuja Flotante

La burbuja flotante consiste en un pequeño rectángulo que irá cambiando de color según el valor polar recibido desde el servidor donde se interpretan los sentimientos de los textos leídos por pantalla el diseño de esta burbuja debía ser fácil de mover pequeño para no aplastar durante la lectura pero lo suficientemente grande para mostrar el valor polar por pantalla y que fuera reconocible y visible.

En las Figuras 7.1 y 7.3 podemos observar como la burbuja es desplazable en caso de tapar el texto. señalar también que se realizó esta prueba en un articulo de opinión en inglés

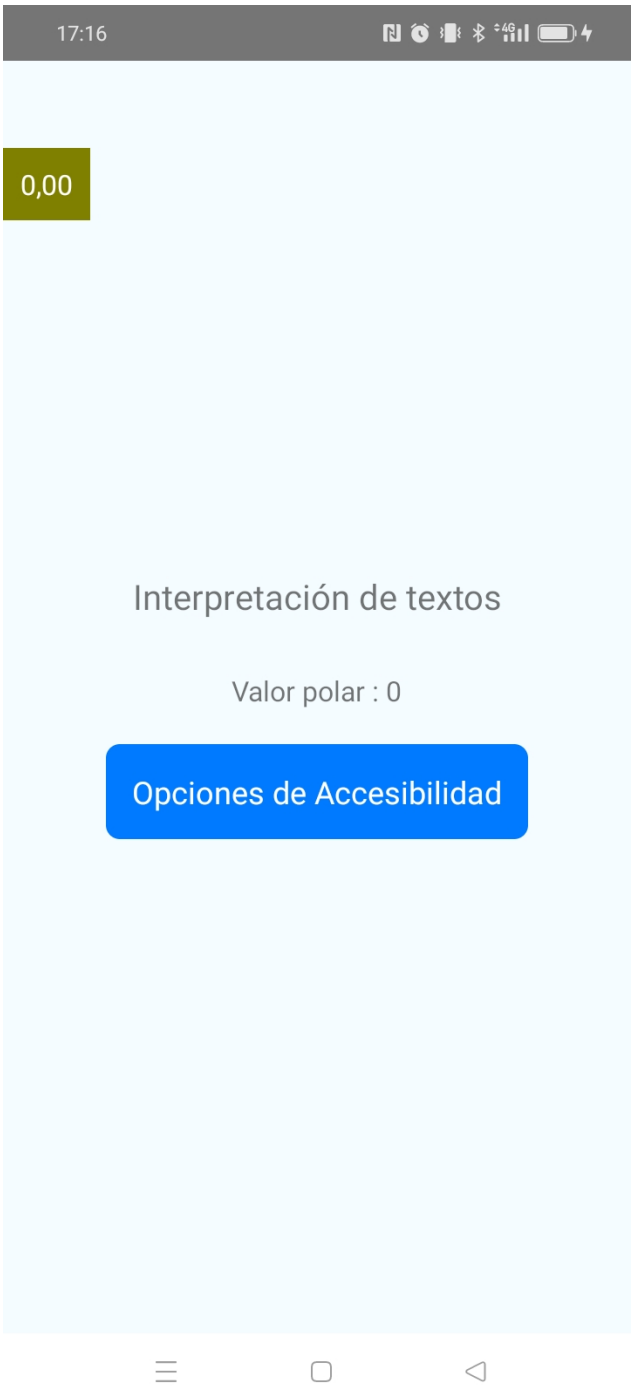

Ilustración 8.1: Pantalla Principal

para probar que efectivamente la librería VADER es más precisa a la hora de interpretar los sentimientos de los textos cuando es en inglés ya que el diccionario que usa para el español es una traducción directa y pierde presición.

