Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

NorVID

Óscar González García, Alu0100954609

La Laguna, 21 de junio de 2021

D. **Alejandro Pérez Nava**, con N.I.F. 43821179-S profesor Asociado de Universidad adscrito al Departamento de Ingeniería informática de la Universidad de La Laguna, como tutor

D. **Fernando Andrés Pérez Nava**, con N.I.F. 42091420-V profesor Titular de Universidad adscrito al Departamento de Ingeniería informática de la Universidad de La Laguna, como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:

“NorVID”

ha sido realizada bajo su dirección por D. **Óscar González García**,

con N.I.F. 54109814-Z.

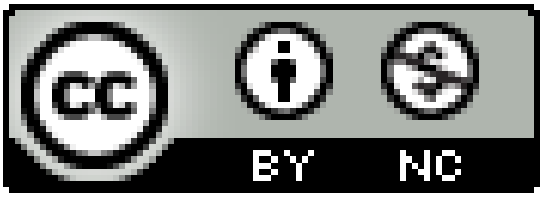
Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 21 de junio de 2021.

Agradecimientos

En primer lugar quiero agradecer a mi tutor Alejandro Pérez Nava ya que con sus conocimientos y experiencia en los distintos campos, ha facilitado y amenuado cada una de las etapas vividas/trabajadas durante el proyecto.

También quiero agradecer a mis amigos, familiares y novia que me han brindado apoyo, comprensión y cariño durante estos meses.

Y por último a la Universidad de La Laguna y los profesores asignados en la carrera de ingeniería informática por el aprendizaje y experiencia que me han dado durante todos estos años.



Resumen

En la actualidad, la sociedad se encuentra en una pandemia generada por la COVID-19 donde entran en rigor restricciones para poner freno a los contagios. Dichas normativas cambian cada cierto tiempo y son distintas dependiendo de la zona. Para la población, es complicado y confuso recordarlas todas sin recurrir a una fuente de información. Por ello, se plantea la aplicación para móviles la cual alerte de las normativas en vigor del COVID-19 por geolocalización sobre el lugar donde el usuario se encuentre, y además que permita identificar a través de un buscador, las restricciones de lugares diferentes. Por último, mencionar que las entidades que vertirán la información a la aplicación serán los ayuntamientos y gobiernos para que los ciudadanos tengan una vía rápida y fiable para corroborar las normativas.

El proyecto será realizado con Android Studio en Java y requerirá de una base de datos y sistema de autenticación de usuarios.

Palabras clave: COVID-19, restricción, geolocalización, Android Studio, Java, base de datos, autenticación

Abstract

Currently, society is in a pandemic generated by COVID-19 where restrictions are in place to put a stop to infections. These regulations change from time to time and are different depending on the area. For the population, it is difficult and confusing to remember them all without resorting to a source of information. For this reason, the application for mobile phones is proposed which alerts to the current regulations of COVID by geolocation on the place where the user is, and also that allows to identify through a search engine, the restrictions of different places. Finally, mention that the entities that will pour the information into the application will be the municipalities and governments so that citizens have a fast and reliable way to corroborate the regulations.

The project will be carried out with Android Studio in Java and will require a database and user authentication system.

Keywords: COVID-19, restriction, geolocation, Android Studio, Java, database, authentication

Índice general

Índice de tablas

Índice de figuras

1. Introducción

1.1. Antecedentes

El 31 de diciembre de 2019, la Organización Mundial de la Salud (OMS) recibió reportes de presencia de neumonía, de origen desconocido, en la ciudad de Wuhan, en China. Rápidamente, a principios de enero, las autoridades de este país identificaron la causa como coronavirus.

El coronavirus es un grupo de virus que causan enfermedades que van desde el resfriado común hasta otras más graves como neumonía, síndrome respiratorio de Oriente Medio y síndrome respiratorio agudo grave. La nueva cepa que se está extendiendo en la actualidad se denomina COVID-19. Es una enfermedad infecciosa causada por el coronavirus que se ha descubierto recientemente. Ambos eran desconocidos antes de que estallara el brote en Wuhan (China) en diciembre de 2019.

La enfermedad se ha expandido a diversos continentes tales como Asia, Europa y América. El 11 de marzo de 2020 fue reconocida como pandemia.

1.2. Problemática y estado del arte

Dado los acontecimientos explicados en el apartado anterior, entraron en vigor una serie de restricciones que irían cambiando dependiendo de las fases del estado de alarma en las que se encuentren las comunidades autónomas, provincias y municipios. Esto genera una serie de problemas:

- Las normativas cambian después de un tiempo determinado teniendo en cuenta la evolución de la enfermedad. Para la

población es confuso recordar todas las normativas implementadas, en adición que pueden ser confundidas con las establecidas con anterioridad.
- Por otro lado, las restricciones son distintas entre comunidades autónomas, provincias y municipios son mayoritariamente distintas. En caso de que se vaya a viajar entre zonas en España, se deberán conocer las normativas de estas.

Por dichos problemas, se han creado páginas web y aplicaciones donde muestran las normativas de provincias y comunidades autónomas para ayudar a la población, tales como ***QueVirus*** e ***Info Restricciones***. De las contempladas, la primera es la única que trabaja por geolocalización.

El proyecto desarrollado tiene como objetivo darle a las personas un recurso para solventar estos problemas de manera cómoda y rápida, que haga uso de la geolocalización y tenga un campo adicional para los municipios.

1.3. Objetivos

El proyecto presenta una serie de objetivos:

- Mantener informada a la población de las normativas en vigor.
- Facilitar el acceso a la información relacionada con las restricciones y centralizarla en una aplicación móvil.
- Otorgar a los ayuntamientos y gobiernos un canal específico para mantener a la población actualizada.
- Modificar las normativas de manera sencilla en un entorno amigable de comunidades autónomas, provincias y municipios.
- Acceder a las restricciones del lugar en el que el usuario se encuentre por geolocalización.

1.4. Resultados esperados

El resultado que se espera conseguir al completar el proyecto es la creación de una aplicación móvil para la solventación de los problemas mencionados en el apartado de “Problemática y estado del arte”.

En concreto esto implica:

- Creación de una estructura de la base de datos bien pensada para la aplicación.
- Creación de una pantalla principal que traiga las restricciones de la base de datos por geolocalización y por barras de búsqueda.
- Creación de un sistema de autenticación.
- Creación de un sistema de registro al cual se le asignen al usuario comunidades autónomas, provincias o municipios.
- Creación de una pantalla de inserción de restricciones.
- Creación de una pantalla de creación de restricciones.

A nivel personal, los resultados esperados son:

- Aprender el uso de la geolocalización para futuras aplicaciones o programas.
- Aprender a usar con soltura el software *Android Studio*.
- Ganar destreza a la hora del uso de peticiones a una base de datos.
- Ganar experiencia en el lenguaje de programación de *Java*, ya que es el primer contacto que tiene el redactor con él.
- Completar la aplicación móvil.

2. Softwares y herramientas utilizadas

2.1. Android Studio

Figura 1: Logo de Android Studio

Android Studio es el entorno de desarrollo integrado oficial para la plataforma Android. Fue anunciado el 16 de mayo de 2013 en la conferencia Google I/O, y reemplazó a Eclipse como el IDE oficial para el desarrollo de aplicaciones para Android.

El sistema de compilación de *Android Studio* está basado en *Gradle*, y el complemento de Android para Gradle agrega varias funciones que son específicas para la compilación de aplicaciones para Android. Si bien el complemento suele actualizarse en el paso de bloqueo con *Android Studio*, el complemento y el resto del sistema *Gradle* se pueden ejecutar independientemente de Android Studio y se pueden actualizar por separado.

Para el proyecto a desarrollar, se utilizó la versión “4.1.2” ya que es la versión más sólida, puesto que las más novedosas continuaban en fase de prueba. En cuanto a la versión de *Gradle*, fue la “6.5”.

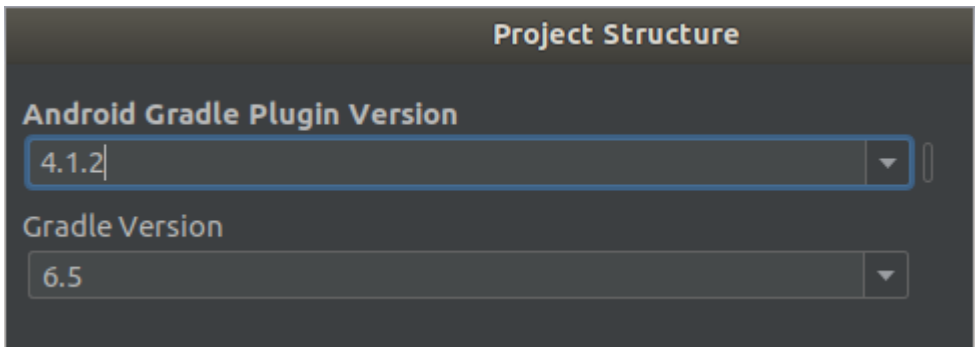


Figura 2: Versión usada de Android Studio y Gradle

2.2. Firebase

Figura 3: Logo de Firebase

Firebase de Google es una plataforma en la nube para el desarrollo de aplicaciones web y móviles. Está disponible para distintas plataformas (iOS, Android y web), y su función esencial es simplificar la creación y desarrollo de aplicaciones, procurando que el trabajo sea más rápido, pero sin renunciar a la calidad requerida. Sus herramientas son variadas y de fácil uso.

En los siguientes apartados se describirán los usados en la aplicación.

2.2.1. Authentication

Figura 4: Logo de Firebase Authentication

La mayoría de las apps requieren de la identificación de usuarios. Conocer la identidad permite que una aplicación guarde sus datos en la nube de forma segura y proporcione la misma experiencia personalizada en todos los dispositivos de este.

Firebase Authentication proporciona servicios de backend, SDK fáciles de usar y bibliotecas de IU ya elaboradas para autenticar a los usuarios en las aplicaciones. Admite la autenticación de diversas maneras, las principales son:

- mediante contraseñas.
- Números de teléfono.
- Proveedores de identidad federada populares, algunas son:
    - Google.
    - Facebook.
    - Twitter.

2.2.2. Cloud Firestore

Figura 5: Logo de Firestore

Cloud Firestore es una base de datos NoSQL alojada en la nube a la que pueden acceder las aplicaciones iOS, Android y Web directamente desde los SDK nativos. También está disponible en Node.js, Java, Python, Unity, C++ y Go, además de las API de REST y RPC.

A partir del modelo de datos NoSQL de Cloud Firestore, los datos se almacenan en documentos que contienen campos que se asignan a valores. Estos documentos se almacenan en colecciones, que son contenedores y que se pueden usar para organizar la información y compilar consultas. Los documentos admiten distintos tipos de datos, desde cadenas de texto y números simples, hasta objetos anidados complejos. También se pueden crear subcolecciones dentro de documentos y crear estructuras de datos jerárquicas que se ajustan a escala a medida que la base de datos crece.

2.2.3. Google Analytics

Figura 6: Logo de Google Analytics para Firebase

Google Analytics es una solución de análisis ilimitada y gratuita que ocupa un lugar central en Firebase. Analytics se integra a distintas funciones de Firebase y proporciona una capacidad ilimitada de generar informes. Estos permiten entender claramente cómo se comportan los usuarios para que se puedan tomar decisiones fundamentadas en relación con el marketing de las apps y las optimizaciones del rendimiento.

2.3. Github

Figura 7: Logo de GitHub

GitHub es un servicio basado en la nube que aloja un sistema de control de versiones llamado Git. Éste permite a los desarrolladores colaborar y realizar cambios en proyectos compartidos, a la vez que mantienen un seguimiento detallado de su progreso.

El control de versiones es un sistema que ayuda a rastrear y gestionar los cambios realizados en un archivo o conjunto de archivos. El sistema de control de versiones les permite analizar todos los cambios y revertirlos sin repercusiones si se comete un error, utilizado el código fuente para hacer modificaciones.

3. Desarrollo del proyecto

3.1. Backend

Como bien se mostró en los apartados anteriores, el backend utilizado fue *Firebase*. A continuación, se explicará la estructura de la base de datos utilizada. Como bien se dijo, la herramienta *Cloud Firestore*, contiene una serie de colecciones, los cuales son los contenedores de la información. Los creados fueron los siguientes:

Figura 8: Colecciones de la base de datos

- CCAA: Contiene todas las comunidades autónomas de España.
- Provincias: Contiene todas las provincias de España.
- Municipios: Contiene todos los municipios de España.
- Listas: Engloba *Arrays* de los municipios, provincias, comunidades autónomas y restricciones. Utilizados para los *AutoCompleteTextView* de *Android Studio*, los cuales son unas barras de búsqueda que completan el texto que se inserta por una de las palabras pasadas en el *Array*.
- Restricciones: Engloba las restricciones creadas por los usuarios.
- Users: Contiene los usuarios creados.

