



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



TFG del Grado en Ingeniería Informática

Seguimiento de
requerimientos para gestionar,
medir, evaluar y mejorar
Documentación Técnica



Presentado por Pablo Ahíta del Barrio
en Universidad de Burgos — 8 de julio de 2024
Tutor: Pedro Luis Sánchez Ortega

Índice general

Índice general	i
Índice de figuras	iii
Índice de tablas	vii
Apéndice A Plan de Proyecto Software	1
A.1. Edición del informe	1
A.2. Formato del informe	1
A.3. Comprobación de la ortografía del informe	2
A.4. Precedente	2
A.5. Planificación temporal	23
A.6. Estudio de viabilidad	24
Apéndice B Especificación de Requisitos	29
B.1. Introducción	29
B.2. Objetivos generales	29
B.3. Catálogo de requisitos	31
B.4. Especificación de requisitos	32
Apéndice C Especificación de diseño	45
C.1. Introducción	45
C.2. Diseño de datos	45
C.3. Diseño procedimental	99
C.4. Diseño arquitectónico	108
Apéndice D Documentación técnica de programación	161

D.1. Introducción	161
D.2. Estructura de directorios	161
D.3. Manual del programador	162
D.4. Compilación, instalación y ejecución del proyecto	162
Apéndice E Documentación de usuario	217
E.1. Requisitos de usuarios	217
E.2. Instalación	217
E.3. Manual del usuario	217
Bibliografía	219

Índice de figuras

A.1. Menú principal de OTEA	3
A.2. Apartado de introducción de datos de la organización o del servicio	4
A.3. Apartado de realización del test de indicadores. Primer indicador	6
A.4. Ventana flotante de finalización del test de indicadores antes de guardar los datos	7
A.5. Ventana flotante de finalización del test de indicadores después de guardar los datos	7
A.6. Gráfico de muestra de resultados del servicio u organización . .	8
A.7. Informe con la puntuación del test de indicadores	9
A.8. Apartado de informe final con los datos del centro, las observaciones y las conclusiones	11
A.9. Página principal de <i>Print to PDF Pro</i>	12
A.10. Mensaje de bienvenida del programa de instalación de <i>Print to PDF Pro</i>	13
A.11. Contrato de licencia de <i>Print to PDF Pro</i> , de obligada aceptación para la instalación del programa	14
A.12. Selección del directorio de instalación de <i>Print to PDF Pro</i> . .	15
A.13. Registro de <i>Print to PDF Pro</i> , de cumplimiento opcional.	16
A.14. Finalización del asistente de instalación de <i>Print to PDF Pro</i> . .	17
A.15. Ventana emergente donde se pregunta si se desea visualizar el fichero recién generado	17
A.16. Fichero de prueba de <i>Print to PDF Pro</i>	18
A.17. Informe final con el botón de la impresora seleccionado	19
A.18. Informe en PDF generado por el programa en <i>Microsoft Access</i>	20
A.19. Información general de la suscripción en Azure	24
A.20. Velocidad y previsión de gastos junto con los costes por cada recurso	25

A.21. Ejemplo del registro de un usuario de la <i>Fundación Miradas</i> , con la casilla de la aceptación de los datos de acuerdo con la ley	27
B.1. Diagrama de casos de uso para el actor <i>Administrador</i>	41
B.2. Diagrama de casos de uso para el actor <i>Director de organización evaluada</i>	42
B.3. Diagrama de casos de uso para el actor <i>Usuario de organización</i>	43
C.1. Paquete <i>user</i>	46
C.2. Paquete <i>organization</i>	48
C.3. Paquete <i>indicator</i>	50
C.4. Clase <i>Ambit</i>	51
C.5. Clase <i>SubAmbit</i>	53
C.6. Clase <i>SubSubAmbit</i>	55
C.7. Clase <i>Indicator</i>	57
C.8. Clase <i>Evidence</i>	60
C.9. Clase <i>IndicatorsEvaluation</i>	63
C.10. Clase <i>IndicatorsEvaluationIndicatorReg</i>	67
C.11. Clase <i>IndicatorsEvaluationEvidenceReg</i>	70
C.12. Clase <i>IndicatorsEvaluationSimpleEvidenceReg</i>	73
C.13. Paquete <i>orgData</i>	77
C.14. Clase <i>City</i>	78
C.15. Clase <i>Province</i>	80
C.16. Clase <i>Region</i>	82
C.17. Clase <i>Country</i>	84
C.18. Clase <i>Address</i>	86
C.19. Clase <i>Center</i>	88
C.20. Clase <i>EvaluatorTeam</i>	90
C.21. Clase <i>ExpertSystem</i>	94
C.22. Diagrama de flujo para el inicio de sesión	100
C.23. Diagrama de flujo para el registro del usuario	101
C.24. Diagrama de flujo para el registro de la organización	103
C.25. Diagrama de flujo para el registro del equipo evaluador	105
C.26. Diagrama de flujo para la realización del test de indicadores	107
D.1. Página de descarga del instalador de Android Studio	163
D.2. Sección de términos y condiciones anterior a la descarga del instalador de Android Studio	163
D.3. El fichero de instalación ya ha sido descargado	164
D.4. Mensaje de bienvenida del instalador de Android Studio	165