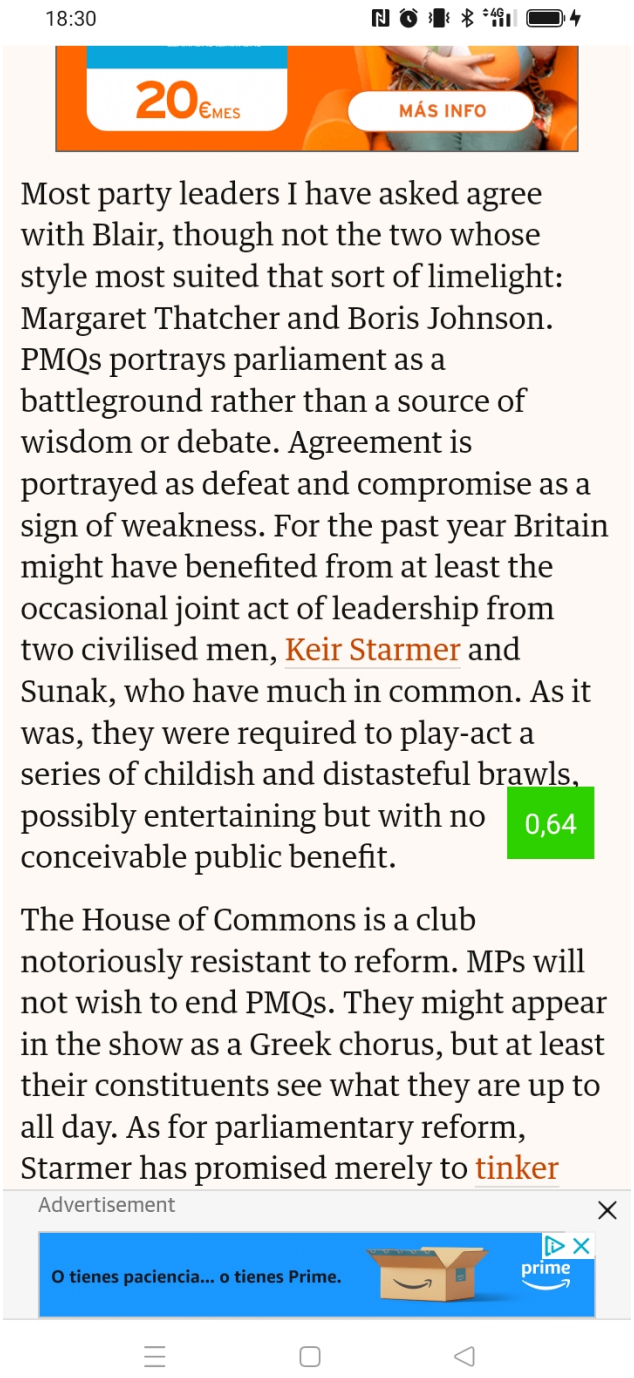

Ilustración 8.2: Burbuja Verde Abajo Derecha

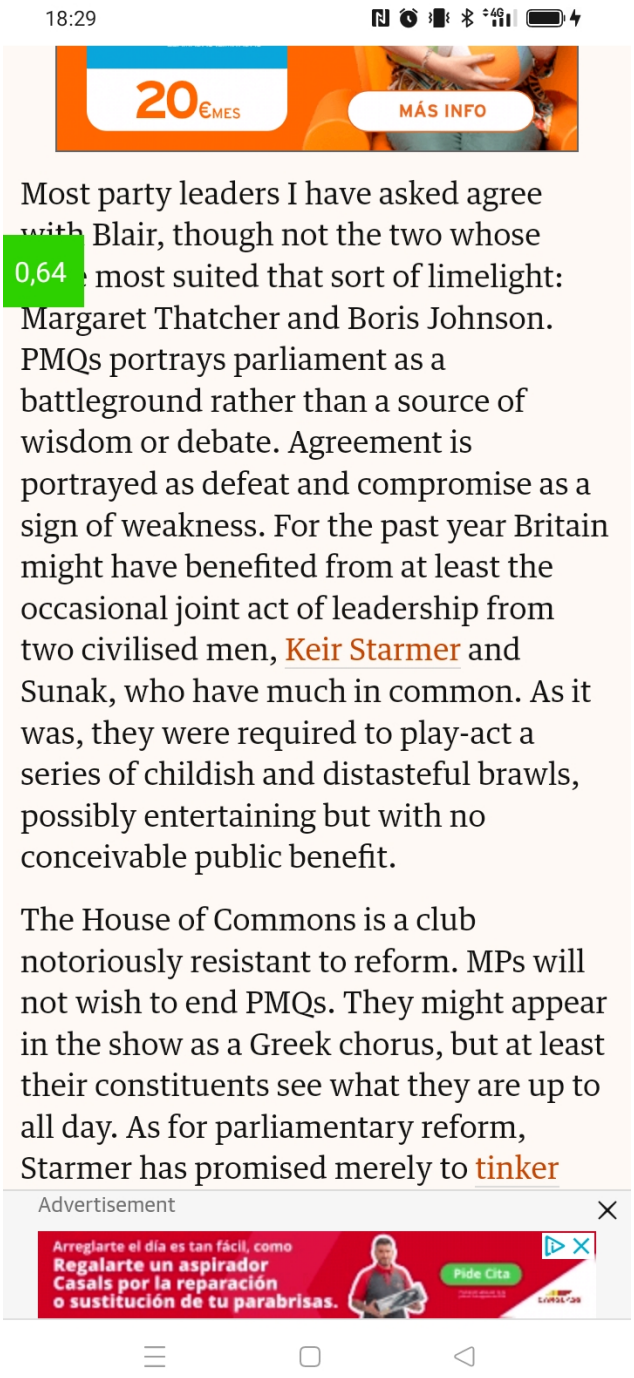

Ilustración 8.3: Burbuja Verda Arriba izquierda

Capítulo 9

Conclusiones y trabajo futuro

9.1. Conclusiones

Una vez terminado el desarrollo de este trabajo podemos decir que hemos cumplido la mayoría de los objetivos previamente establecidos.

Esta aplicación tal y como se diseñó en un principio se ejecuta en segundo plano leyendo el texto que ve el usuario por pantalla y enviándolo a un servidor donde se procesa este texto y utilizando la librería Vader se analiza la carga sentimental del texto leído por pantalla.

Este se devuelve a la aplicación un valor polar entre -1 y +1 el resultado de este valor polar será mostrado en una burbuja que se superpondrá al resto de elementos gráficos de la pantalla persiguiendo al usuario por todas las aplicaciones y ventanas y cambiando de color a verde cuando llega su valor más positivo y descendiendo por un gradiente hacia rojo cuando alcanza su valor más negativo asesorando así al usuario sobre la carga sentimental de los textos que está leyendo en tiempo real.

9.2. Propuestas para el futuro

Como contrapartida se estudió al inicio del proyecto la idea de realizar la aplicación también para el entorno iOS pero debido a la carga de trabajo no se ha realizado por lo que esto queda abierto a futuros proyectos.

Por otro lado también se podría mejorar el rendimiento de la aplicación ya que cuando el dispositivo tiene poca potencia o es antiguo puede dar problemas de rendimiento.

Como de retraso al actualizar la burbuja esto también se debe a que en el hilo principal de ejecución se actualiza más veces de lo necesaria tanto la burbuja como la lectura de texto quizá aquí se podría introducir a la implementación del código algún elemento que sincroniza la lectura para que no se realice con tanta frecuencia y mejore así su rendimiento.

Otro problema que nos encontramos es que la mayoría de las aplicaciones por ejemplo periódicos locales incluso nacionales utilizan para sus aplicaciones un web viewer es decir no muestran su periódico en la aplicación de forma nativa sino que simplemente carga la información desde la web y esto genera problemas a la hora de leer el texto que ve el usuario ya que cada web se diseña usando tecnologías diferentes aunque en la última versión de la aplicación que hemos creado esto está bastante corregido porque limitamos la lectura de texto a los límites de la pantalla de la ventana actual.

También se recomienda usar esta aplicación con un bloqueador de anuncios ya que estas páginas de sobre todo de periódicos tanto las web como las aplicaciones suelen estar fuertemente cargadas de publicidad y al recorrer el árbol de nodos la aplicación se encuentra con estos bloques de publicidad y aunque a veces los ignora con bastante eficacia otras veces no y en la pantalla del servidor se puede ver que no identifica esos textos aunque sí identifica los que hay alrededor.

Capítulo 10

Manual de usuario

De cara que a que en un futuro tanto profesores como compañeros como cualquiera que quiera usar este código fuente para mejorar o avanzar el proyecto queremos este manual de usuario a modo también de guía de instalación para instalar el proyecto React-Native en su equipo.

También para poder configurar correctamente el resto de tecnologías así como cargar el código del repositorio de forma que los archivos de configuración y el proyecto en sí queden de forma que el usuario o desarrollador pueda seguir más sencilla guía de pasos para poder ejecutar el proyecto en su equipo.