Los documentos de las colecciones “CCAA”, “Provincias” y “Municipios” contienen subcolecciones que engloban las restricciones existentes en el lugar correspondiente.

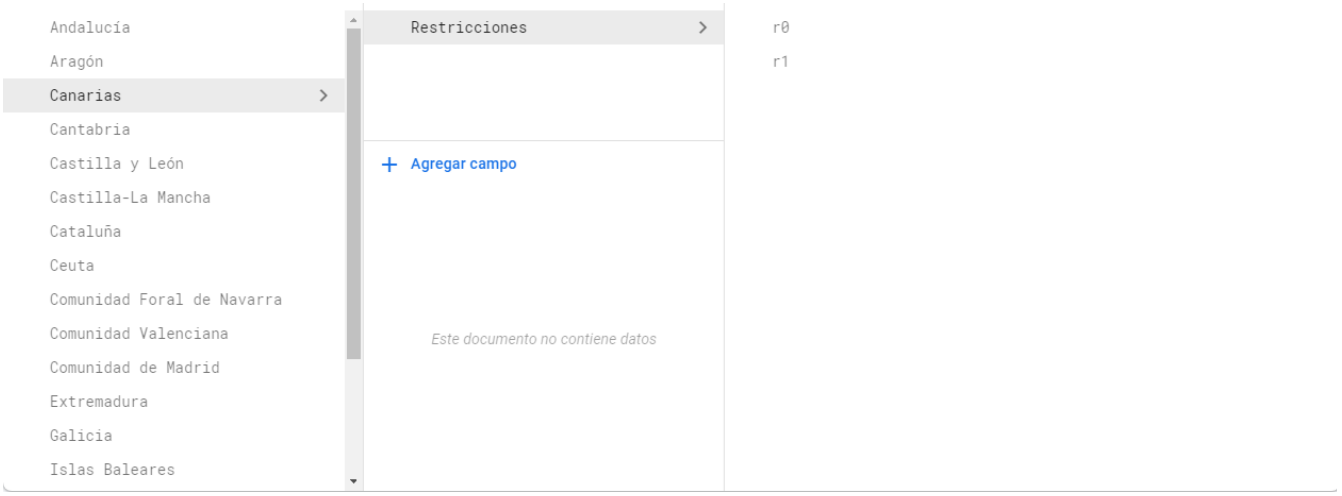


Figura 9: Subcolección de restricciones asignadas

En la figura anterior se puede comprobar que “Canarias” contiene una subcolección llamada “Restricciones”, y en este caso, contiene dos restricciones. Al acceder a una de ellas, se pueden ver los siguientes datos:

Figura 10: Restricción toque de queda de Canarias

“r0”, en este caso, corresponde con la restricción de toque de queda. Los datos almacenados en claves que son índices, son texto plano, que siempre será ese y nunca cambiará. Los que contienen la palabra “data” delante, son los almacenados por el usuario, los que varían.

Los documentos de listas están estructuradas de la siguiente manera:

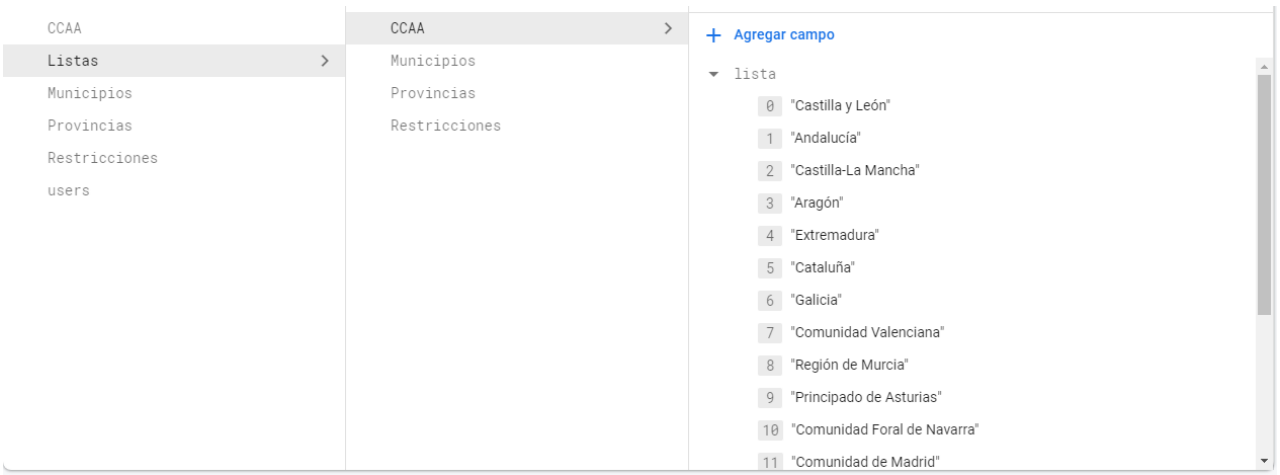


Figura 11: Documentos de "Listas"

En cuanto a los documentos de la colección "Restricciones":

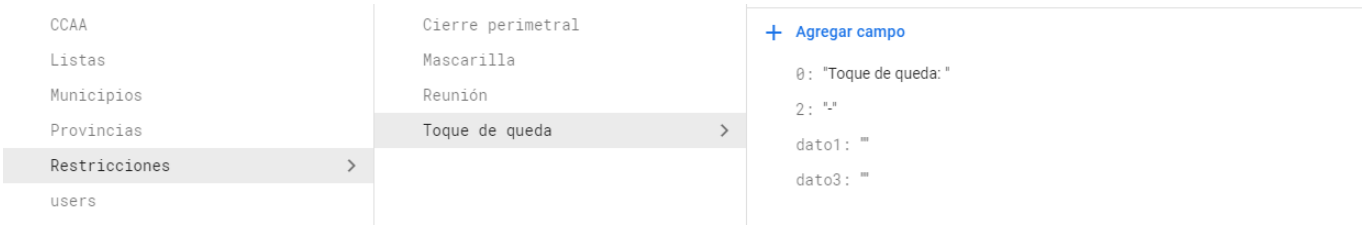


Figura 12: Documentos de "Restricciones"

Estos documentos almacenan las restricciones de una manera similar a las de la Figura[10] y la idea es la misma.

Por último, los documentos de "users", contiene los *email* de los usuarios, y los datos que contienen son 3 *Arrays*, "CCAA", "Prov" y "Mun". Estos contienen los nombres de las zonas en las cuales dicho usuario puede modificar las normativas.

Figura 13: Documentos de "users"

En cuanto a las demás herramientas del *backend*, el estilo de autenticación por el que se optó fue Usuario/Contraseña.

Con respecto a *Google Analytics*, sólo se colocó a modo de prueba puntos de control en las aperturas de cada pantalla. La vista de dicha analítica es la siguiente:

Clase de pantalla	% del total		Tiempo medio	
InsertRestrictions	40,04 %	↑ 26... %	0 mi...4 s	↑ 200,2 %
MainActivity	36,77 %	↓ 58,7 %	0 mi...1 s	↓ 58,0 %
loginActivity	17,53 %	-	0 mi...9 s	-
adminActivity	3,84 %	-	0 mi...0 s	-
NewRestriction	1,83 %	-	0 mi...0 s	-

Figura 14: Pantalla de interacción de usuarios por pantalla

La figura contiene las cinco pantallas de la aplicación, las cuales se explicarán más adelante, con una serie de columnas:

- "Clase de pantalla": Contiene el nombre de las pantallas.
- "% del total": Incluye los porcentajes de contención de los usuarios en dicha pantalla, cuanto mayor sea, más tiempo pasan los usuarios ahí.
- "Tiempo medio": Como su nombre indica, muestra el tiempo medio que el usuario se queda en la pantalla.

En este caso, tener en cuenta de que solo se recabaron datos de un usuario, porque la aplicación solo ha sido probada por el redactor del documento. La pantalla con los mayores valores es la de "InsertRestrictions" porque fue la que más problemas dio a nivel de código, junto a la principal.

Contiene más pantallas las cuales recaban otro tipo de datos, como la ubicación en la que se conectan los usuarios, visto desde un mapamundi. En este caso es irrelevante, dado que esta aplicación solo tiene sentido su uso en España.

Otra pantalla importante es la del número de usuarios que inician la aplicación por dia.

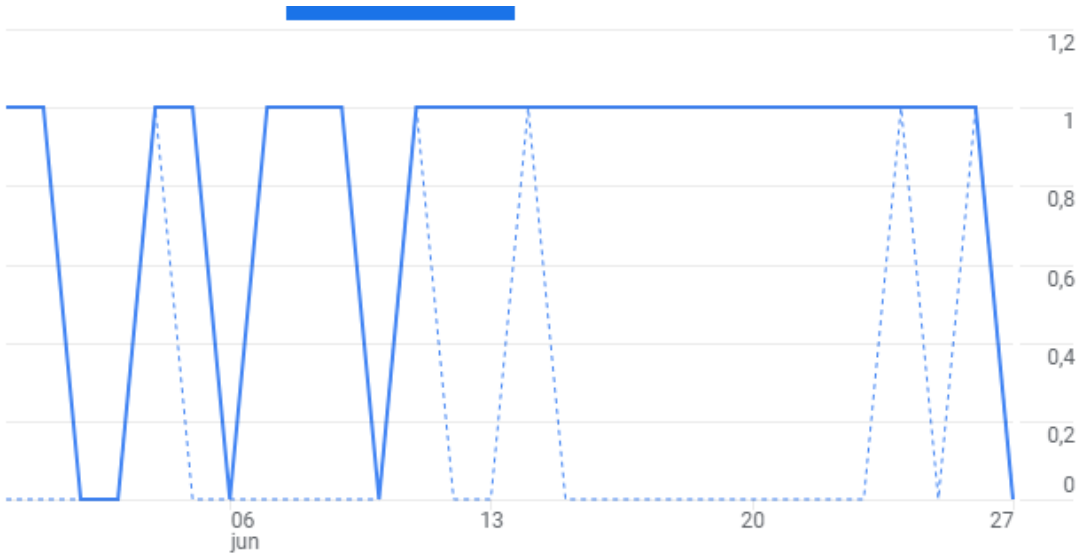


Figura 15: Pantalla de usuarios activos

3.2. Pantalla principal

La pantalla principal tiene como objetivo mostrar las normativas por geolocalización y que los usuarios puedan buscarlas por zonas.

Al entrar por primera vez en la aplicación, se pide permiso para acceder a la ubicación del dispositivo, si se niega, se le expulsa de la app. En caso de que acepte empezarán a cargar los datos.

Existen varios elementos en la vista que han sido colocados de manera estática. Estos elementos son los textos de “Comunidad autónoma”, “Provincia” y “Municipio”. También las barras de búsqueda, el botón, el menú de navegación y los textos de información. Estos últimos sólo aparecen en caso de que no haya ninguna normativa en vigor o no esté el gps activo, en caso contrario, es invisible. Lo demás es cargado dinámicamente.

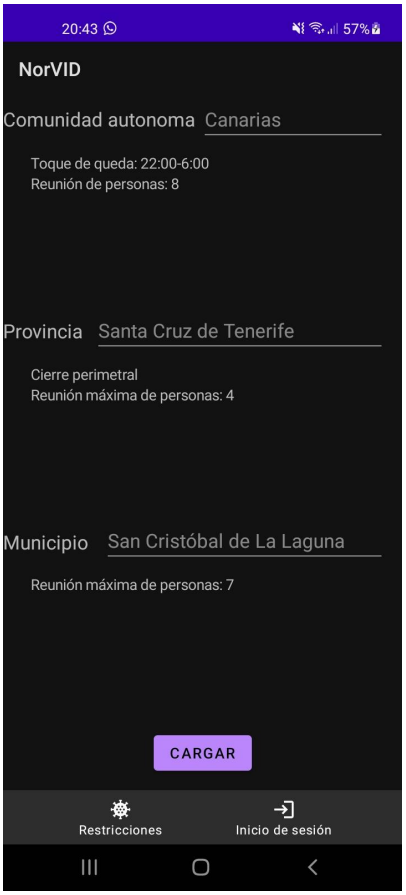


Figura 16: Pantalla principal

Al entrar a la aplicación y aceptar los permisos, ocurren una serie de eventos.