D.5. Selección de componentes a instalar en el instalador de Android Studio	166
D.6. Selección del directorio de instalación	167
D.7. Selección del directorio de creación de accesos directos en el menú de inicio	168
D.8. Inicio de instalación de Android Studio	169
D.9. Finalización de la instalación de Android Studio	170
D.10. Mensaje de finalización de la instalación	171
D.11. Ventana emergente de importación de configuración	172
D.12. Mensaje de bienvenida del asistente de configuración inicial de Android Studio	173
D.13. Selección del tipo de instalación de los componentes de Android Studio	174
D.14. Selección del directorio predeterminado del JDK	175
D.15. Selección del diseño de la interfaz de Android Studio	176
D.16. Selección de componentes SDK a instalar en Android Studio	177
D.17. Advertencia por falta de componentes a instalar	177
D.18. Configuración de la memoria RAM utilizada por el emulador de Android	178
D.19. Verificación de componentes a instalar en Android Studio	179
D.20. Licencia de software del paquete SDK	180
D.21. Licencia de software del acelerador de emulación de Intel	181
D.22. Instalación de los componentes en la configuración inicial	182
D.23. Finalización del asistente de instalación de componentes	183
D.24. Pantalla de selección de proyecto	184
D.25. Pantalla de selección de actividad principal	185
D.26. Pantalla de configuración de la actividad	186
D.27. Desarrollo del proceso de instalación del soporte JDK	187
D.28. Finalización del proceso de instalación del soporte JDK	187
D.29. Interfaz de Android Studio con el proyecto Android recién creado	188
D.30. Acceso a las opciones de Android Studio	188
D.31. Pantalla de control de cuentas de GitHub	189
D.32. Pantalla de tokens de usuario antes de ir a la pantalla de generación de tokens	190
D.33. Pantalla de generación de tokens con los permisos de usuario	190
D.34. Siguientes permisos de usuario	191
D.35. Últimos permisos de usuario	191
D.36. Pantalla de tokens de usuario con el token de usuario generado	192
D.37. Pantalla de control de cuentas de GitHub lista para añadir usuario	192
D.38. Añadir token de usuario de GitHub en Android Studio	193
D.39. Cuenta de GitHub recién añadida	193

D.40.Acceso al administrador de dispositivo mediante el menú desplegable de dispositivos	194
D.41.Administrador de dispositivos	195
D.42.Selección del nuevo dispositivo virtual	195
D.43.Selección de sistema operativo para el nuevo dispositivo virtual	196
D.44.Processo de instalación del sistema operativo y de configuración del dispositivo virtual	197
D.45.Finalización del proceso de instalación del sistema operativo y de configuración del dispositivo virtual	197
D.46.Finalización del asistente de creación del dispositivo y comprobación de la configuración	198
D.47.Acceso a la configuración del nuevo commit	199
D.48.Configuración del nuevo commit	199
D.49.Soluciones a warnings durante el proceso de commit	200
D.50.Menú de publicación de cambios realizados	200
D.51.Commit y push realizados satisfactoriamente	201
D.52.Acceso a la creación de la web app desde el menú principal del portal de <i>Azure</i>	201
D.53.Configuración de los detalles del proyecto y de la aplicación web del nuevo <i>App Service</i> con <i>Azure SQL</i>	202
D.54.Configuración de la base de datos del nuevo <i>App Service</i> con <i>Azure SQL</i>	203
D.55.La web app ya ha empezado a crearse	203
D.56.Implementación en curso	204
D.57.Página oficial de descarga de <i>Visual Studio 2022</i>	204
D.58.Antes de empezar hay que preparar el instalador	205
D.59.Preparando el instalador...	205
D.60.Selección de las cargas de trabajo	206
D.61.Instalación en marcha	207
D.62.Tareas iniciales	208
D.63.Selección de plantilla	208
D.64.Configuración de proyecto	209
D.65.Selección de framework	210
D.66.Ir a publicar	211
D.67.Agregar perfil de publicación	211
D.68.Selección de publicación en <i>Azure</i>	212
D.69.	213
D.70.Selección de web service	214
D.71.Despliegue completado	215