10.1. Instalación del proyecto.

Se expondrán aquí las diferentes tecnologías que hay que instalar en su equipo para poder ejecutar el proyecto y generar su propio apk de android con la aplicación lista para instalar.

10.1.1. Instalar Node.js

Para instalar Node.js solo hay que ir a la webnodejs.org y en la sección de descargas elegir el último instalador para su equipo.

Y solo hay que seguir las sencillas instrucciones del instalador.Y para comprobar que todo se ha instalado correctamente abrimos la consola de comandos y escribimos node -v. y si el resultado es como en la ilustración 9.1 es que se ha instalado bien Node.js

10.1.2. Instalar Android Studio

Se recomienda la instalación del IDE oficial de Android: Android Studio se puede descargar desde su página oficial. Esta recomendación viene dada en base a que no es solo un IDE

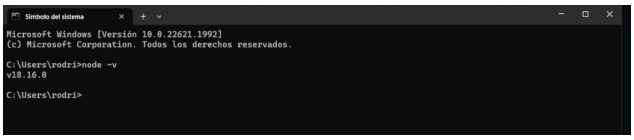

Ilustración 10.1: node -v

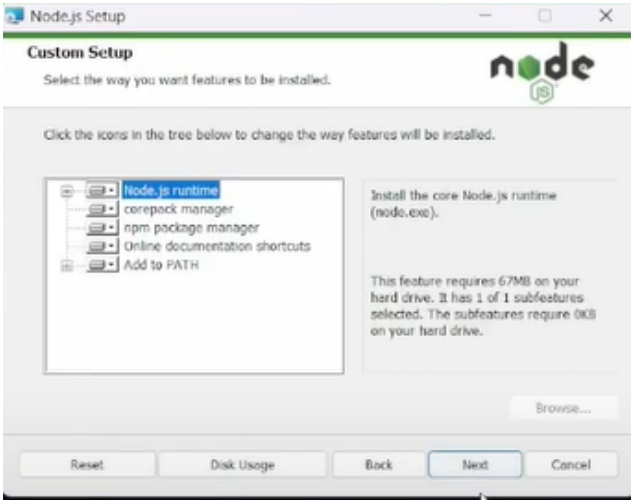

Ilustración 10.2: Instalador Node.js

si no que facilita enormemente la instalación del SDK de Android y un sistema de emulación de dispositivos android que serán utiles para para probar el dispositivo.

Esto llevará algo de tiempo ya que tanto para descargar el instalador como para instalar después el SDK son tareas pesadas y tardarán unos 20 minutos.

Cuando termine el instalador seremos capaces de abrir el cliente donde podremos acceder al Device Manager para crear nuestro emulador. Cuando se haya configurado el emulador debera ver una pantalla similar a la de la ilusustración 10.5.

A continuación se debe instalar el SDK de Android para ello debe volver a la página principal del cliente y se leccionar donde antes eligió Device Manager el SDK Manager. Eliges la versión 34 de Android y descargas el SDK u otros que te interesen.

Una vez terminado deberías poder abrir la consola de comandos y escribiendo el comando adb si se ha instalado correctamente te saldrá una lista de comandos si no dirá que no se reconoció el comando y lo que hay que hacer es agregar el comando a las variables de entorno. Hay que agregar tanto el contenido de /Android/Sdk como el de Android/Sdk/platform-tools escogiendo la ruta de donde se haya instalado el SDK.

10.1.3. Crear el proyecto React-Native

Se creará un proyecto base React-Native en una carpeta donde quiera tener instalado su proyecto se hará una instalación limpia vía línea de comandos como en la ilustración 10.8 .