Primero se configura que se actualice la localización del usuario cada 5 segundos. Hasta la primera iteración, el botón de cargar estará inhabilitado, ya que si se pulsa, mandará una petición a la base de datos en busca de un documento vacío. Esto provocaría un error y cerraría la aplicación. Dentro del bucle se comprueba si los campos de texto del buscador están vacíos, si es así, actualiza las 3 variables para la búsqueda de información en la base de datos (comunidad autónoma, provincia y municipio). Aparte de lo ya explicado, actualiza los *hints* de los buscadores al valor de dichas variables.

Acto seguido, si es la primera iteración, hace 3 peticiones a la base de datos por cada apartado (comunidad autónoma, provincia y municipio). El código se puede ver aquí. Primero, será necesario el uso de 3 *Arrays* para almacenar los *ids* de los elementos creados de manera dinámica. Antes de las peticiones, se eliminan de la vista los elementos con cuyos *ids* estén en los *Arrays* y después son vaciados

para ser rellenados con los datos nuevos. Al realizar las llamadas a la base de datos, se recorre cada restricción en el lugar correspondiente y dentro de cada una de ellas, sus campos. Se crea un elemento de texto por cada campo y se van colocando de manera ordenada en la vista. Ya por último se guarda el *id* asignado. En caso de que no exista ninguna restricción, se mostrará un mensaje en el que pone "Ninguna restricción activa".

Lo siguiente que se hará es preparar las 3 barras de búsqueda con el código alojado aquí. Primero realiza una petición a la base de datos a su lista correspondiente, se realiza una conversión de *ArrayList* a *String[]* para poder insertarlo en el elemento. Al final se crea un *Listener* para cuando se haga clic en una de las opciones, se guarde el valor en la variable correspondiente para la búsqueda de la información. El resultado es el siguiente:

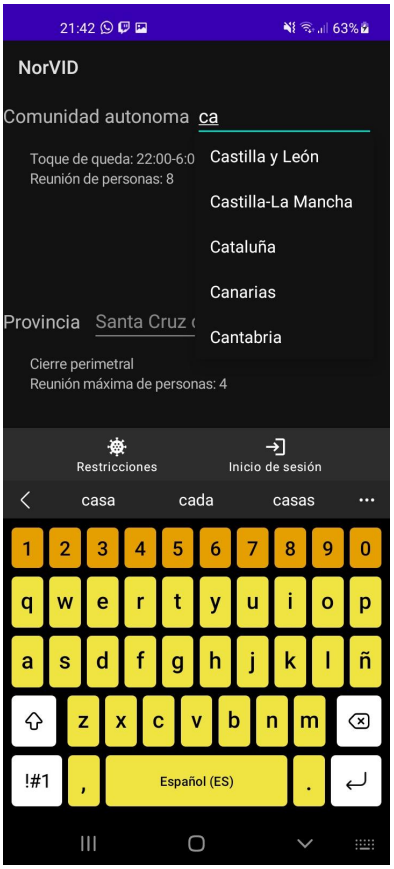


Figura 17: Barras de búsqueda

Por último, se le asigna al botón de cargar las 3 peticiones mencionadas anteriormente, volviendo a llamar a la misma función.

La barra de navegación, es visible tanto en esta pantalla como en la de inicio de sesión, y su funcionalidad es navegar entre ellas. El código

es simple, solamente realiza un redirect cuando se pulsa el botón de “Inicio de sesión”.

3.3. Pantalla de inicio de sesión

El objetivo principal de esta pantalla, es como el de todos los inicio de sesión, autenticar a los usuarios que tengan una cuenta creada.

La plantilla es estática completamente, tiene un campo para email y otro para contraseña, a la vez que un botón que desencadena el inicio de sesión.

Figura 18: Pantalla de inicio de sesión

La función usada para la autenticación se puede ver aquí. Al pulsar el botón, lo primero que se hará es comprobar que los campos de usuario y contraseña estén rellenos. Si no es así mandara un error a través de un *popup* con el mensaje: “Rellene los campos de correo electrónico y contraseña”.

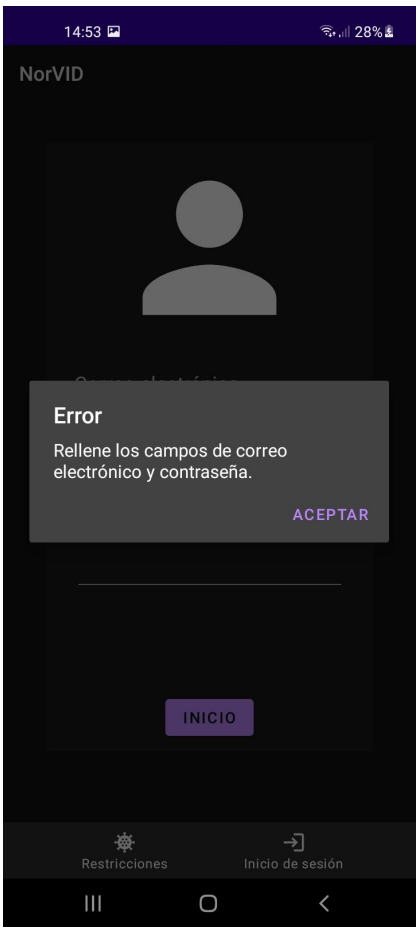


Figura 19: error: Rellene los campos

En caso de que el usuario haya insertado los campos se realizará la autenticación, sacándoles el texto. Dicha función lanza un evento cuando se completa y allí se comprueba si se realizó la operación correctamente. Si no es válida, quiere decir que el correo y la contraseña no pertenecen a ninguna cuenta, por lo que se lanza un error de la misma forma que el anterior.

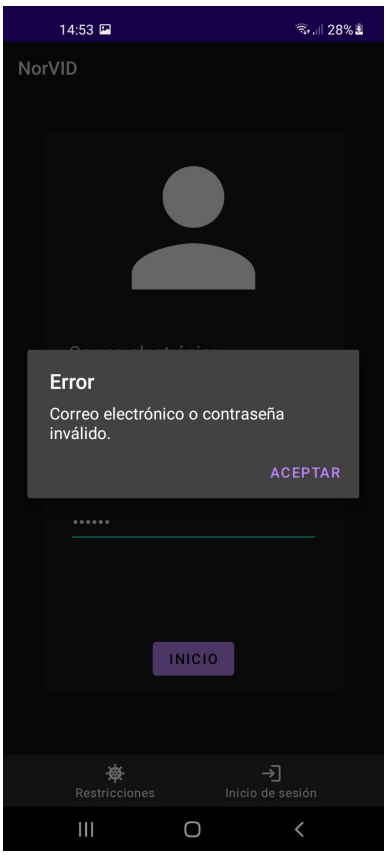


Figura 20: error: email o contraseña inválido

En caso de que coincidan con las de algún usuario se hace una comprobación. Si el email coincide con el del administrador, se redirige a la "pantalla del administrador", en caso contrario a la "pantalla de inserción de restricciones". En este último caso, se manda también como campo extra el correo del usuario, para poderlos diferenciar.

Cabe destacar un detalle, y es que desde el comienzo de la operación hasta el final, se inutiliza el botón de "Inicio". Si la conexión a internet es mala, la petición de autenticación puede durar varios segundos. Si se deja el botón activo, el usuario puede volverlo a pulsar realizando la misma operación varias veces añadiendo carga al programa y al *backend* sin ningún beneficio.

En cuanto al menú de navegación, el funcionamiento es el mismo que el de la "pantalla principal", pero esta vez te redirecciona a ella si se pulsa el botón "Restricciones".

3.4. Pantalla de administrador

La pantalla del administrador tiene como objetivo crear cuentas para los ayuntamientos y gobiernos, asignándoles una serie de zonas para que puedan modificar las normativas.

Toda la vista es estática, menos las comunidades autónomas, provincias y municipios que se vayan seleccionando.

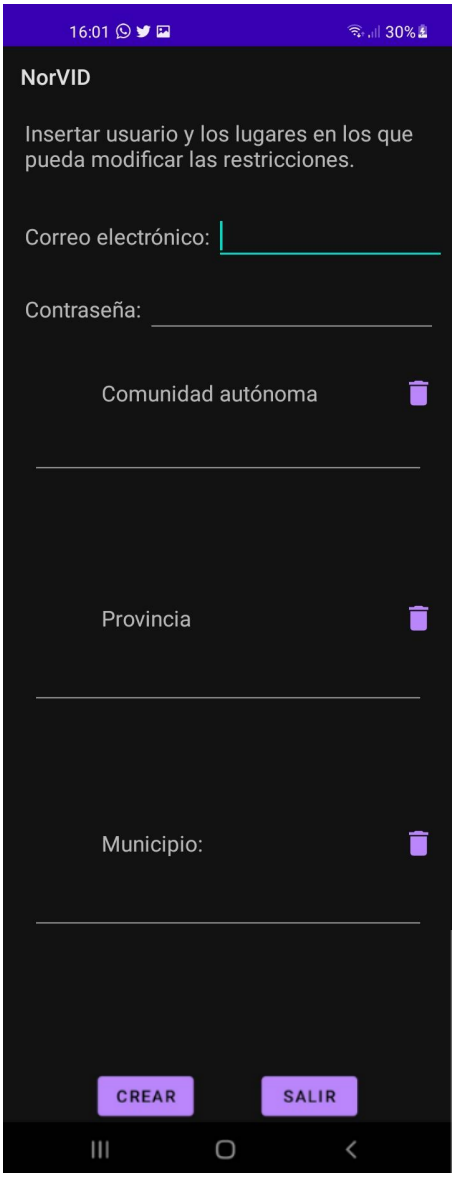


Figura 21: Pantalla de administrador

Al abrir esta pantalla, se inicia una serie de eventos.

Existen 3 barras de búsqueda que, al igual que en “Pantalla principal”, sirven para seleccionar las zonas. Sigue la misma mecánica, solo que en el Listener de cuando se pulsa una de las opciones, en este caso, se crea un texto que contendrá el lugar seleccionado, se añade el *id* del elemento creado en un *Array* y se colocará debajo de este, la barra de búsqueda. De esta manera se irán seleccionando las comunidades, provincias y municipios para el usuario. El código se puede ver en aquí.

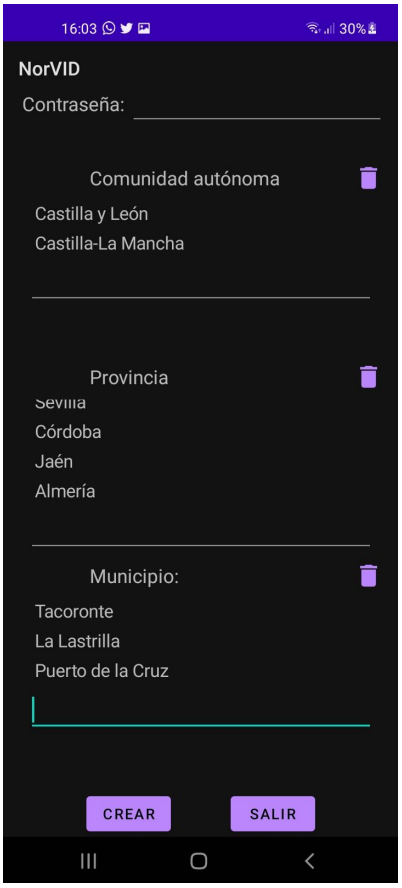


Figura 22: Barras de búsqueda del administrador

Existen a su vez, 3 botones de borrado para cada apartado, lo único que hacen es eliminar todos los elementos que contengan los *ids* de su respectivo apartado, vacía el *Array* y colocará arriba otra vez la barra de búsqueda. El código es visible aquí.