Índice de tablas

A.1. Tiempo invertido en cada actividad	23
B.1. CU-1 Realizar test de indicadores	33
B.2. CU-2 Muestra de resultados	34
B.3. CU-3 Inicio de sesión en la app	35
B.4. CU-4 Gestión de organizaciones	35
B.5. CU-5 Gestión de organizaciones	36
B.6. CU-6 Gestión de registros de usuarios	37
B.7. CU-7 Gestión de registros de usuarios	37
B.8. CU-8 Gestión de registros de usuarios	38
B.9. CU-9 Gestión de equipos evaluadores	39
B.10.CU-10 Gestión de centros o servicios	40

Apéndice A

Plan de Proyecto Software

A.1. Edición del informe

En cuanto a la edición del informe se han tenido en cuenta diferentes aspectos para poder escoger el formato que debe tener informe, como por ejemplo la personalización del documento, la preocupación por los márgenes de cada una de las páginas, el control de la ortografía, la introducción de fórmulas matemáticas, de tablas, de imágenes y de gráficos y el acabado profesional de este informe. Por tanto, dichos aspectos son los expuestos en los sub-apartados de este punto.

A.2. Formato del informe

El editor de documentos que se ha decidido utilizar para realizar dicho informe es *LaTeX* por encima de editores de documentos convencionales como *OpenOffice Write* o *Microsoft Word*. El motivo por el cual se ha decidido utilizar *LaTeX* es por su gran cantidad de funcionalidades y por la libertad con la que se puede diseñar el documento en todos los aspectos mencionados con anterioridad. Para poder crear los documentos pdf en (*LaTeX*) es necesaria la instalación de un compilador. Existen diferentes opciones para poder realizar la compilación del documento, como es el caso de *Overleaf*, el cual es un editor en línea de código en *LaTeX*, de uso compartido y con compilador integrado. En el caso de este informe, se ha utilizado el compilador *MiKTeX*, que es un compilador de libre distribución el cual permite instalar todos los paquetes que el usuario necesite para su documento. Como editor de *LaTeX* se ha utilizado *Microsoft Visual*

Studio Code. Se ha escogido ese editor por encima del editor predefinido de *MikTeX* (de nombre *TexWorks*), puesto que es un editor bastante más completo, ya que se puede editar el código fuente de la aplicación además del propio documento, puesto que soporta la gran mayoría de los lenguajes de programación más utilizados de la actualidad, aparte de dar soporte al propio *LaTeX* y a su compilador *MikTeX*.

A.3. Comprobación de la ortografía del informe

Gracias al editor *Microsoft Visual Studio Code* es posible agregar complementos que faciliten el soporte a todo tipo de lenguajes y que faciliten también un buen uso para todo tipo de usuarios. Uno de estos complementos es el que se ha utilizado para facilitar la comprobación de la ortografía de todo tipo de lenguajes, entre ellos *LaTeX*, el cual recibe el nombre de ***Code Spell Checker***, para el idioma español, el cual se instala de una manera muy sencilla en *Visual Studio Code*, presionando el botón de Instalar dentro de la página web del *Marketplace*, el cual está disponible también para su búsqueda dentro del propio editor. Posteriormente, para activarlo se puede hacer de diferentes maneras:

1. El primer método consiste en presionar el botón *F1* y posteriormente escribir el comando *Enable Spanish Spell Checker Dictionary* y se presiona el botón *Enter* para que empiece a trabajar.
2. El segundo método consiste en ir al menú de la barra superior de tareas *Ver* y posteriormente al sub-menú *Paleta de comandos*. Posteriormente se escribe el comando *Enable Spanish Spell Checker Dictionary* y se presiona el botón *Enter* para que empiece a trabajar.

La versión original de este complemento realiza las comprobaciones ortográficas para el idioma **inglés**, el cual se instala en *Visual Studio Code* de la misma manera, presionando el botón de Instalar dentro de la página web del *Marketplace*. Al tener instalada la versión en español, que es una extensión de la versión original, no precisa instalarlo después de la versión en español.

A.4. Precedente

Para dar contexto a la aplicación desarrollada se tiene que tener en cuenta el punto de partida de este proyecto, el cual se trata de un programa *.mdb*,

el cual está implementado sobre *Microsoft Access*, el cual estaba grabado en un CD-ROM. Dicho programa recibe el nombre de *Guía de indicadores de calidad para Organizaciones que presentan apoyo a personas con Trastorno del Espectro Autista*, cuyo acrónimo es **OTEA**.