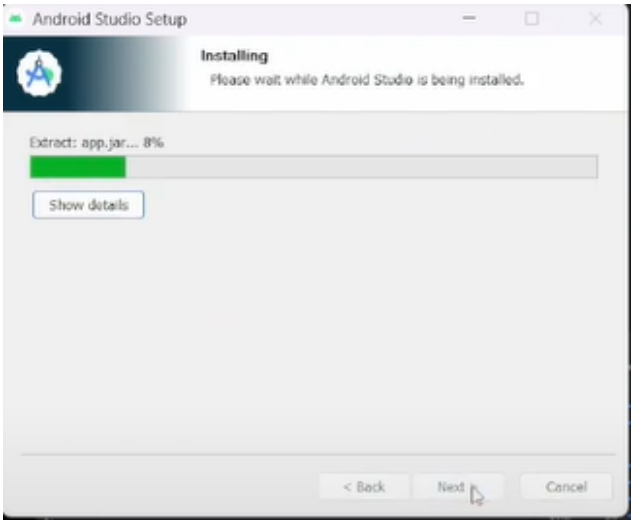

Ilustración 10.3: Android Studio Instaler

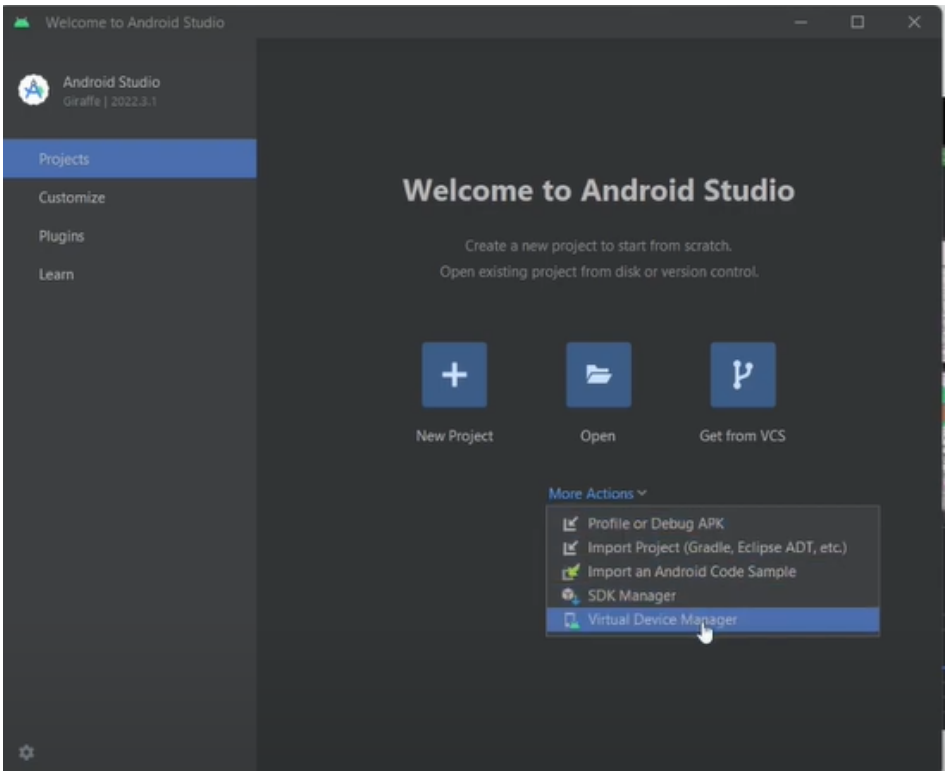

Ilustración 10.4: Cliente Android Studio

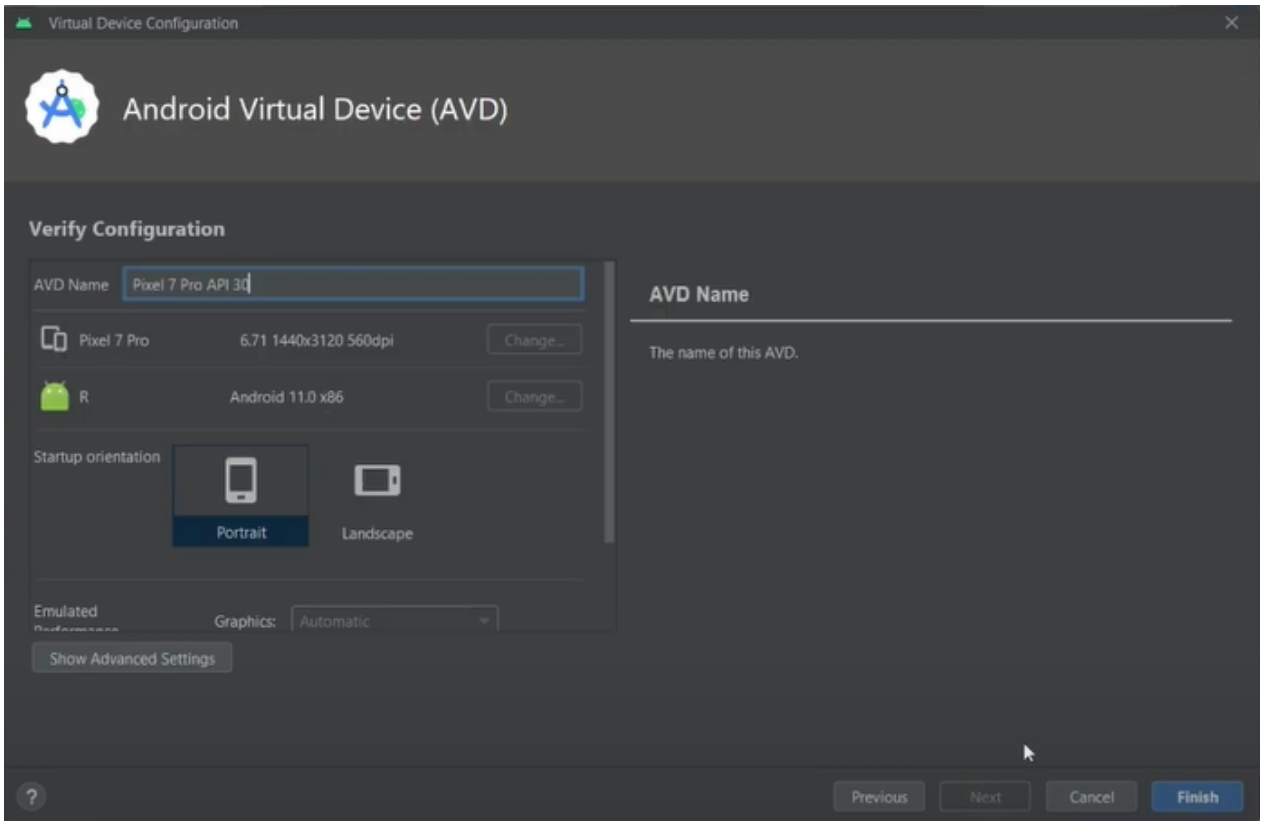

Ilustración 10.5: Emulador Configurado

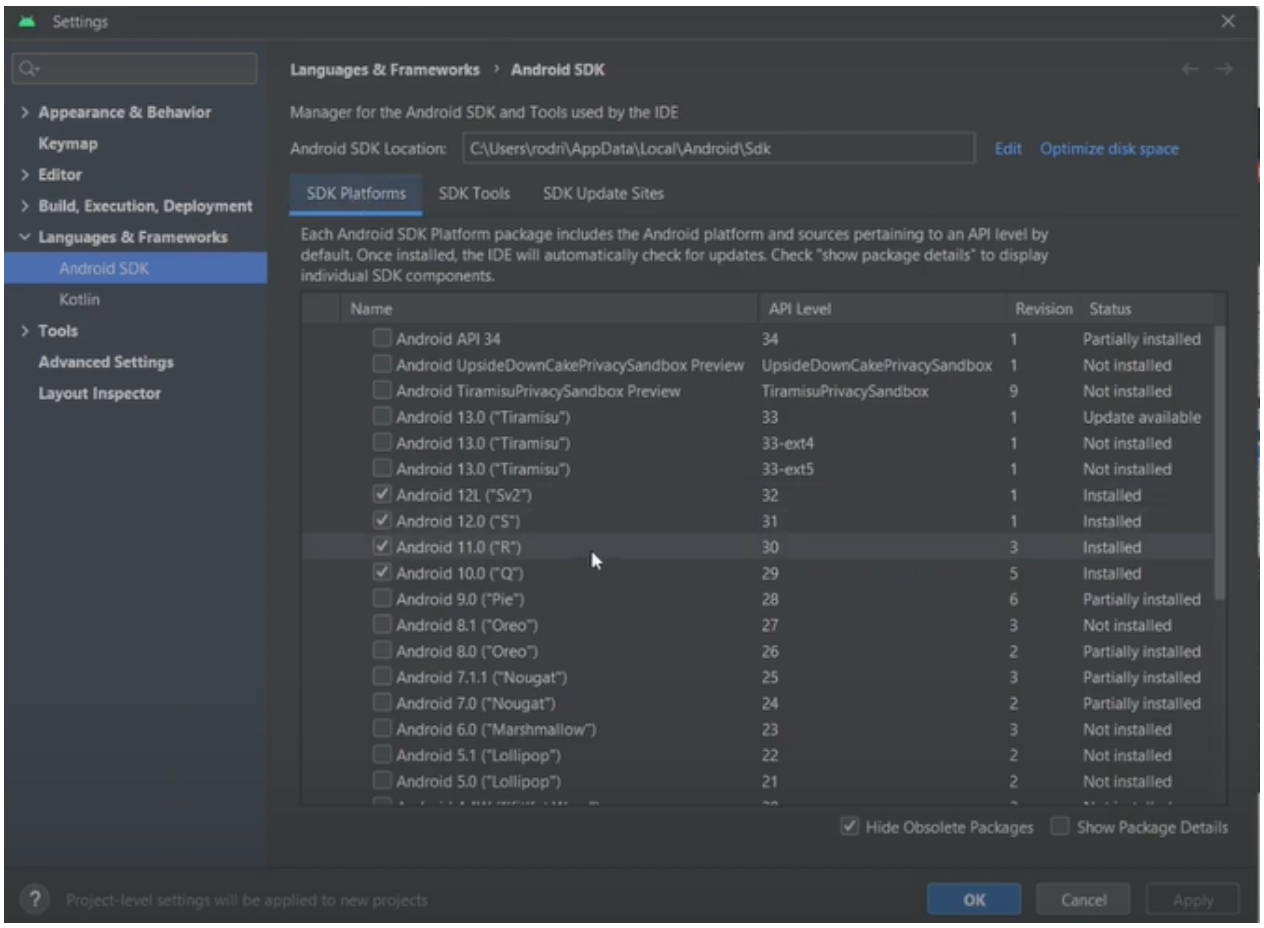