Son visibles 2 botones más en la parte inferior:

- “Salir”: Simplemente se cierra la sesión y se realiza una redirección a la “pantalla principal”.
- “Crear”: Primero, comprueba si el campo de email y contraseña están rellenos, en caso de que no, como en la “pantalla de inicio

de sesión", sale el mismo *popup* de error de "Rellene los campos de correo electrónico y contraseña". Si están insertados, se llama a una función para la creación de la cuenta, si esta no se completa con éxito, salta el siguiente error:

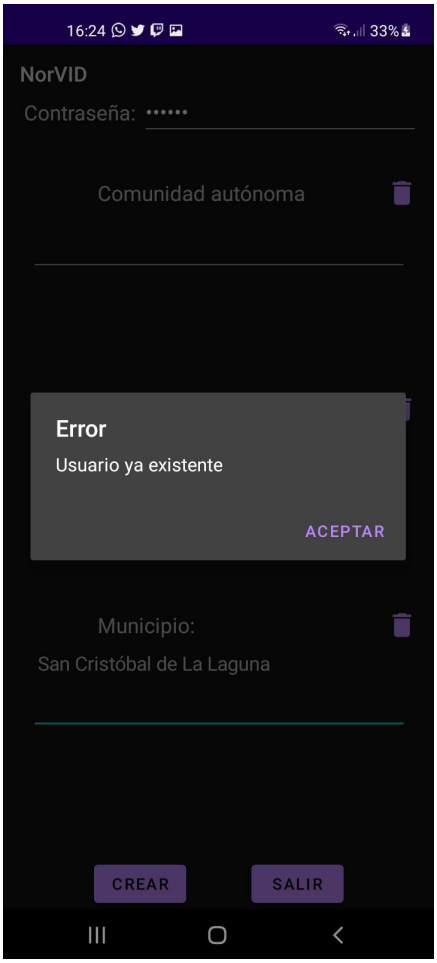


Figura 23: error: Cuenta ya existente

En caso de que esté todo en orden, a partir de los *ids* de los *Arrays*, se busca el elemento en la vista y se le saca el texto. Se crean 3 vectores de textos, para las comunidades autónomas, provincias y municipios. y se crea en la base de datos un documento cuyo título es el email del usuario creado, y se inserta dentro los 3 *Arrays*. Una vez hecho esto, se muestra un mensaje por pantalla, ya que no existe otra manera de alertar al administrador. El código se puede ver en aquí.

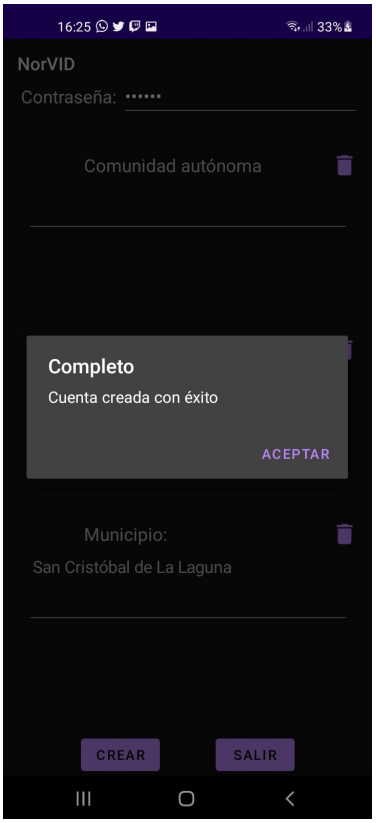


Figura 24: Cuenta creada

3.5. Pantalla de inserción de restricciones

La funcionalidad principal de esta pantalla, es la modificación de las restricciones por parte del usuario.

La vista es completamente dinámica, menos los dos botones de “Salir” y “Guardar”. Lo demás es generado a partir de la información del usuario autenticado. Se crean apartados para cada lugar y se traen las restricciones pertenecientes a cada una de dichas zonas. Después se le da la libertad al usuario para que las modifique.

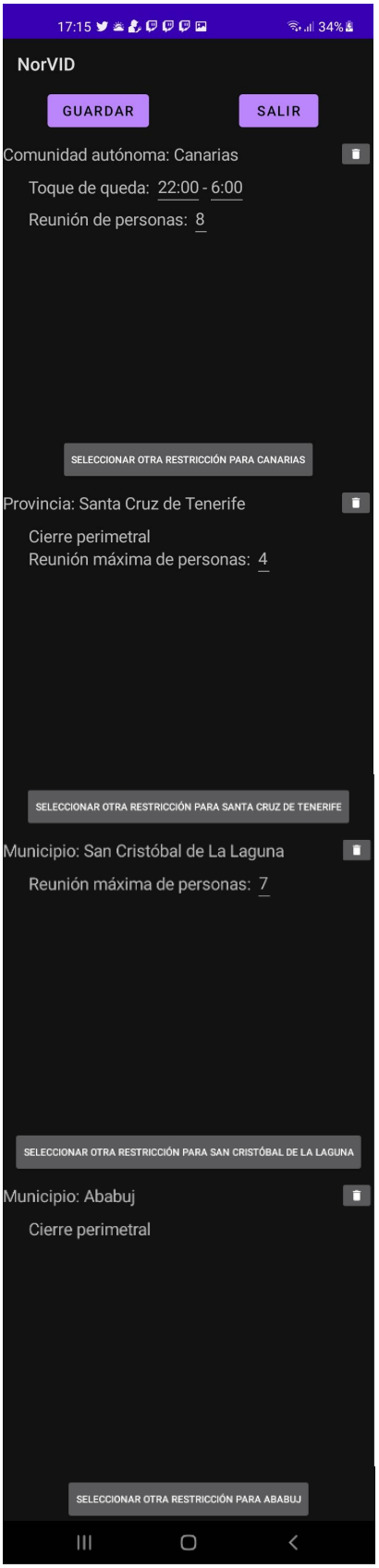


Figura 25: Pantalla de inserción de restricciones

Al entrar a la pantalla, se llevan a cabo una serie de eventos.

Primero se genera todo la vista a través del código escrito aquí. Se realiza una llamada a la base de datos, y al tener el email del usuario autenticado, se pueden traer las zonas en las que se le permite modificar. Con los datos conseguidos, se realizan 3 bucles cuya funcionalidad será crear cada apartado. Los 3 bucles funcionan igual, pero uno recorre "CCAA", otro "Prov" y el último "Mun", que son los valores *Array* sacados de la base de datos. Dentro de cada bucle, se crean los siguientes elementos:

- Texto con "Comunidad autónoma: ", "Provincia: " o "Municipio: ".
- Texto con el nombre correspondiente del lugar.
- Un botón que será usado para borrado.
- Un *ScrollView* y *ConstraintLayout*, que permite la inserción de elementos en él dentro de una altura y anchura determinadas. En caso de que se superen, permite hacer *Scroll* tanto en vertical como en horizontal. Los *ids* de los ConstraintLayout son guardados.
- Un botón para añadir más restricciones.
- Dentro del *ScrollView* se insertarán las restricciones ya existentes de ese lugar. Para ello, realizará otra petición a la base de datos para traerlas. Como se muestran en la "pantalla principal" se mostrarán aquí, con un cambio. Los valores sacados de una clave que empiece por "data", serán editables, ya que estos son los valores cambiantes de la restricción.

Hay que destacar que todos los *ids* de los elementos que forman las restricciones, serán guardados en un ArrayList<ArrayList<ArrayList<Integer>>>. Se utiliza dicha jerarquía para guardar de cada zona, cada restricción y de cada normativa, los *ids* de cada elemento que la compone.

Esto último, tiene un problema. Dado que no se pueden hacer peticiones síncronas a *Cloud Firestore*, el *ArrayList* que se comentó anteriormente se rellenaba bajo condición de carrera, la petición que antes acabara, metía los *ids*. La vista se mostraba bien, pero según la lógica del vector, las restricciones de un lugar podían llegar a pertenecer a otro. Esto genera problemas a la hora del borrado y del

guardado. Ya que si no se solucionaba, al pulsar el botón de borrado de Canarias, podrían eliminarse las restricciones de Santa Cruz de Tenerife en la vista y en el *ArrayList*. Al igual que si se guardaba, no se sabía donde se guardaría cada restricción.

Para solucionarlo se creó una función de restructuración del *ArrayList*, cuyo código se puede ver aquí. Al tener los *ids* de los *ConstraintLayout* donde estaban alojadas las normativas. Se fue recorriendo y comprobando que elementos son hijos. De esta manera, si era hijo, se sabía a qué lugar correspondía la restricción. De esta manera se fueron guardando los resultados en otro Array y después el resultado se volcaba en el original.

El método propuesto para solventar el problema de la asincronía no era perfecto, porque existía un caso en el que no funcionaba. Cuando el lugar no tenía restricciones de antemano. Si existían dos lugares así, no había manera de saber de cual era cual, dado que no existía ningún *id* para comprobarlo. Así que, la función debía ser llamada siempre que surgiera un cambio en el ArrayList grande de *ids,* de esta manera, si aparecían nuevos elementos en algún vacío, lo colocaría perfectamente. En caso de que se guarden 2 zonas sin restricciones, tampoco es un problema, porque ambos dejarían vacías sus respectivas entradas a la base de datos.

Acabando con la solución del problema, como se comentó, Se les crea las funcionalidades a los botones de borrado. Eliminan todos los elementos hijos del ConstraintLayout que se le pase. Esto siempre después de llamar al método de reestructuración. Es necesario que después del borrado, se deba insertar en la fila correspondiente del *ArrayList* de *ids*, uno vacío en vez de eliminarlo.

En cuanto a los botones para añadir restricciones, reestructurará el *ArrayList* y realizarán una petición a la base de datos para traer la lista con todas las restricciones. Se añadirá una opción adicional, la cual es “Insertar nueva restricción” y se mostrará a través de un *popup*.

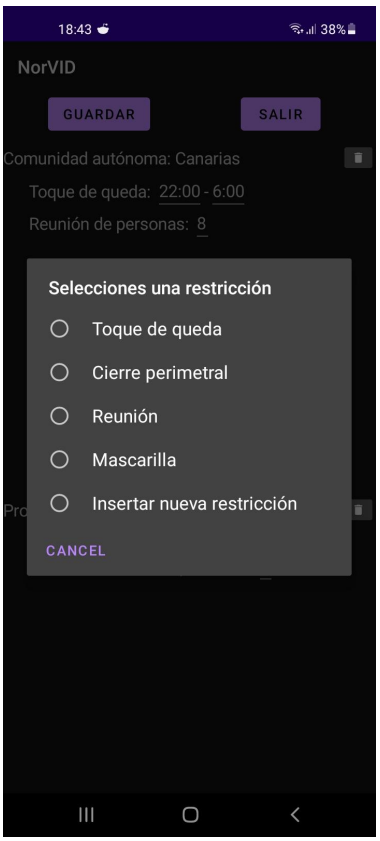


Figura 26: Menú para añadir otra restricción

Si se selecciona “Insertar nueva restricción”, el usuario será redirigido a la “pantalla de nueva restricción”. En otro caso, seguirá una misma mecánica con respecto a las insertadas anteriormente. Se trae de la base de datos la restricción correspondiente y va colocándola de manera ordenada. Los datos con clave iterador se colocarán como texto plano y las que tengan “dato” delante como texto editable.

Figura 27: Restricción añadida

En cuanto a los botones estáticos:

- “Salir” cierra la sesión y redirige a la “Pantalla principal”.
- “Guardar”, el código se puede ver aquí. Inutiliza los botones por si acaso la operación tarda más de lo previsto, reestructura el *ArrayList* e inicializa un bucle. Recorre todos los lugares y por cada uno de ellos realiza un borrado de sus restricciones y después la inserción de los nuevos datos. Empieza un bucle para cada restricción y otro para cada elemento. En la inserción tiene en cuenta de que tipo es el contenedor de cada dato, si es un *TextView* (texto plano), lo guarda con clave iterador, y si es un *EditText* (texto insertado), lo guardará con “data” delante del resto de la clave. Al acabar todo el bucle, muestra el siguiente mensaje:

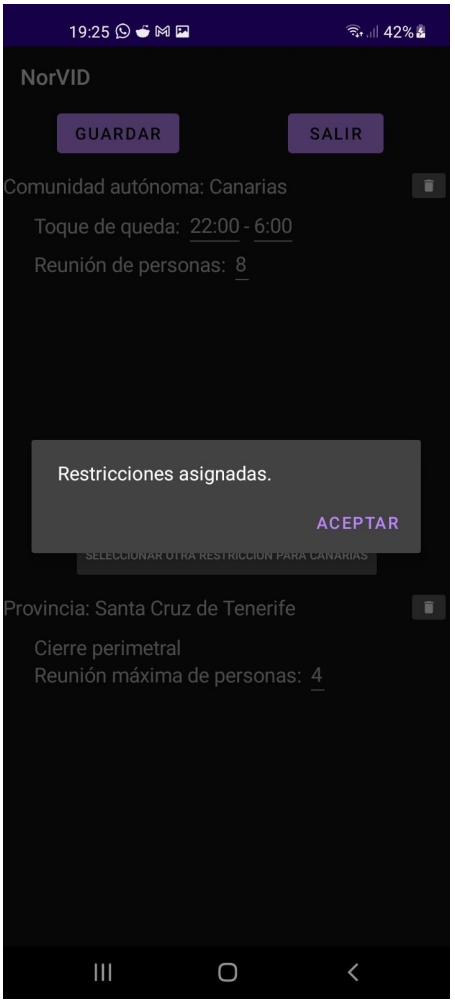


Figura 28: Restricciones guardadas

3.6. Pantalla de nueva restricción

La funcionalidad principal de esta pantalla es la creación y eliminación de las restricciones por parte del usuario.