OTEA. Guía de indicadores de calidad para Organizaciones que presentan apoyo a personas con Trastorno del Espectro Autista

OTEA, como se ha expuesto con anterioridad es una aplicación implementada en Access implementada por la *Fundación Miradas* en el año 2009. Dicha aplicación estaba implementada en un CD-ROM, el cual debía introducirse en una bandeja conectada al equipo para que posteriormente pudiese ejecutar la aplicación. El menú principal de dicha aplicación es el siguiente:



Figura A.1: Menú principal de OTEA

El menú principal de la aplicación tiene una apariencia bastante sencilla para la época, con diferentes apartados para realizar el diagnóstico de la forma indicada. Dichos apartados son los siguientes:

- **Datos de la organización o servicio:** Este apartado consiste en un formulario en el que se introducen los datos de la organización

evaluadora, los datos del equipo evaluador y las fechas en las que se realizaron las cuatro evaluaciones. Su interfaz es la siguiente:

The screenshot shows a software window titled "tbl_datos_centro" with a background illustration of a person's head. The form is titled "Datos de la organización o servicio". It contains the following fields:

- CENTRO O SERVICIO:** [Redacted]
- DIRECCIÓN:** [Redacted]
- TELÉFONO:** [Redacted]
- E-MAIL:** [Redacted]
- OTROS DATOS DE INTERÉS:** Terapias para niños y orientación a padres
- CONSULTOR/A EXTERNO/A:** Miguel Gómez Gentil
- ORGANIZACIÓN A LA QUE PERTENECE:** Fundación Miradas (Burgos)
- DIRECTOR/A DE LA ORGANIZACIÓN:** [Redacted]
- PROFESIONAL DE ATENCIÓN DIRECTA:** [Redacted]
- FAMILIAR:** [Redacted]
- OTROS PARTICIPANTES:** [Redacted]
- CONSTITUCIÓN DEL EQUIPO EVALUADOR:** 08/09/2022
- VISITA AL CENTRO:** 14/11/2022
- PRIMERA SESIÓN DE VALORACIÓN:** 16/01/2023
- SEGUNDA SESIÓN DE VALORACIÓN:** 14/04/2023
- TERCERA SESIÓN DE VALORACIÓN:** 14/07/2023
- CUARTA SESIÓN DE VALORACIÓN:** 16/10/2023

At the bottom are buttons for "Guardar" (Save), "Cancelar" (Cancel), and a print icon. The footer reads "OTEA, José Luis Cuesta Gómez, jl.cuesta@ubu.es".

Figura A.2: Apartado de introducción de datos de la organización o del servicio

Como se puede comprobar en la captura anterior, los datos a rellenar son los siguientes:

- **Centro o servicio:** En este campo se introduce el nombre del centro o servicio al que se va a evaluar.
- **Dirección:** En este campo se introduce la dirección donde se ubica el centro a evaluar.
- **Teléfono:** En este campo se introduce el número telefónico de la organización a la que se va a evaluar.

- **Email:** En este campo se introduce la dirección de correo electrónico del centro a evaluar.
- **Otros datos de interés:** En este campo se introduce información adicional sobre el centro o servicio a evaluar
- **Consultor/a externo/a:** En este campo se introduce el nombre completo de la persona que lidera el equipo que va a realizar la valoración a ese centro o servicio en concreto.
- **Organización a la que pertenece:** En este campo se introduce la organización a la que pertenece el consultor externo que lidera el equipo de valoración
- **Director/a de la organización:** En este campo se introduce el nombre del director o directora del centro a evaluar.
- **Profesional de atención directa:** En este campo se introduce el nombre del responsable del profesional del centro a evaluar que actúa de intermediario entre la organización y el equipo evaluador.
- **Familiar:** En este campo se introduce el nombre del familiar que conoce la organización o servicio a evaluar.
- **Otros participantes:** En este campo se introducen los nombres de los demás componentes del equipo evaluador.
- **Fecha de constitución del equipo evaluador:** En este campo se introduce el día en el que se constituyó el equipo evaluador.
- **Fecha de visita al centro:** En este campo se introduce el día en el que se realizó la primera visita al centro a evaluar
- **Fechas de las correspondientes sesiones de evaluación:** En estos campos se introducen todas las fechas de evaluación