Ilustración 10.6: Configuración del SDK Manager

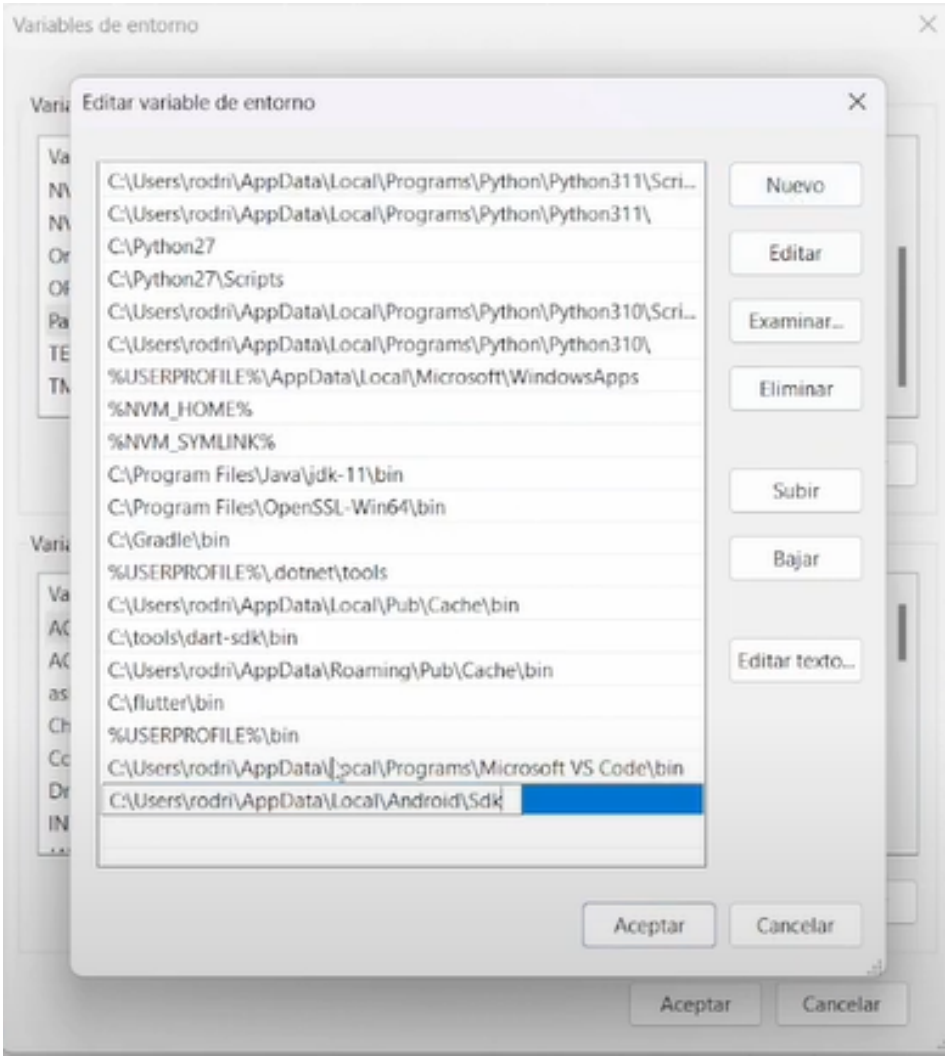

Ilustración 10.7: Añadir la variable de entorno del Sdk

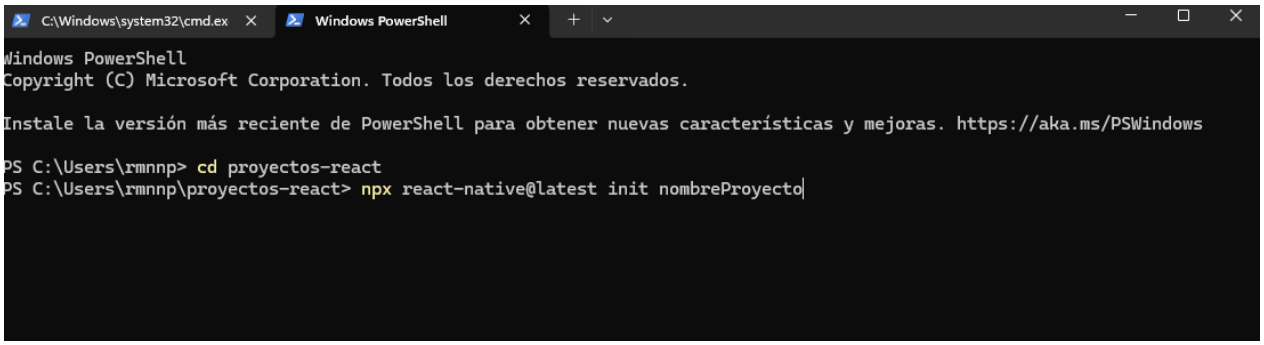

Ilustración 10.8: Crear proyecto React-native

Una vez se cree el proyecto tendremos el proyecto base de React-Native creado en nuestro directorio.