La vista es mayoritariamente estática.

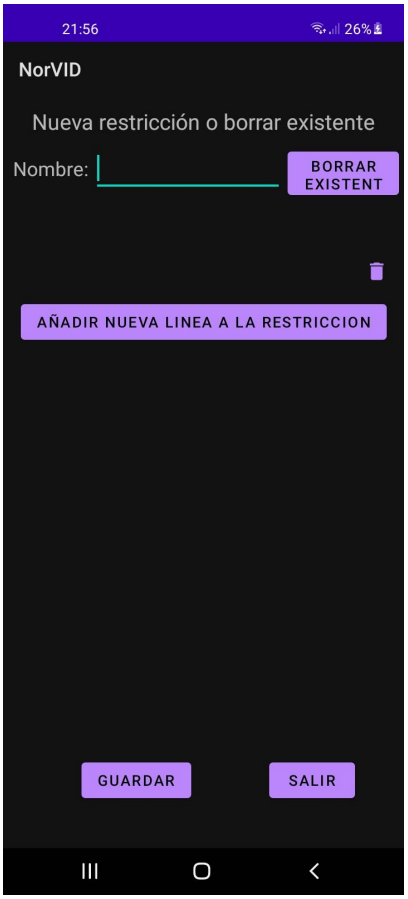


Figura 29: Pantalla de nueva restricción

Al iniciar la pantalla, se le asignan funcionalidades a los 5 botones.

- El botón de “Borrar existente” comprobará si el campo “Nombre” está vacío, si es así, mandará un error.

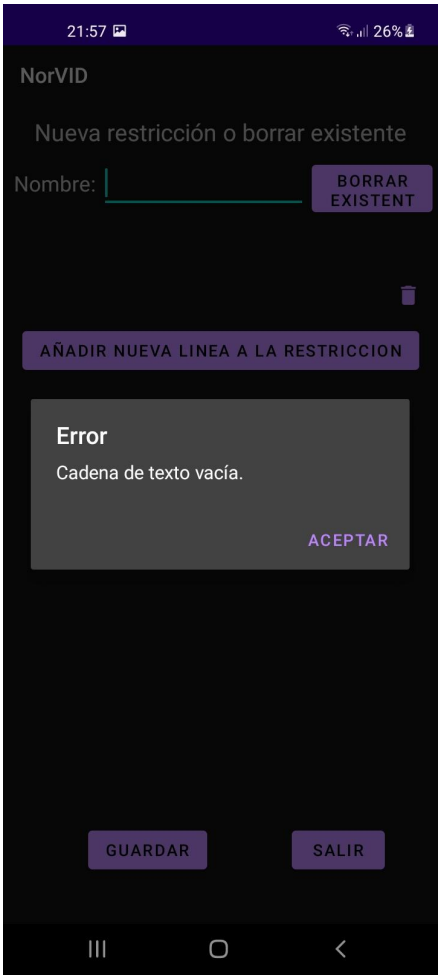


Figura 30: error: Cadena de texto vacía

En caso de estar relleno, realizará una petición a la base de datos y traerá la lista de restricciones. De allí buscará en el *Array* la normativa mencionada, y si no existe ningún *String* que coincida significa que no existe. Si se da ese caso se mostrará un error.

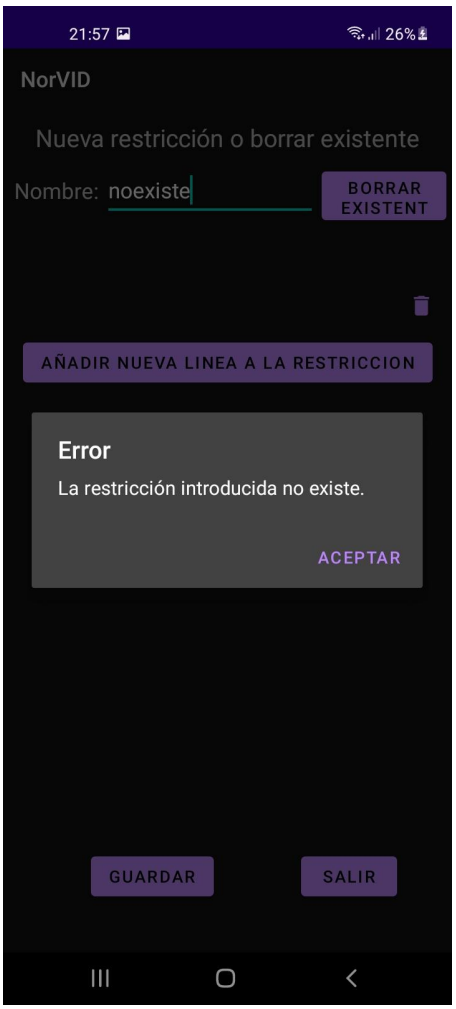


Figura 31: error: Restricción no existente

Si al final, la restricción está presente en la lista, se eliminará y se realizará otra petición. Esta vez para borrar la restricción en la colección de "Restricciones". Al eliminarla satisfactoriamente mostrará un mensaje de éxito. Código visible aquí.

Figura 32: Restricción borrada

- El botón para añadir nueva línea es un selector entre “Mostrar texto” e “Insertar texto”. Dentro de una restricción hay 2 categorías en cuanto a los textos. La primera es mostrar un texto, y que nunca cambiará, como podría ser “Toque de queda: “. La segunda será el campo para que los usuarios viertan la información, como por ejemplo las horas, números de personas, etc. Por ello, la primera opción, colocará texto editable y si se selecciona la otra, un texto plano. Un ejemplo de la creación de la restricción toque de queda es el siguiente:

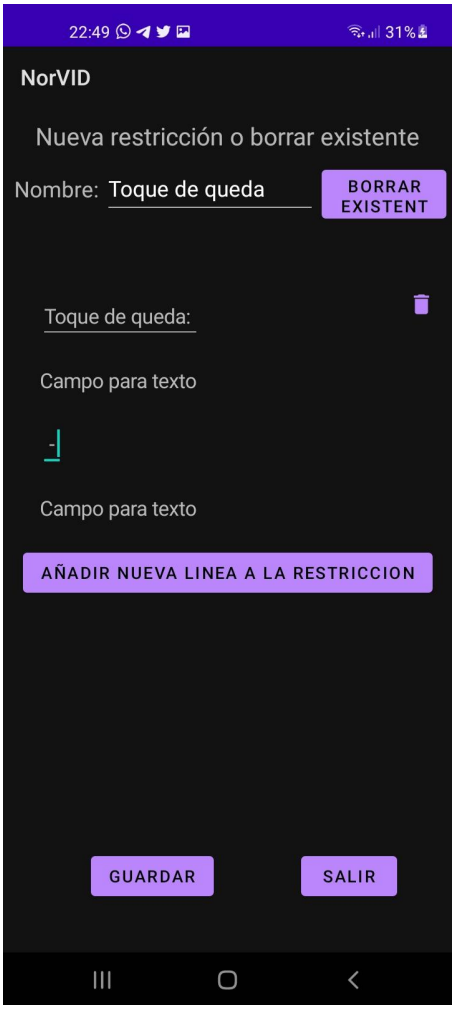


Figura 33: Ejemplo de creación de toque de queda

Como se puede observar, una vez se añade una línea, el botón se pondrá debajo de esta.

- El botón de borrado con el icono de una papelera, eliminará todas las líneas añadidas de la misma manera que los demás botones iguales que se han visto a lo largo de la aplicación.
- El botón de “Salir” simplemente redirigirá al usuario a la pantalla de inserción de restricciones.
- El botón de “Guardar” comprobará si se le ha asignado un nombre a la restricción y si no es así, mandará un error similar al de la figura[30]. Después comprobará si se le ha insertado algún campo a la restricción y no está vacía. Si no se cumple mostrará:

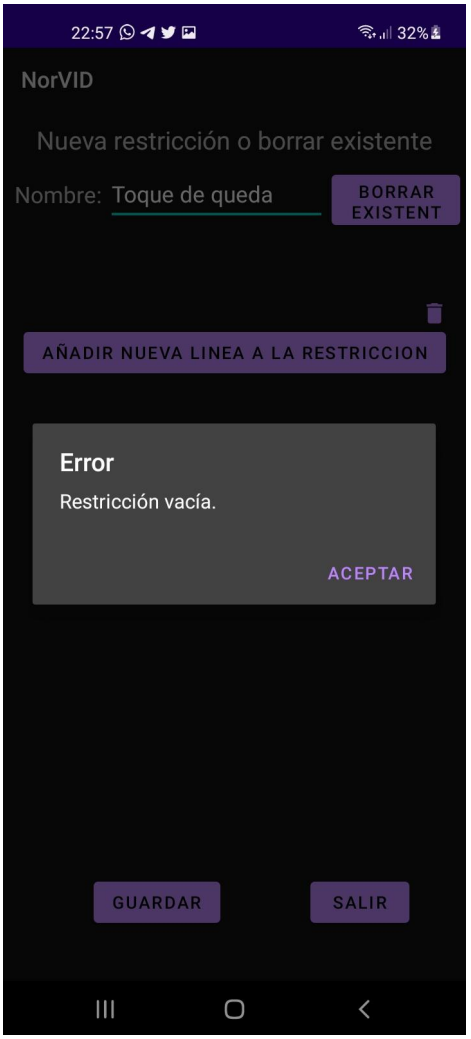


Figura 34: error: Restricción vacía

Los siguientes pasos son similares a los de eliminación de restricción. Primero se traerá de la base de datos la lista de restricciones y se comprobará si contiene el nombre. Si el nombre ya existe, quiere decir que la restricción ya está creada, o hay otra nombrada igual.

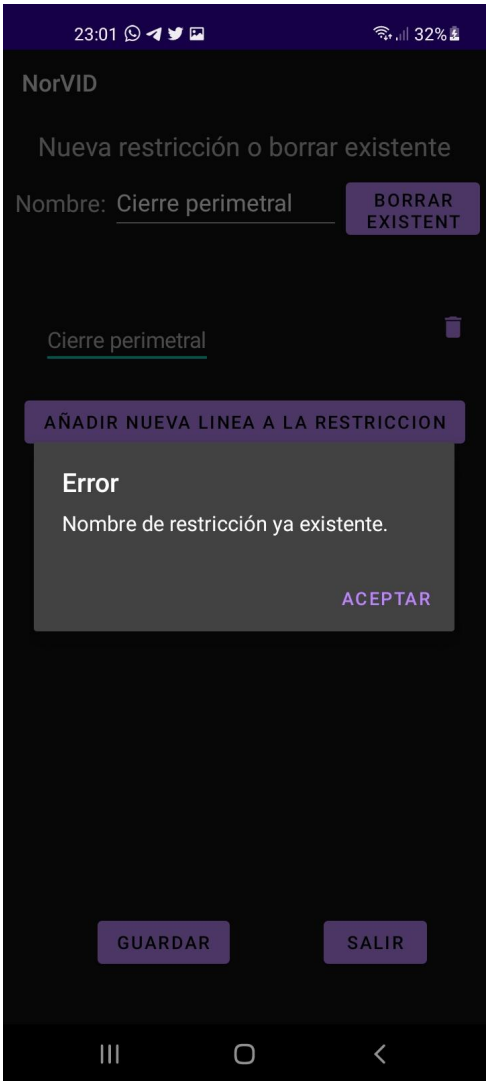


Figura 35: error: Restricción ya existente

En caso de que no esté, se colocará dentro del *Array* y se realizará una petición de escritura a la base de datos para crearla una entrada en la colección “Restricción”, una vez se complete mostrará lo siguiente:

Figura 36: Restricción creada

Una vez guardada se borrarán todos los campos de la vista de la restricción.

Una cosa adicional, es que cada vez que se pulsa un botón que requiera de llamadas a la base de datos, llamará a una función que se encargará de inutilizar todos los botones, por si la operación tarda más de lo esperado. En cuanto todas las llamadas acaban, se vuelven a habilitar. Código visible aquí.

4. Presupuesto

En la tabla de a continuación se especifican las horas invertidas aproximadamente en la realización de cada tarea, tomando como referencia que el coste por hora se calcula en base a 20€.