Al igual que con otros aspectos del programa, se dispone de una interfaz bastante intuitiva para la época, ayudando a introducir correctamente los diferentes tipos de datos, como el caso de las fechas, obligando a forzar la introducción de ese tipo de dato. Un problema bastante importante a la hora de introducir los datos es la escasa cantidad máxima de caracteres en algunos campos que lo necesitan, como es el caso de la dirección de la organización y de los otros datos de interés, puesto que, tal y como se ha podido comprobar, no se ha podido introducir correctamente la dirección completa de, en este caso, el *Centro de Organización Neurológica Neocortex* de Majadahonda (Madrid), ya que por la longitud de la misma se han tenido que abbreviar

el tipo de vía y la palabra bloque, aparte de que no se ha podido introducir el código postal ni el municipio donde se encuentra, siendo este último introducido junto con en el nombre de la organización. Además de eso, no existe control sobre el tipo de datos introducidos en cada uno de los campos, puesto que toma todos los datos como tipo string.

- **Comenzar test:** En este apartado se realiza el test de indicadores de la organización a evaluar. El test consta de 68 indicadores, con cuatro evidencias cada uno. Un indicador es una categoría que abarca diferentes aspectos de la organización que alberga a personas con Trastorno del Espectro Autista, siendo cada una de las cuatro evidencias un aspecto específico relacionado con dicha categoría de evaluación.

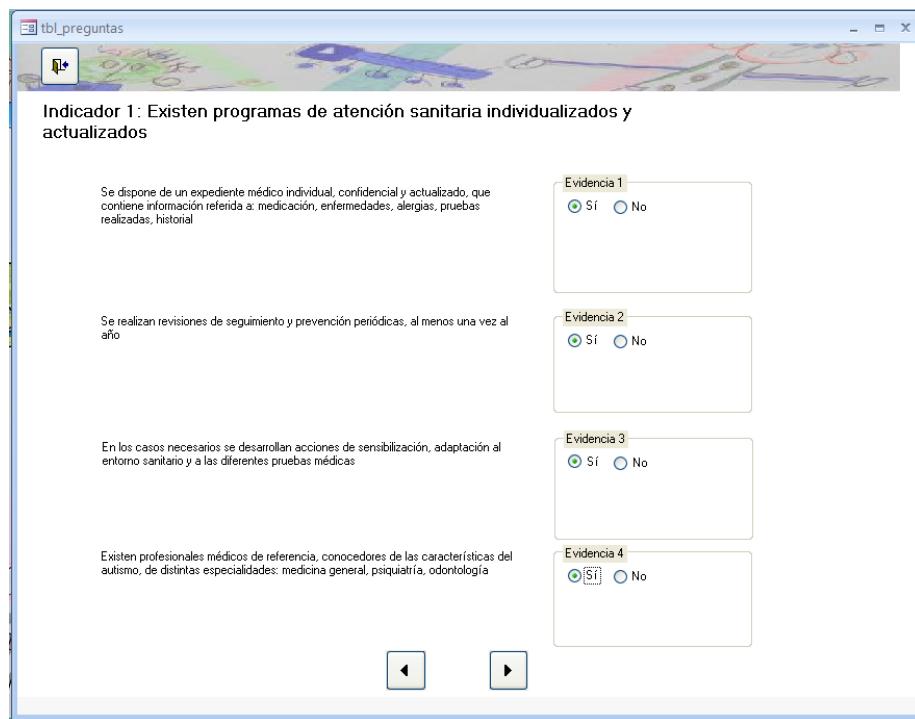


Figura A.3: Apartado de realización del test de indicadores. Primer indicador

Como se puede comprobar en la captura anterior, la interfaz del apartado del test tiene un formato bastante intuitivo para la época. Pero a medida de que se van analizando más indicadores, se vuelve bastante pesado en su uso, puesto que hay que mover el ratón hasta la respuesta seleccionada. Cuando ya se han evaluado los 68 indicadores,

aparece la siguiente ventana flotante invitando al usuario a que guarde los datos introducidos:

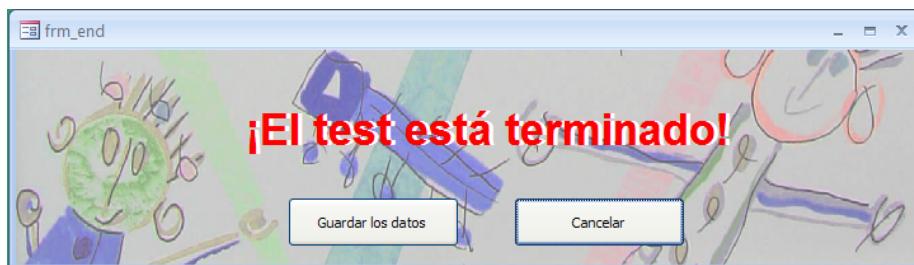


Figura A.4: Ventana flotante de finalización del test de indicadores antes de guardar los datos

Para finalizar se presiona en *Guardar datos* y posteriormente en *Sair*.