10.1.4. Copiar código fuente en el proyecto base.

Para descargar el código fuente del proyecto acceda a este repositorio

https://github.com/bpalmes/TFT-Sentiment

Una vez descargado hay que descomprimirlo y debería encontrarse una estructura como la que vemos en la figura 10.9 .Ena vez descomprimido copiamos la carpeta /android entera y la copiamos en nuestra instalación limpia de React-Native sustituyendo a la carpeta /android original.

Una vez hecho esto hay que copiar el resto de archivos que no están en carpetas de la raíz del proyecto sarcasmo menos los archivos .json que son de configuración los demás se copian a la raíz del proyecto limpio que creaste y sustituyes estos archivos por los copiados desde /sarcasmo.

Con estos cambios el proyecto deberia estar listo para funcionar ahora hay que abrir una consola de comandos moverse a la carpeta del proyecto y ejecutar npm run android como en la ilustracion 10.10 esto generará un apk y se instala en el dispositivo de destino que se haya seleccionado con Adroid Studio.

10.2. Otros extractos relevantes del código

En otras secciones de esta memoria se han explicado con detalle las clases principales como CapturaAccessibilityService.kt y la clase alojada en el servidor main.py que controla el analisis del sentimiento en los textos. que dan clases de menor importancia que no se han tratado y que se explicarán aquí brevemente.