Tareas	Horas empleadas	Precio
Investigación	20 horas	400€
Conexión Frontend/Firebase	5 horas	100€
Creación de vistas	17 horas	340€
Implementación de la geolocalización	6 horas	120€
Añadir las búsquedas	10 horas	200€
Implementación de usuarios	8 horas	160€
Modificación de restricciones	30 horas	600€
Muestra de restricciones	18 horas	360€
Creación de restricciones	14 horas	280€
Corrección de errores	40 horas	800€
Total	**168 horas**	**3360€**

Tabla 1: Coste del proyecto

5. Conclusiones y líneas futuras

En cuanto a las conclusiones sacadas del producto creado, se obtiene que:

- Este proyecto implica una aplicación móvil, la cual puede traer beneficios a la sociedad y a su posible clientela. Se considera que lleva un retraso considerable, puesto que en el mercado ya existen 2 apps similares y el tiempo de vida del producto se ve limitado a la situación actual de pandemia. Por lo que se incita a que salga con la menor brevedad posible al mercado
- Una característica positiva que contiene la aplicación, es que está realizada de forma versátil. Esto se debe a que las normativas son modificables, es decir, se pueden borrar y crear por parte de los usuarios. En adición, la aplicación tiene la propiedad de ser reutilizable para enfermedades que puedan surgir a posteriori, ya que los campos de información no incluyen el nombre (solo en el apartado del título).

En cuanto a las conclusiones propias:

- Se ha ganado experiencia y soltura en el uso del lenguaje de programación *java* y el software *Android Studio*.
- A su vez, fue emocionante trabajar por primera vez con la geolocalización y observar su funcionamiento.
- Se comprende mejor el uso de las herramientas de *Firebase*. Se ve la facilidad y variedad para realizar peticiones a la base de datos y para autenticar usuarios.
- Por último, fue conmovedor, después de tanto altibajo y problemas que ha conllevado, ver cómo evolucionó y se convirtió en algo funcional.

Existen varias mejoras que se podrían aplicar al programa y no se han llevado a cabo por falta de tiempo o complejidad.

- Permitir el borrado de restricción en restricción en la pantalla de inserción de normativas. El mayor problema es la colocación, recolocar

los inferiores y ponerlos debajo del superior es complicado sobre todo si los *ids* de los elementos son generados dinámicamente. La técnica de irlos guardando sobre la marcha no funciona del todo y crear un botón que borre los elementos pasados como parámetros no sirve, porque es necesario al menos un *id* de la restricción inferior, y para la asignación del método al botón todavía no existe.

- En la creación de una nueva normativa, poder seleccionar el tipo de texto que se inserte, y no solo sea texto plano. Que sea posible que en normativas horarias, los campos usados fueran *date*, y que las numéricas, *number*. De esta manera los usuarios sabrían con más exactitud que escribir en las normativas creadas por otros. Los problemas son cómo tratarlos en la base de datos y tratarlos a lo largo de todas las peticiones. En vez de *data* delante, se podría poner el tipo que sea y realizar donde reciban los datos más *if/else statements* para tratarlos.
- Diseñar de alguna manera más intuitiva la pantalla de creación de nuevas normativas.
- Dejar al usuario cambiar la contraseña, y no se la den creada. En la creación del usuario, añadir un campo más a la base de datos del usuario a *true*. Esto haría que la primera vez que inicie sesión, le envíe a una pantalla de cambio de contraseña, y al modificarla, el campo se pone a *false*.
- Mejorar estéticamente las vistas.

6. Summary and conclusions

Regarding the conclusions drawn from the created product, it is obtained that:

- This project involves a mobile application, which can bring benefits to society and its potential clientele. It is considered to be a considerable delay, since there are already 2 similar apps on the market and the life of the product is limited to the current pandemic situation. Therefore, it is encouraged to go out on the market as soon as possible

- A positive feature that the application contains is that it is made in a versatile way. This is because the regulations are modifiable, that is, they can be deleted and created by users. In addition, the application has the property of being reusable for diseases that may arise later, since the information fields do not include the name (only in the title section).

Regarding the own conclusions:

- Experience and fluency in the use of the Java programming language and Android Studio software has been gained.
- At the same time, it was exciting to work with geolocation for the first time and to observe how it works.
- The use of Firebase tools is better understood. You see the ease and variety to make requests to the database and to authenticate users.
- Lastly, it was moving, after all the ups and downs and problems that it has entailed, to see how it evolved and became something functional.

There are several improvements that could be applied to the program and they have not been carried out due to lack of time or complexity.

- Allow the deletion of restriction in restriction in the policy insertion screen. The biggest problem is the placement, repositioning the lower ones and putting them under the upper one is complicated especially if the ids of the elements are dynamically generated. The technique of saving them on the fly does not work at all and creating a button that deletes the elements passed as parameters does not work, because at least one id of the lower restriction is necessary, and for the assignment of the method to the button it does not yet exist.
- In the creation of a new regulation, to be able to select the type of text to be inserted, and not just plain text. That it is possible that in time regulations, the fields used were date, and that the numerical ones, number. In this way users would know more exactly what to write in the regulations created by others. The problems are how to deal with them in the database and deal with them throughout all requests. Instead of data in front, you

could put whatever type it is and make where the data is received more if/else statements to deal with it.

- Design the screen for creating new regulations in a more intuitive way.
- Let the user change the password, and do not give it created. On user creation, add one more field to the user's database to true. This would cause the first time you log in to send you to a password change screen, and when you change it, the field is set to false.
- Aesthetically improve the views.

7. Bibliografía

[1] Universidad de La Laguna, Grado de Ingeniería Informática
https://www.ull.es/grados/ingenieria-informatica/

[2] Android Studio, Guía para desarrolladores
https://developer.android.com/guide

[3] Documentación de Firebase
https://firebase.google.com/docs

[4] Stack Overflow
https://stackoverflow.com/

8. Métodos y código fuente referenciado

Función “showData()”:

```
private void showData() {
    if(!infos.isEmpty()){
        ConstraintLayout layout = (ConstraintLayout) CargandoMun.getParent();
        for(TextView r : infos){
            layout.removeView(r);
        }
    }
    if(!infosCCAA.isEmpty()){
        ConstraintLayout layout = (ConstraintLayout) CargandoCCAA.getParent();
        for(TextView r : infosCCAA){
            layout.removeView(r);
        }
    }
    if(!infosProv.isEmpty()){
        ConstraintLayout layout = (ConstraintLayout) CargandoProv.getParent();
        for(TextView r : infosProv){
            layout.removeView(r);
        }
    }
    infos.clear();
    infosCCAA.clear();
    infosProv.clear();
    lid = 1000;

    CollectionReference docRef =
db.collection("Municipios").document(municipio).collection("Restricciones");
    CollectionReference docRefProv =
db.collection("Provincias").document(provincia).collection("Restricciones");
    CollectionReference docRefCCAA =
db.collection("CCAA").document(comunidadAutonoma).collection("Restricciones");
    docRef.get().addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            List<DocumentSnapshot> documents = task.getResult().getDocuments();
            try {
                DocumentSnapshot totry = documents.get(0);
                ConstraintLayout layout = (ConstraintLayout) CargandoMun.getParent();
                int idlast = -1;
                for (int i = 0; i < documents.size(); i++) {
                    boolean first = true;
                    long datalen = (long) documents.get(i).getData().size();
                    for (int j = 0; j < datalen; j++) {
                        ConstraintSet set = new ConstraintSet();
                        TextView restriction = new TextView(MainActivity.this);
```

```
                        restriction.setId(lid);
                        ConstraintLayout.LayoutParams lp = new
ConstraintLayout.LayoutParams(ConstraintLayout.LayoutParams.WRAP_CONTENT,
ConstraintLayout.LayoutParams.WRAP_CONTENT);
                        layout.addView(restriction, lp);
                        if ((String)
documents.get(i).getData().get(Integer.toString(j)) != null) {
                            restriction.setText((String)
documents.get(i).getData().get(Integer.toString(j)));
                        } else {
                            restriction.setText((String)
documents.get(i).getData().get("data" + j));
                        }
                        restriction.setTextSize(14);
                        restriction.setTextColor(Color.parseColor("#BBBBBB"));
                        set.clone(layout);
                        if (idlast == -1) {
                            set.connect(restriction.getId(), ConstraintSet.TOP,
ConstraintSet.PARENT_ID, ConstraintSet.TOP);
                            set.connect(restriction.getId(), ConstraintSet.START,
ConstraintSet.PARENT_ID, ConstraintSet.START, 50);
                            first = false;
                        } else {
                            if (first) {
                                set.connect(restriction.getId(), ConstraintSet.TOP,
idlast, ConstraintSet.BOTTOM);
                                set.connect(restriction.getId(), ConstraintSet.START,
ConstraintSet.PARENT_ID, ConstraintSet.START, 50);
                                first = false;
                            } else {
                                set.connect(restriction.getId(), ConstraintSet.START,
idlast, ConstraintSet.END);
                                set.connect(restriction.getId(), ConstraintSet.TOP,
idlast, ConstraintSet.TOP);
                                set.connect(restriction.getId(), ConstraintSet.BOTTOM,
idlast, ConstraintSet.BOTTOM);
                            }
                        }
                        set.applyTo(layout);
                        idlast = lid;
                        lid++;
                        infos.add(restriction);
                        CargandoMun.setText("");
                    }
                }
            } catch (Exception err) {
                CargandoMun.setText("Ninguna restricción activa");
            }
        }
    });
    docRefCCAA.get().addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            List<DocumentSnapshot> documents = task.getResult().getDocuments();
```

```
            try {
                DocumentSnapshot totry = documents.get(0);
                ConstraintLayout layout = (ConstraintLayout) CargandoCCAA.getParent();
                int idlast = -1;
                for (int i = 0; i < documents.size(); i++) {
                    boolean first = true;
                    long datalen = (long) documents.get(i).getData().size();
                    for (int j = 0; j < datalen; j++) {
                        ConstraintSet set = new ConstraintSet();
                        TextView restriction = new TextView(MainActivity.this);
                        restriction.setId(lid);
                        ConstraintLayout.LayoutParams lp = new
ConstraintLayout.LayoutParams(ConstraintLayout.LayoutParams.WRAP_CONTENT,
ConstraintLayout.LayoutParams.WRAP_CONTENT);
                        layout.addView(restriction, lp);
                        if ((String)
documents.get(i).getData().get(Integer.toString(j)) != null) {
                            restriction.setText((String)
documents.get(i).getData().get(Integer.toString(j)));
                        } else {
                            restriction.setText((String)
documents.get(i).getData().get("data" + j));
                        }
                        restriction.setTextSize(14);
                        restriction.setTextColor(Color.parseColor("#BBBBBB"));
                        set.clone(layout);
                        if (idlast == -1) {
                            set.connect(restriction.getId(), ConstraintSet.TOP,
ConstraintSet.PARENT_ID, ConstraintSet.TOP);
                            set.connect(restriction.getId(), ConstraintSet.START,
ConstraintSet.PARENT_ID, ConstraintSet.START, 50);
                            first = false;
                        } else {
                            if (first) {
                                set.connect(restriction.getId(), ConstraintSet.TOP,
idlast, ConstraintSet.BOTTOM);
                                set.connect(restriction.getId(), ConstraintSet.START,
ConstraintSet.PARENT_ID, ConstraintSet.START, 50);
                                first = false;
                            } else {
                                set.connect(restriction.getId(), ConstraintSet.START,
idlast, ConstraintSet.END);
                                set.connect(restriction.getId(), ConstraintSet.TOP,
idlast, ConstraintSet.TOP);
                                set.connect(restriction.getId(), ConstraintSet.BOTTOM,
idlast, ConstraintSet.BOTTOM);
                            }
                        }
                        set.applyTo(layout);
                        idlast = lid;
                        lid++;
                        infos.add(restriction);
                        CargandoCCAA.setText("");
```

```
                }
            }
        } catch (Exception err) {
            CargandoCCAA.setText("Ninguna restricción activa");
        }
    }
});
docRefProv.get().addOnCompleteListener(task -> {
    if (task.isSuccessful()) {
        List<DocumentSnapshot> documents = task.getResult().getDocuments();
        try {
            DocumentSnapshot totry = documents.get(0);
            ConstraintLayout layout = (ConstraintLayout) CargandoProv.getParent();
            int idlast = -1;
            for (int i = 0; i < documents.size(); i++) {
                boolean first = true;
                long datalen = (long) documents.get(i).getData().size();
                for (int j = 0; j < datalen; j++) {
                    ConstraintSet set = new ConstraintSet();
                    TextView restriction = new TextView(MainActivity.this);
                    restriction.setId(lid);
                    ConstraintLayout.LayoutParams lp = new
ConstraintLayout.LayoutParams(ConstraintLayout.LayoutParams.WRAP_CONTENT,
ConstraintLayout.LayoutParams.WRAP_CONTENT);
                    layout.addView(restriction, lp);
                    if ((String)
documents.get(i).getData().get(Integer.toString(j)) != null) {
                        restriction.setText((String)
documents.get(i).getData().get(Integer.toString(j)));
                    } else {
                        restriction.setText((String)
documents.get(i).getData().get("data" + j));
                    }
                    restriction.setTextSize(14);
                    restriction.setTextColor(Color.parseColor("#BBBBBB"));
                    set.clone(layout);
                    if (idlast == -1) {
                        set.connect(restriction.getId(), ConstraintSet.TOP,
ConstraintSet.PARENT_ID, ConstraintSet.TOP);
                        set.connect(restriction.getId(), ConstraintSet.START,
ConstraintSet.PARENT_ID, ConstraintSet.START, 50);
                        first = false;
                    } else {
                        if (first) {
                            set.connect(restriction.getId(), ConstraintSet.TOP,
idlast, ConstraintSet.BOTTOM);
                            set.connect(restriction.getId(), ConstraintSet.START,
ConstraintSet.PARENT_ID, ConstraintSet.START, 50);
                            first = false;
                        } else {
                            set.connect(restriction.getId(), ConstraintSet.START,
idlast, ConstraintSet.END);
```

```
                                set.connect(restriction.getId(), ConstraintSet.TOP,
idlast, ConstraintSet.TOP);
                                set.connect(restriction.getId(), ConstraintSet.BOTTOM,
idlast, ConstraintSet.BOTTOM);
                            }
                        }
                        set.applyTo(layout);
                        idlast = lid;
                        lid++;
                        infos.add(restriction);
                        CargandoProv.setText("");
                    }
                }
            } catch (Exception err) {
                CargandoProv.setText("Ninguna restricción activa");
            }
        }
    });
}
```