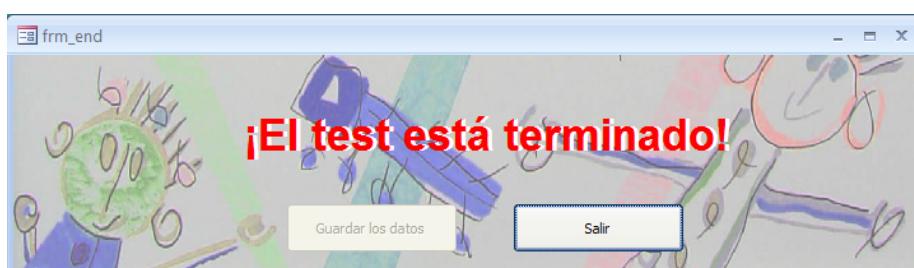


Figura A.5: Ventana flotante de finalización del test de indicadores después de guardar los datos

- **Gráfico del servicio u organización:** Este gráfico se utiliza como muestra de resultados del test de indicadores y evidencias que se realiza con anterioridad. Este gráfico es una tabla en la que las filas reflejan el interés que tiene el indicador y las columnas reflejan la clasificación de cada indicador dependiendo de cada aspecto a evaluar de esa organización. Cada indicador es identificado como un cuadrado en el que se muestra el número del mismo y el color verde, amarillo o rojo dependiendo del nivel mejor, promedio o peor de cumplimiento de cada indicador.

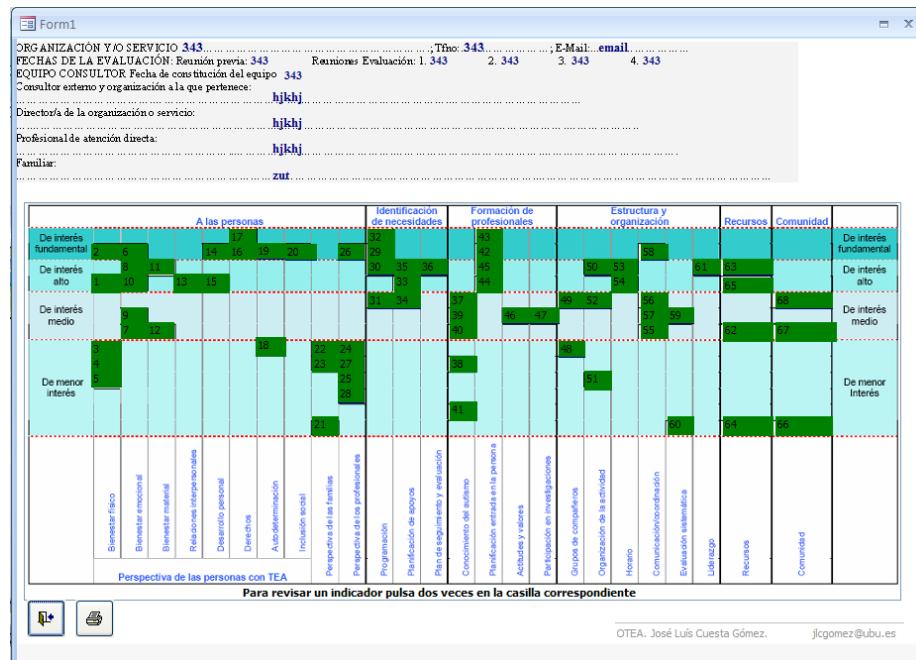


Figura A.6: Gráfico de muestra de resultados del servicio u organización

Se puede comprobar en este gráfico que no aparece la información introducida en el apartado de Datos de Información y Servicio, sino que se muestran valores aleatorios para cada uno de los campos. También se puede comprobar que esta evaluación de los indicadores ha sido perfecta como refleja cada uno de los cuadrados de los indicadores, los cuales son rellenos de color verde. Si se da doble clic encima de cualquiera de los cuadrados del gráfico, se puede mostrar información de dicho indicador, la cual se muestra en la interfaz del apartado de realización de test.

- **Perfil general:** En este apartado se obtiene un informe general con la información obtenida desde Datos de organización o servicio, con la puntuación obtenida en el test de indicadores y en el rango en el que se ubica.