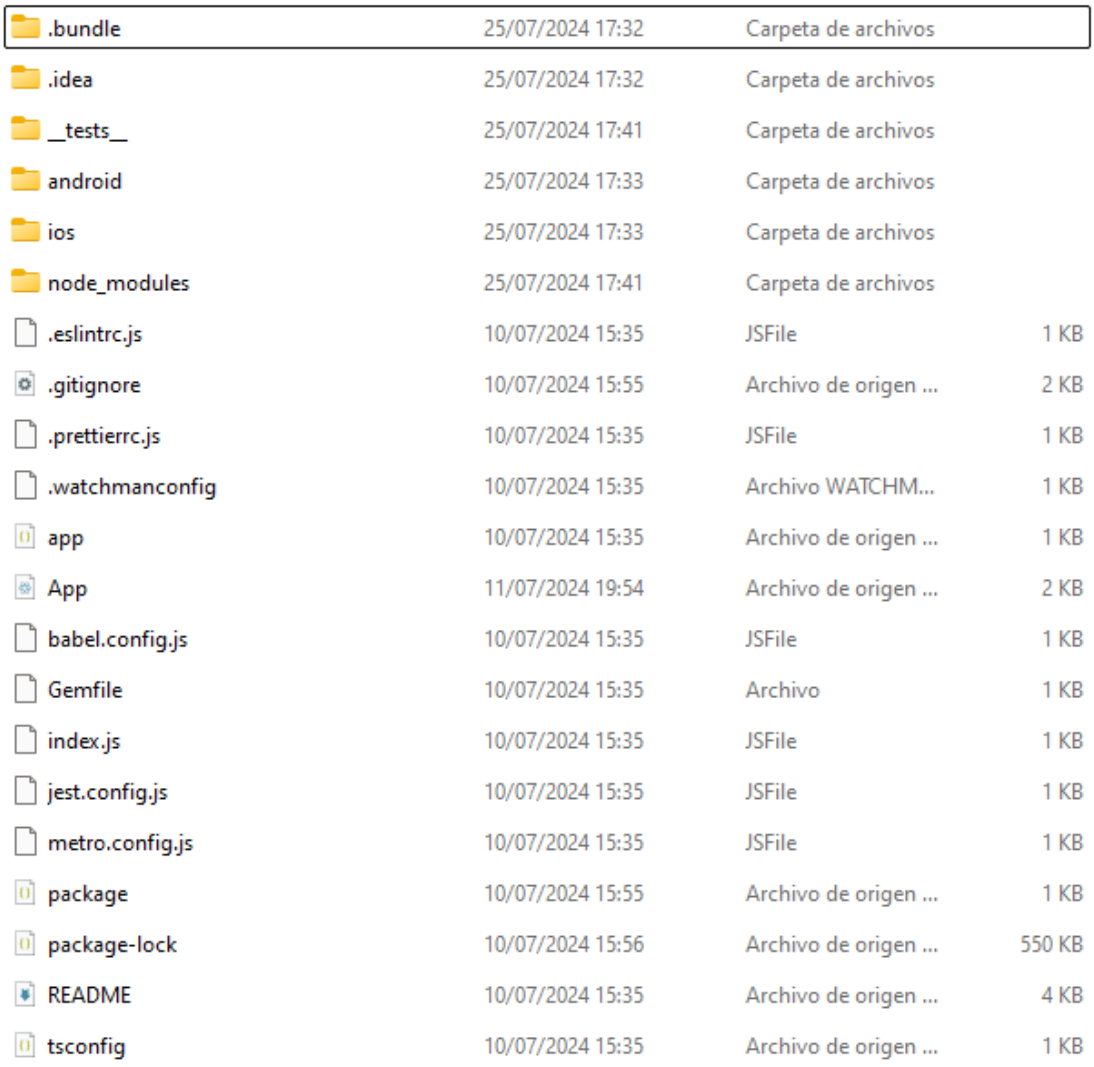

Ilustración 10.9: Estructura del proyecto

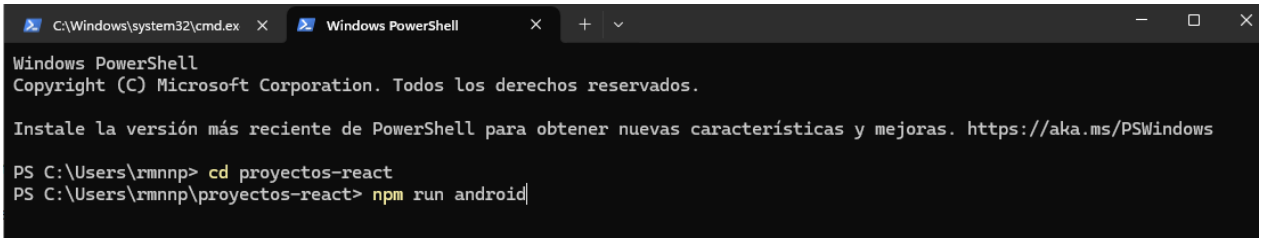

Ilustración 10.10: npm run android

10.2.1. EventoReact.kt

Esta clase se encuentra dentro del paquete com.sarcasmo y hace de puente entre eventos nativos de Kotlin o Java con el lado de React-Native

```
package com.sarcasmo

import com.facebook.react.bridge.ReactApplicationContext
import com.facebook.react.bridge.ReactContextBaseJavaModule
import com.facebook.react.bridge.ReactMethod
import com.facebook.react.modules.core.DeviceEventManagerModule

class EventoReact(private val reactContext: ReactApplicationContext) :
ReactContextBaseJavaModule(reactContext) {

    override fun getName(): String {
        return "EventoReact"
    }

    @ReactMethod
    fun sendEvent(eventName: String, eventData: String) {
        reactContext
            .getJSModule(DeviceEventManagerModule.RCTDeviceEventEmitter::class.java)
            .emit(eventName, eventData)
    }
}
```

10.2.2. EventoReactPackage.kt

Esta clase actúa como un paquete que agrupa módulos y view managers que se pueden usar en una aplicación React-Native.en concreto registra el módulo EventoReact para que en la parte de Javascript para que esté disponible.

```
package com.sarcasmo

import com.facebook.react.ReactPackage
import com.facebook.react.bridge.ReactApplicationContext
import com.facebook.react.uimanager.ViewManager
import com.facebook.react.bridge.NativeModule
import java.util.*

class EventoReactPackage : ReactPackage {
    override fun createNativeModules(reactContext: ReactApplicationContext): List<NativeModule> {
        return listOf(EventoReact(reactContext))
    }

    override fun createViewManagers(reactContext: ReactApplicationContext): List<ViewManager<*, *>> {
        return Collections.emptyList()
    }
}
```

10.2.3. MainActivity.kt

Esta clase configura la actividad principal de React-Native y permite que la aplicación se ejecute dentro de una actividad Android de manera que los componentes de React-Native funcionen correctamente.

```
package com.sarcasmo

import com.facebook.react.ReactActivity
import com.facebook.react.ReactActivityDelegate
import com.facebook.react.defaults.DefaultNewArchitectureEntryPoint.fabricEnabled
import com.facebook.react.defaults.DefaultReactActivityDelegate

class MainActivity : ReactActivity() {


  override fun getMainComponentName(): String = "sarcasmo"


  override fun createReactActivityDelegate(): ReactActivityDelegate =
      DefaultReactActivityDelegate(this, mainComponentName, fabricEnabled)
}
```

10.2.4. MainApplication.kt

Esta clase configura e inicializa la aplicación React-Native y sus paquetes en Android y habilita ciertas características según la configuracion de la compilación. En general se asegura de que la aplicación esté lista para ejecutar componentes de React-Native.

```
package com.sarcasmo

import android.app.Application
import com.facebook.react.PackageList
import com.facebook.react.ReactApplication
import com.facebook.react.ReactNativeHost
import com.facebook.react.ReactPackage
import com.facebook.soloader.SoLoader
import com.facebook.react.defaults.DefaultNewArchitectureEntryPoint.load
import com.facebook.react.defaults.DefaultReactNativeHost

class MainApplication : Application(), ReactApplication {

  override val reactNativeHost: ReactNativeHost =
      object : DefaultReactNativeHost(this) {
        override fun getPackages(): List<ReactPackage> =
            PackageList(this).packages.apply {

              add(EventoReactPackage())
            }

        override fun getJSMainModuleName(): String = "index"

        override fun getUseDeveloperSupport(): Boolean = BuildConfig.DEBUG

        override val isNewArchEnabled: Boolean = BuildConfig.IS_NEW_ARCHITECTURE_ENABLED
```

```
        override val isHermesEnabled: Boolean = BuildConfig.IS_HERMES_ENABLED
      }

  override fun onCreate() {
    super.onCreate()
    SoLoader.init(this, false)
    if (BuildConfig.IS_NEW_ARCHITECTURE_ENABLED) {
      load()
    }
  }
}
```

10.2.5. AndroidManifest.xml

Este archivo configura los permisos, define la MainAplication y declara el servicio de Accesibilidad. En general configura todos permisos que necesita la aplicación por ejemplo para capturar eventos de accesibilidad o acceder a internet.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
    <uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
    <uses-permission android:name="android.permission.BIND_ACCESSIBILITY_SERVICE"/>
    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:name=".MainApplication"
        android:label="@string/app_name"
        android:icon="@mipmap/ic_launcher"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:allowBackup="false"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|screenSize|
␣␣␣␣␣␣␣␣␣␣␣␣smallestScreenSize|uiMode"
            android:launchMode="singleTask"
            android:windowSoftInputMode="adjustResize"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service
            android:name=".CapturaAccessibilityService"
            android:permission="android.permission.BIND_ACCESSIBILITY_SERVICE"
            android:exported="true">
            <intent-filter>
                <action android:name="android.accessibilityservice.AccessibilityService" />
            </intent-filter>
            <meta-data
                android:name="android.accessibilityservice"
                android:resource="@xml/accessibility_service_config" />
        </service>
    </application>
</manifest>
```

10.2.6. App.tsx

Esta clase es la interfaz principal de la aplicación proporciona un boton para poder acceder a la configuración de accesibilidad del dispositivo del usuario y en general define los estilos y componentes React-Native para la interfaz.