Inicializar barra de búsqueda de CCAA en la pantalla principal:

```
db.collection("Listas").document("CCAA").get()
            .addOnSuccessListener(documentSnapshot -> {
                ArrayList<String> ccaaList = (ArrayList<String>)
documentSnapshot.getData().get("lista");
                String[] arrayAdapter = new String[ccaaList.size()];
                for (int i = 0; i < ccaaList.size(); i++)
                    arrayAdapter[i] = ccaaList.get(i);
                ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, arrayAdapter);
                ccaaView.setAdapter(adapter);

                ccaaView.setOnItemClickListener((parent, view, position, id) -> {
                    CCAAViewText = parent.getItemAtPosition(position).toString();
                    comunidadAutonoma = CCAAViewText;
                });
            });
```

Botón de autenticado:

```
loginButton.setOnClickListener(v -> {
   loginButton.setEnabled(false);
   if(email.getText()!=null&&!email.getText().toString().equals("")&&
pass.getText()!=null&&!pass.getText().toString().equals("")){
       mAuth.getInstance().signInWithEmailAndPassword(email.getText().toString(),
pass.getText().toString())
           .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
               @Override
               public void onComplete(@NonNull Task<AuthResult> task) {
                   if (task.isSuccessful()) {
                       FirebaseUser user = mAuth.getCurrentUser();
                       if(!user.getEmail().equals(administrator)) {
                           Intent insertIntent = new Intent(loginActivity.this,
InsertRestrictions.class).putExtra("email", user.getEmail());
                           loginActivity.this.startActivity(insertIntent);
                       }
                       else{
                           Intent adminIntent = new Intent(loginActivity.this,
adminActivity.class);
                           loginActivity.this.startActivity(adminIntent);
                       }
                   } else {
                       AlertDialog.Builder error = new
AlertDialog.Builder(loginActivity.this);
                       error.setTitle("Error");
                       error.setMessage("Correo electrónico o contraseña inválido.");
                       error.setPositiveButton("Aceptar", null);

                       AlertDialog dialog = error.create();
                       dialog.show();
                       loginButton.setEnabled(true);
                   }
               }
           });
   }
   else{
       AlertDialog.Builder error = new AlertDialog.Builder(loginActivity.this);
       error.setTitle("Error");
       error.setMessage("Rellene los campos de correo electrónico y contraseña.");
       error.setPositiveButton("Aceptar", null);

       AlertDialog dialog = error.create();
       dialog.show();
       loginButton.setEnabled(true);
   }

});
```

Inicializar barra de búsqueda de CCAA de la pantalla de administrador:

```
db.collection("Listas").document("CCAA").get()
```

```
            .addOnSuccessListener(documentSnapshot -> {
                ArrayList<String> ccaaList = (ArrayList<String>)
documentSnapshot.getData().get("lista");
                String[] arrayAdapter = new String[ccaaList.size()];
                for(int i = 0; i < ccaaList.size(); i++)
                    arrayAdapter[i] = ccaaList.get(i);
                ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, arrayAdapter);
                autoccaa.setAdapter(adapter);

                autoccaa.setOnItemClickListener((parent, view, position, id) -> {
                    ConstraintSet set = new ConstraintSet();
                    TextView v = new TextView(adminActivity.this);
                    v.setId(lid);
                    ConstraintLayout.LayoutParams lp = new
ConstraintLayout.LayoutParams(ConstraintLayout.LayoutParams.WRAP_CONTENT,
ConstraintLayout.LayoutParams.WRAP_CONTENT);
                    v.setText(parent.getItemAtPosition(position).toString());
                    v.setTextSize(17);
                    v.setTextColor(Color.parseColor("#BBBBBB"));
                    clccaa.addView(v, lp);
                    set.clone(clccaa);
                    ConstraintLayout.LayoutParams params =
(ConstraintLayout.LayoutParams) autoccaa.getLayoutParams();
                    set.connect(v.getId(), ConstraintSet.TOP, params.topToBottom,
ConstraintSet.BOTTOM,10);
                    set.connect(v.getId(), ConstraintSet.START,
ConstraintSet.PARENT_ID, ConstraintSet.START, 30);
                    set.connect(autoccaa.getId(), ConstraintSet.TOP, v.getId(),
ConstraintSet.BOTTOM, 10);
                    set.applyTo(clccaa);
                    ccaas.add(v.getId());
                    lid++;
                    autoccaa.setText("");
                });
            });
```

Botón de borrado de CCAA de la pantalla de administrador:

```
buttonDCCAA.setOnClickListener(v -> {
   for(int i = 0; i < ccaas.size(); i++)
       clccaa.removeView(clccaa.findViewById(ccaas.get(i)));
   ccaas.clear();
   ConstraintSet set = new ConstraintSet();
   set.clone(clccaa);
   set.connect(autoccaa.getId(),ConstraintSet.TOP, ConstraintSet.PARENT_ID,
ConstraintSet.TOP, 10);
   set.applyTo(clccaa);
});
```

Inicializar botón “Crear” de la pantalla de administrador:

```
crear.setOnClickListener(v -> {
   if(email.getText()!=null && !email.getText().toString().equals("") &&
pass.getText()!=null && !pass.getText().toString().equals("")){

FirebaseAuth.getInstance().createUserWithEmailAndPassword(email.getText().toString(),p
ass.getText().toString())
            .addOnCompleteListener(adminActivity.this, task -> {
                if (task.isSuccessful()) {
                    Map<String, Object> data = new HashMap<>();
                    ArrayList<String> nccaa = new ArrayList<>();
                    ArrayList<String> nprov= new ArrayList<>();
                    ArrayList<String> nmun = new ArrayList<>();
                    for(int i = 0; i < ccaas.size(); i++) {
                        TextView v1 = clccaa.findViewById(ccaas.get(i));
                        nccaa.add(v1.getText().toString());
                    }
                    data.put("CCAA", nccaa);
                    for(int i = 0; i < provs.size(); i++) {
                        TextView v1 = clprov.findViewById(provs.get(i));
                        nprov.add(v1.getText().toString());
                    }
                    data.put("Prov", nprov);
                    for(int i = 0; i < muns.size(); i++) {
                        TextView v1 = clmun.findViewById(muns.get(i));
                        nmun.add(v1.getText().toString());
                    }
                    data.put("Mun", nmun);

db.collection("users").document(email.getText().toString()).set(data);
                    AlertDialog.Builder error = new
AlertDialog.Builder(adminActivity.this);
                    error.setTitle("Completo");
                    error.setMessage("Cuenta creada con éxito");
                    error.setPositiveButton("Aceptar", null);

                    AlertDialog dialog = error.create();
                    dialog.show();
                }
                else{
                    AlertDialog.Builder error = new
AlertDialog.Builder(adminActivity.this);
                    error.setTitle("Error");
                    error.setMessage("Usuario ya existente");
                    error.setPositiveButton("Aceptar", null);

                    AlertDialog dialog = error.create();
                    dialog.show();
                }
            });
   }
   else{
       AlertDialog.Builder error = new AlertDialog.Builder(adminActivity.this);
```

```
        error.setTitle("Error");
        error.setMessage("Rellene los campos de correo electrónico y contraseña");
        error.setPositiveButton("Aceptar", null);

        AlertDialog dialog = error.create();
        dialog.show();
    }
});
```