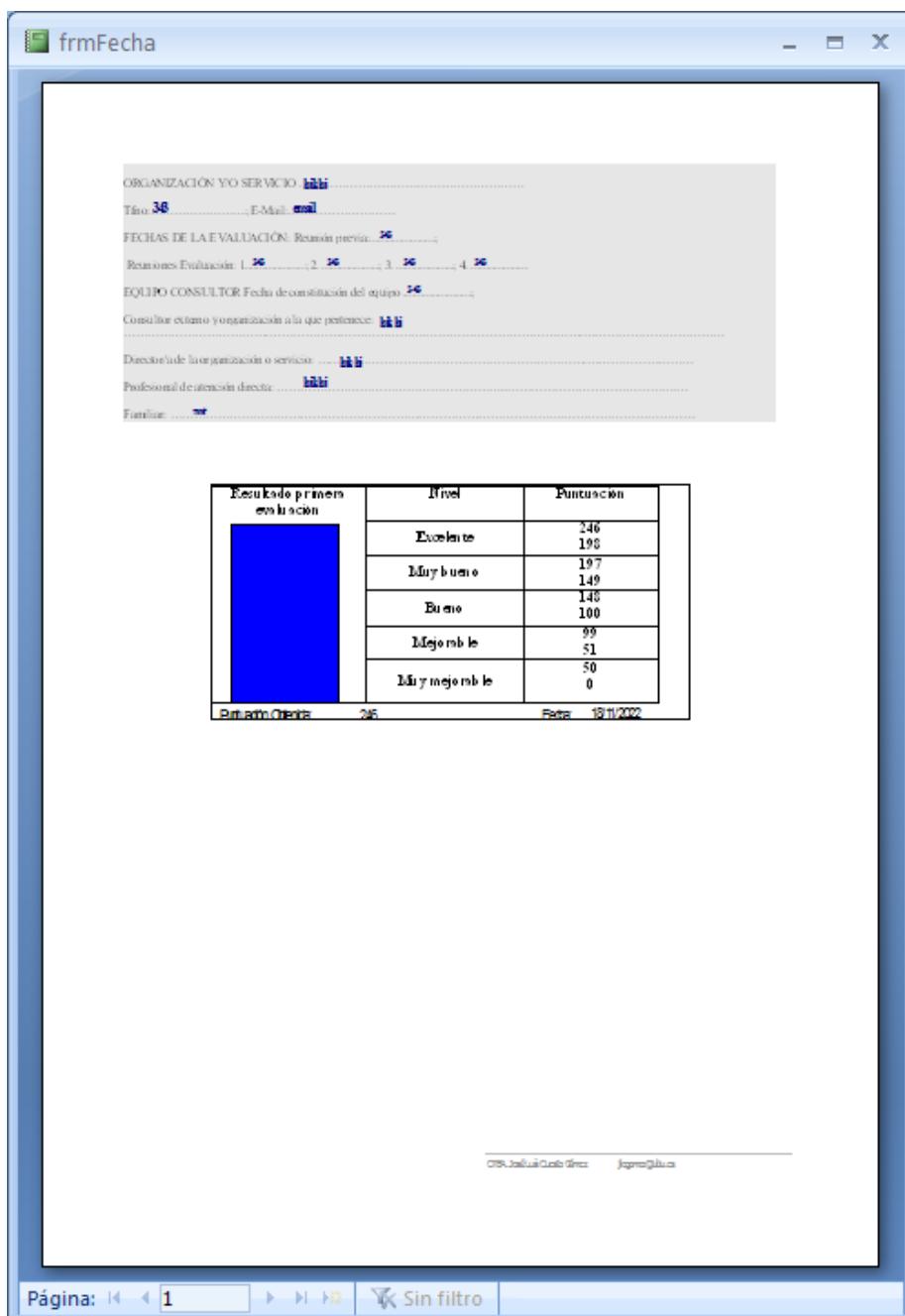


Figura A.7: Informe con la puntuación del test de indicadores

- **Informe final:** Este apartado es similar al de Datos de la organización o servicio, con los mismos campos que se han introducido con

anterioridad, con la diferencia de que hay dos cuadros de relleno de párrafo, los cuales son

- **Observaciones:** En este apartado se escriben las observaciones que se han tenido en cuenta durante las cuatro sesiones de valoración.
- **Informe final:** En este apartado se escriben las conclusiones de todas las sesiones de evaluación, así como algún apunte sobre el resultado del test de indicadores.