```
import React, { useState, useEffect } from 'react';
import { View, Text, StyleSheet, TouchableOpacity, Linking,
NativeEventEmitter, NativeModules } from 'react-native';

const App = () => {
  const [charCount, setCharCount] = useState(0);

  useEffect(() => {
    const eventEmitter = new NativeEventEmitter(NativeModules.EventoReact);
    const eventListener = eventEmitter.addListener('TextAnalysisEvent', (eventData) => {
      const data = JSON.parse(eventData);
      setCharCount(data.compound);
    });

    return () => {
      eventListener.remove();
    };
  }, []);

  const openAccessibilitySettings = () => {
    Linking.openSettings();
  };

  return (
    <View style={styles.container}>
      <Text style={styles.header}>Interpretacion de textos</Text>
      <View style={styles.analysisContainer}>
        <Text style={styles.analysisText}>
          {charCount !== null ? `Valor polar : ${charCount}` : 'Leyendo...'}
        </Text>
      </View>
      <TouchableOpacity
        style={styles.button}
        onPress={openAccessibilitySettings}
      >
        <Text style={styles.buttonText}>Opciones de Accesibilidad</Text>
      </TouchableOpacity>
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: '#F5FCFF',
  },
  header: {
    fontSize: 20,
    textAlign: 'center',
    margin: 10,
```

```
  },
  analysisContainer: {
    justifyContent: 'center',
    alignItems: 'center',
    margin: 20,
  },
  analysisText: {
    fontSize: 16,
  },
  button: {
    backgroundColor: '#007AFF',
    padding: 15,
    borderRadius: 8,
  },
  buttonText: {
    color: '#fff',
    fontSize: 18,
  },
});

export default App;
```

Bibliografía

[1] Baron-Cohen, S. (2000). Theory of mind and autism: A fifteen year review. *Understanding Other Minds: Perspectives from Developmental Cognitive Neuroscience*, pages 3–20.

[2] Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media, Inc.

[3] Boduch, A. (2020). *React Native in Action*. Manning Publications.

[4] Bogdashina, O. (2003). *Sensory Perceptual Issues in Autism and Asperger Syndrome*. Jessica Kingsley Publishers.

[5] Chandna, P. and Chugh, R. (2018). Designing accessible mobile applications. *International Journal of Human-Computer Interaction*, 34:1324–1345.

[6] Dawson, G. (2012). Early behavioral intervention, brain plasticity, and the prevention of autism spectrum disorder. *Development and Psychopathology*, 24(2):493–507.

[7] de la Unión Europea, C. (2000). *Directiva 2000/78/EC del Consejo, que establece un marco general para la igualdad de trato en el empleo y la ocupación*. Diario Oficial de las Comunidades Europeas.

[8] Developers, A. (2016). *Android Accessibility Developer Guide*. Google.

[9] Europea, C. (2021). Propuesta de reglamento sobre la privacidad y las comunicaciones electrónicas (reglamento eprivacy). Pendiente de adopción.

[for Disease Control and Prevention] for Disease Control, C. and Prevention. Data & statistics on autism spectrum disorder. https://www.cdc.gov/ncbddd/autism/data.html. Accessed: 2023-07-26.

[11] Freeman, M. (2016). *Mobile Media and Applications, from Concept to Cash: Successful Service Creation and Launch*. Wiley.

[Goodman] Goodman, R. The overlap between autism and sensory processing disorder: What you need to know. https://childmind.org/article/autism-sensory-processing-disorder/. Accessed: 2023-07-26.

[Google] Google. Textview — android developers. Accessed: 2024-07-26.

[14] Google (2020). *Android Accessibility Overview*. Google.

[15] Google (n.d.). Android accessibility overview. https://developer.android.com/guide/topics/ui/accessibility. https://developer.android.com/guide/topics/ui/accessibility.

[16] Holmes, W. et al. (2019). Artificial intelligence and its application in education. *Learning, Media and Technology*, 44(2):97–112.

[17] Hunt, A. and Thomas, D. (2000). *The Pragmatic Programmer: From Journeyman to Master*. Addison-Wesley.

[18] Hutto, C. J. and Gilbert, E. (2014). Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth International Conference on Weblogs and Social Media (ICWSM-14)*.

[Inc.] Inc., A. Apple developer documentation. Accessed: 2024-07-26.

[20] Isaacson, W. (2011). *Steve Jobs*. Simon & Schuster.

[21] Ling, R. and Campbell, S. W. (2011). *The Reconstruction of Space and Time: Mobile Communication Practices*. Transaction Publishers.

[22] Loper, E. and Bird, S. (2002). Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, pages 63–70.

[23] Lord, C., Elsabbagh, M., Baird, G., and Veenstra-Vanderweele, J. (2018). Autism spectrum disorder. *The Lancet*, 392(10146):508–520.

[24] Maciaszek, L. A. (2005). *Requirements Analysis and System Design*. Addison-Wesley.

[25] Marx, L. (1964). *The Machine in the Garden: Technology and the Pastoral Ideal in America*. Oxford University Press.

[26] Myles, B. S. and Southwick, J. (2005). *Asperger Syndrome and Difficult Moments: Practical Solutions for Tantrums, Rage, and Meltdowns*. AAPC Publishing.

[27] Pang, B. and Lee, L. (2008). *Opinion Mining and Sentiment Analysis*. Now Publishers Inc.

[28] Parsons, S. and Cobb, S. (2011). State-of-the-art of virtual reality technologies for children on the autism spectrum. *European Journal of Special Needs Education*, 26(3):355–366.

[29] Priestley, N. (2019). *React Native in Action*. Manning Publications.

[30] Royce, W. (1970). Managing the development of large software systems. *Proceedings of IEEE WESCON*.

[31] Rubin, K. S. (2012). *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. Addison-Wesley.

[32] Schwaber, K. (2004). *The Enterprise and Scrum*. Microsoft Press.

[33] Schwaber, K. and Beedle, M. (2002). *Agile Software Development with Scrum*. Prentice Hall.

[Society] Society, A. How technology helps individuals with autism. https://www.autism-society.org/living-with-autism/how-technology-helps-individuals-with-autism/. Accessed: 2023-07-26.

[35] Sommerville, I. (2011). *Software Engineering*. Addison-Wesley.

[Speaks] Speaks, A. Treatment and therapies. https://www.autismspeaks.org/treatment. Accessed: 2023-07-26.

[37] Sutherland, J. and Sutherland, J. (2014). *Scrum: The Art of Doing Twice the Work in Half the Time*. Crown Business.

[38] Union, E. (2016). General data protection regulation.

[39] Williams, D. (2019). Accessible app design: A comprehensive guide. *Journal of Accessibility and Inclusion*, 6:45–67.

[40] y Consejo de la Unión Europea, P. E. (2016a). *Directiva (UE) 2016/2102 del Parlamento Europeo y del Consejo*. Diario Oficial de la Unión Europea.

[41] y Consejo de la Unión Europea, P. E. (2016b). *Reglamento General de Protección de Datos (GDPR)*. Diario Oficial de la Unión Europea.