Inicializador de la vista de la pantalla de inserción de restricciones solo para las comunidades autónomas:

```
if (document.getData().get("CCAA") != null) {
    List<String> CCAAs = (List<String>) document.getData().get("CCAA");
    ArrayList<String> Lugar = new ArrayList<>();
    for (int i = 0; i < CCAAs.size(); i++) {
        Lugar.add(CCAAs.get(i));
        ConstraintSet set = new ConstraintSet();
        TextView restriction = new TextView(InsertRestrictions.this);
        restriction.setId(lid);
        lid++;
        ConstraintLayout.LayoutParams lp = new
ConstraintLayout.LayoutParams(ConstraintLayout.LayoutParams.WRAP_CONTENT,
ConstraintLayout.LayoutParams.WRAP_CONTENT);

        restriction.setText("Comunidad autónoma: " + CCAAs.get(i));
        restriction.setTextSize(17);
        restriction.setTextColor(Color.parseColor("#BBBBBB"));
        layoutGeneral.addView(restriction, lp);
        set.clone(layoutGeneral);
        if (lid != 2001)
            set.connect(restriction.getId(), ConstraintSet.TOP, lid - 2,
ConstraintSet.BOTTOM, 20);
        else
            set.connect(restriction.getId(), ConstraintSet.TOP, logout.getId(),
ConstraintSet.BOTTOM, 20);

        set.applyTo(layoutGeneral);

        ScrollView scrollView = new ScrollView(InsertRestrictions.this);
        scrollView.setId(lid);
        lid++;
        ConstraintLayout.LayoutParams sclp = new
ConstraintLayout.LayoutParams(ConstraintLayout.LayoutParams.MATCH_PARENT, 500);
        layoutGeneral.addView(scrollView, sclp);
        set.clone(layoutGeneral);
        set.connect(scrollView.getId(), ConstraintSet.TOP, restriction.getId(),
ConstraintSet.BOTTOM, 20);
        set.applyTo(layoutGeneral);

        ConstraintLayout cl = new ConstraintLayout(InsertRestrictions.this);
```

```
        cl.setId(lid);
        lid++;
        ConstraintLayout.LayoutParams cllp = new
ConstraintLayout.LayoutParams(ConstraintLayout.LayoutParams.MATCH_PARENT, 500);
        layoutGeneral.addView(cl, cllp);
        set.clone(layoutGeneral);
        set.connect(cl.getId(), ConstraintSet.TOP, scrollView.getId(),
ConstraintSet.TOP);
        set.connect(cl.getId(), ConstraintSet.BOTTOM, scrollView.getId(),
ConstraintSet.BOTTOM);
        set.connect(cl.getId(), ConstraintSet.END, scrollView.getId(),
ConstraintSet.END);
        set.connect(cl.getId(), ConstraintSet.START, scrollView.getId(),
ConstraintSet.START);
        set.applyTo(layoutGeneral);

        cls.add(cl);

        ImageButton deleteButton = new ImageButton(InsertRestrictions.this);
        ConstraintLayout.LayoutParams lpButton = new
ConstraintLayout.LayoutParams(ConstraintLayout.LayoutParams.WRAP_CONTENT,
ConstraintLayout.LayoutParams.WRAP_CONTENT);
        deleteButton.setId(lid);
        lid++;
        deleteButton.setImageResource(R.drawable.delete);
        layoutGeneral.addView(deleteButton, lpButton);
        set.clone(layoutGeneral);
        set.connect(deleteButton.getId(), ConstraintSet.END, ConstraintSet.PARENT_ID,
ConstraintSet.END);
        set.connect(deleteButton.getId(), ConstraintSet.TOP, restriction.getId(),
ConstraintSet.TOP);
        set.connect(deleteButton.getId(), ConstraintSet.BOTTOM, restriction.getId(),
ConstraintSet.BOTTOM);
        set.applyTo(layoutGeneral);
        int fila = contador;
        deleteClick(deleteButton, cl, fila);

        Button newRes = new Button(InsertRestrictions.this);
        ConstraintLayout.LayoutParams lpNew = new
ConstraintLayout.LayoutParams(ConstraintLayout.LayoutParams.WRAP_CONTENT,
ConstraintLayout.LayoutParams.WRAP_CONTENT);
        newRes.setId(lid);
        lid++;
        newRes.setText("Seleccionar otra restricción para " + CCAAs.get(i));
        newRes.setTextSize(10);
        layoutGeneral.addView(newRes, lpNew);
        set.clone(layoutGeneral);
        set.connect(newRes.getId(), ConstraintSet.TOP, cl.getId(),
ConstraintSet.BOTTOM,10);
        set.connect(newRes.getId(), ConstraintSet.START, ConstraintSet.PARENT_ID,
ConstraintSet.START, 20);
        set.connect(newRes.getId(), ConstraintSet.END, ConstraintSet.PARENT_ID,
ConstraintSet.END, 20);
```

```
        set.applyTo(layoutGeneral);
        int copiaContador = contador;
        restriccionesClick(newRes, cl, copiaContador, emailUse);

        String comunidad = CCAAs.get(i);

        ArrayList<ArrayList<Integer>> idCadaRestriccion = new
ArrayList<ArrayList<Integer>>();

        CollectionReference dbCCAA =
db.collection("CCAA").document(comunidad).collection("Restricciones");
        dbCCAA.get().addOnCompleteListener(task1 -> {
            if (task1.isSuccessful()) {
                List<DocumentSnapshot> documents = task1.getResult().getDocuments();
                int idlast = -1;
                ConstraintSet setIn = new ConstraintSet();
                try {
                    DocumentSnapshot totry = documents.get(0);
                    for (int finalI = 0; finalI < documents.size(); finalI++) {
                        boolean firstIn = true;

                        ArrayList<Integer> idCadaPalabra = new ArrayList<Integer>();
                        for (int j = 0; j < documents.get(finalI).getData().size();
j++) {

                            if ((String)
documents.get(finalI).getData().get(Integer.toString(j)) != null) {
                                TextView restrictions = new
TextView(InsertRestrictions.this);
                                restrictions.setId(lid);

                                ConstraintLayout.LayoutParams lpRes = new
ConstraintLayout.LayoutParams(ConstraintLayout.LayoutParams.WRAP_CONTENT,
ConstraintLayout.LayoutParams.WRAP_CONTENT);

                                restrictions.setText((String)
documents.get(finalI).getData().get(Integer.toString(j)));
                                restrictions.setTextSize(17);
                                restrictions.setTextColor(Color.parseColor("#BBBBBB"));
                                cl.addView(restrictions, lpRes);
                                setIn.clone(cl);

                                if (idlast == -1) {
                                    setIn.connect(restrictions.getId(),
ConstraintSet.TOP, ConstraintSet.PARENT_ID, ConstraintSet.TOP);
                                    setIn.connect(restrictions.getId(),
ConstraintSet.START, ConstraintSet.PARENT_ID, ConstraintSet.START, 50);
                                    firstIn = false;
                                } else {
                                    if (firstIn) {
                                        setIn.connect(restrictions.getId(),
ConstraintSet.TOP, idlast, ConstraintSet.BOTTOM);
```

```
                                    setIn.connect(restrictions.getId(),
ConstraintSet.START, ConstraintSet.PARENT_ID, ConstraintSet.START, 50);
                                    firstIn = false;
                                } else {
                                    setIn.connect(restrictions.getId(),
ConstraintSet.START, idlast, ConstraintSet.END);
                                    setIn.connect(restrictions.getId(),
ConstraintSet.TOP, idlast, ConstraintSet.TOP);
                                    setIn.connect(restrictions.getId(),
ConstraintSet.BOTTOM, idlast, ConstraintSet.BOTTOM);
                                }
                            }
                        } else {
                            EditText restrictions = new
EditText(InsertRestrictions.this);
                            restrictions.setId(lid);

                            ConstraintLayout.LayoutParams lpRes = new
ConstraintLayout.LayoutParams(ConstraintLayout.LayoutParams.WRAP_CONTENT,
ConstraintLayout.LayoutParams.WRAP_CONTENT);

                            restrictions.setText((String)
documents.get(finalI).getData().get("data" + j));
                            restrictions.setTextSize(17);
                            restrictions.setTextColor(Color.parseColor("#BBBBBB"));
                            cl.addView(restrictions, lpRes);
                            setIn.clone(cl);

                            if (idlast == -1) {
                                setIn.connect(restrictions.getId(),
ConstraintSet.TOP, ConstraintSet.PARENT_ID, ConstraintSet.TOP);
                                setIn.connect(restrictions.getId(),
ConstraintSet.START, ConstraintSet.PARENT_ID, ConstraintSet.START, 50);
                                firstIn = false;
                            } else {
                                if (firstIn) {
                                    setIn.connect(restrictions.getId(),
ConstraintSet.TOP, idlast, ConstraintSet.BOTTOM);
                                    setIn.connect(restrictions.getId(),
ConstraintSet.START, ConstraintSet.PARENT_ID, ConstraintSet.START, 50);
                                    firstIn = false;
                                } else {
                                    setIn.connect(restrictions.getId(),
ConstraintSet.START, idlast, ConstraintSet.END);
                                    setIn.connect(restrictions.getId(),
ConstraintSet.TOP, idlast, ConstraintSet.TOP);
                                    setIn.connect(restrictions.getId(),
ConstraintSet.BOTTOM, idlast, ConstraintSet.BOTTOM);
                                }
                            }
                        }
                        setIn.applyTo(cl);
```

```
                            idlast = lid;
                            lid++;
                            idCadaPalabra.add(idlast);
                        }
                        idCadaRestriccion.add(idCadaPalabra);
                    }

                } catch (Exception err) { }
            }
            Guardar.add(idCadaRestriccion);
            contadorAsincrono++;
        });
        contador++;
    }
    Lugares.add(Lugar);
}
```

Función “reestructurar()”:

```
private void reestructurar(){
    ArrayList<ArrayList<ArrayList<Integer>>> aux = new ArrayList<>(cls.size());
    for(int i=0; i<cls.size(); i++){
        aux.add(new ArrayList<>());
    }
    for(int i=0; i<cls.size(); i++){
        if(Guardar.get(i).size() > 0){
            for(int j=0; j<cls.size(); j++){
                if(cls.get(j).findViewById(Guardar.get(i).get(0).get(0))!=null){
                    Log.d(TAG, j + "");
                    aux.set(j,Guardar.get(i));
                }
            }
        }
    }
    Guardar = aux;
}
```

Función “insercionDatos()”:

```
private void insercionDatos(String collection, String document, int n){
    for(int i = 0; i < Guardar.get(n).size(); i++){
        DocumentReference docRef =
db.collection(collection).document(document).collection("Restricciones").document("r"+
i);
        Map<String, Object> restriccion = new HashMap<>();
        for(int j = 0; j < Guardar.get(n).get(i).size(); j++){

if(cls.get(n).findViewById(Guardar.get(n).get(i).get(j)).getClass().getName().equals("
android.widget.TextView")){
                TextView v = cls.get(n).findViewById(Guardar.get(n).get(i).get(j));
```

```
                restriccion.put(Integer.toString(j), v.getText().toString());
            }else
if(cls.get(n).findViewById(Guardar.get(n).get(i).get(j)).getClass().getName().equals("
android.widget.EditText")){
                EditText v = cls.get(n).findViewById(Guardar.get(n).get(i).get(j));
                String data = "data"+j;
                restriccion.put(data, v.getText().toString());
            }
        }
        docRef.set(restriccion).addOnCompleteListener(task -> {Log.d(TAG,
"Agregado");});
    }
}
```

Inicializar botón “Borrar existente” de la pantalla de nueva restricción:

```
deleteR.setOnClickListener(v -> {
    setButtons(false);
    if(!documentN.getText().toString().equals("")&&documentN.getText()!=null){
        db.collection("Listas").document("Restricciones").get()
            .addOnCompleteListener(task -> {
                if (task.isSuccessful()) {
                    DocumentSnapshot document = task.getResult();
                    if (document.exists()) {
                        ArrayList listaRestricciones = (ArrayList)
document.getData().get("r");
                        boolean match =
listaRestricciones.contains(documentN.getText().toString());
                        if(match){
                            listaRestricciones.remove(documentN.getText().toString());
                            Map<String, Object> aux = new HashMap<>();
                            aux.put("r", listaRestricciones);
                            db.collection("Listas").document("Restricciones").set(aux)
                                .addOnSuccessListener(aVoid -> {

db.collection("Restricciones").document(documentN.getText().toString()).delete()
                                        .addOnSuccessListener(aVoid1 -> {
                                            AlertDialog.Builder error = new
AlertDialog.Builder(NewRestriction.this);
                                            error.setMessage("Restricción " +
documentN.getText().toString() + " borrada" );
                                            error.setPositiveButton("Aceptar", null);

                                            AlertDialog dialog = error.create();
                                            dialog.show();
                                        });
                                });
                        }
                        else{
                            AlertDialog.Builder error = new
AlertDialog.Builder(NewRestriction.this);
                            error.setTitle("Error");
```

```
                        error.setMessage("La restricción introducida no existe.");
                        error.setPositiveButton("Aceptar", null);

                        AlertDialog dialog = error.create();
                        dialog.show();
                    }
                }
            }
        });
    }
    else{
        AlertDialog.Builder error = new AlertDialog.Builder(NewRestriction.this);
        error.setTitle("Error");
        error.setMessage("Cadena de texto vacía.");
        error.setPositiveButton("Aceptar", null);

        AlertDialog dialog = error.create();
        dialog.show();
    }
    setButtons(true);
});
```

Inicializar botón “Guardar” de la pantalla de nueva restricción:

```
save.setOnClickListener(v -> {
    setButtons(false);
    if(!documentN.getText().toString().equals("")&&documentN.getText()!=null){
        if(ids.size()>0){
            Map<String, Object> nuevaR = new HashMap<>();

            for(int i = 0; i < ids.size(); i++){

if(cl.findViewById(ids.get(i)).getClass().getName().equals("android.widget.TextView"))
{
                    String key = "dato" + i;
                    nuevaR.put(key, "");
                }
                else
if(cl.findViewById(ids.get(i)).getClass().getName().equals("android.widget.EditText"))
{
                    EditText et = findViewById(ids.get(i));
                    nuevaR.put(Integer.toString(i), et.getText().toString());
                }
            }
            db.collection("Listas").document("Restricciones").get()
                .addOnCompleteListener(task -> {
                    if (task.isSuccessful()) {
                        DocumentSnapshot document = task.getResult();
                        if (document.exists()) {
                            ArrayList listaRestricciones = (ArrayList)
document.getData().get("r");
```

```
                            boolean match =
listaRestricciones.contains(documentN.getText().toString());
                            if (match) {
                                AlertDialog.Builder error = new
AlertDialog.Builder(NewRestriction.this);
                                error.setTitle("Error");
                                error.setMessage("Nombre de restricción ya
existente.");
                                error.setPositiveButton("Aceptar", null);
                                AlertDialog dialog = error.create();
                                dialog.show();
                            }
                            else{
                                listaRestricciones.add(documentN.getText().toString());
                                Map<String, Object> aux = new HashMap<>();
                                aux.put("r", listaRestricciones);

db.collection("Listas").document("Restricciones").set(aux)
                                        .addOnSuccessListener(aVoid -> {

db.collection("Restricciones").document(documentN.getText().toString()).set(nuevaR)
                                                .addOnSuccessListener(aVoid2 -> {
                                                    AlertDialog.Builder error = new
AlertDialog.Builder(NewRestriction.this);
                                                    error.setTitle("Completo");
                                                    error.setMessage("Se creó la
restricción " + documentN.getText().toString() + " correctamente.");
                                                    error.setPositiveButton("Aceptar",
null);
                                                    AlertDialog dialog = error.create();
                                                    dialog.show();

                                                    deleteAll();
                                                });
                                        });
                            }
                        }
                    }
                });
        }
        else{
            AlertDialog.Builder error = new AlertDialog.Builder(NewRestriction.this);
            error.setTitle("Error");
            error.setMessage("Restricción vacía.");
            error.setPositiveButton("Aceptar", null);

            AlertDialog dialog = error.create();
            dialog.show();
        }
    }
    else{
        AlertDialog.Builder error = new AlertDialog.Builder(NewRestriction.this);
        error.setTitle("Error");
```

```
        error.setMessage("Ningún nombre añadido a la restricción.");
        error.setPositiveButton("Aceptar", null);

        AlertDialog dialog = error.create();
        dialog.show();
    }
    setButtons(true);
});
```