Dicho apartado tiene la siguiente interfaz:

Datos de la organización o servicio

CENTRO O SERVICIO	CON Neocortex (Majadahonda, Madrid)
DIRECCIÓN	Ctra. Boadilla del Monte, 8-44 Local 1 Bl.3
TELÉFONO	916390390
E-MAIL	cita@neocortex.com
OTROS DATOS DE INTERÉS	Terapias para niños y orientación a padres
CONSULTOR/A EXTERNO/A	Miguel Gómez Gentil
ORGANIZACIÓN A LA QUE PERTENECE	Fundación Miradas (Burgos)
DIRECTOR/A DE LA ORGANIZACIÓN	Mª Jesús López Juez
PROFESIONAL DE ATENCIÓN DIRECTA	Mª Jesús López Juez
FAMILIAR	Sonia Rodríguez Cano
OTROS PARTICIPANTES	Sonia Rodríguez Cano
CONSTITUCIÓN DEL EQUIPO EVALUADOR	08/09/2022
VISITA AL CENTRO	14/11/2022
PRIMERA SESIÓN DE VALORACIÓN	16/01/2023
SEGUNDA SESIÓN DE VALORACIÓN	14/04/2023
TERCERA SESIÓN DE VALORACIÓN	14/07/2023
CUARTA SESIÓN DE VALORACIÓN	16/10/2023

Observaciones:

En este centro se realizan terapias que estimulan el desarrollo cerebral de los niños. Dichas terapias constan de programas tanto fisiológicos y sensoriales como físicos e intelectuales. Para diseñar de forma individualizada los programas que cada niño necesita, el niño y su familia realizan una visita de dos días al centro. El primer día es dedicado a la evaluación, audiometría e historial del menor por la mañana y la realización de la revisión médica y el resumen de todo lo realizado durante el día por la tarde. Al día siguiente se realiza una conferencia de formación y se presenta el programa fisiológico y sensorial por la mañana y por la tarde se presentan los programas físico e intelectual. Dichas visitas son realizadas cada 6 meses como seguimiento de la evolución del niño.

Informe Final:

La dinámica del centro hacia los niños con Trastorno del Espectro Autista, al tratarse de visitas de dos días, cubren al completo todas las necesidades que tiene el niño a tratar, desde las fisiológicas y sensoriales hasta las físicas e intelectuales. Al tener dos días seguidos de visita, de los cuales se dedican bastantes horas a todo el proceso de evaluación, se realizan de manera personalizada los programas que el niño necesita y las pautas que deben seguir los padres y/o tutores del menor en

OTEA. José Luis Cuesta Gómez. jlcgomez@ubu.es

Figura A.8: Apartado de informe final con los datos del centro, las observaciones y las conclusiones

Para obtener el PDF del informe, no es posible hacerlo de forma nativa desde Windows XP, puesto que no posee la herramienta *Microsoft Print to PDF* al ser un sistema operativo bastante antiguo. Para solucionar este problema se ha descargado el programa *Print to PDF*

Pro, de la desarrolladora *Traction Software Limited*, cuya instalación sigue los siguientes pasos:

- En primer lugar, se accede a la página web de descarga del programa y se presiona el botón *Download* para descargar el fichero de instalación:

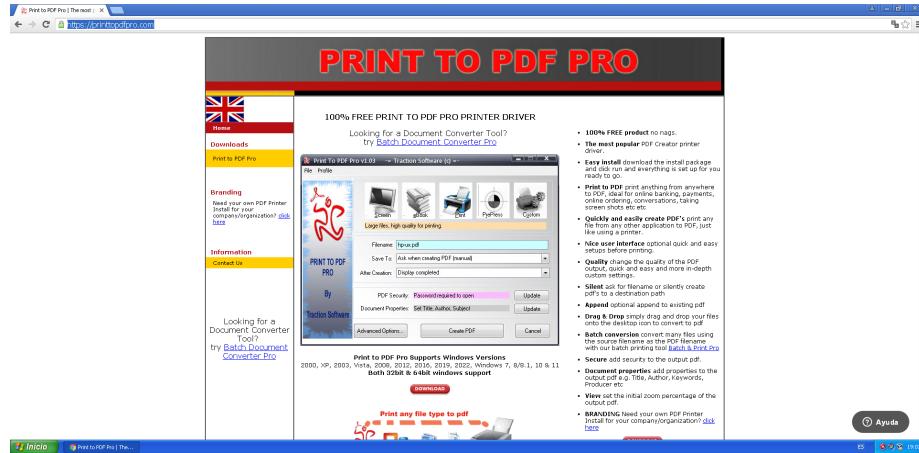


Figura A.9: Página principal de *Print to PDF Pro*

- Posteriormente, se ejecuta el fichero de instalación *PrintToPDF-ProSetup.exe*, el cual ejecuta el asistente de instalación del mismo:
 1. Al ejecutarse el fichero, da un mensaje de bienvenida donde también se recomienda que se cierren otros programas antes de ejecutar el asistente de instalación.

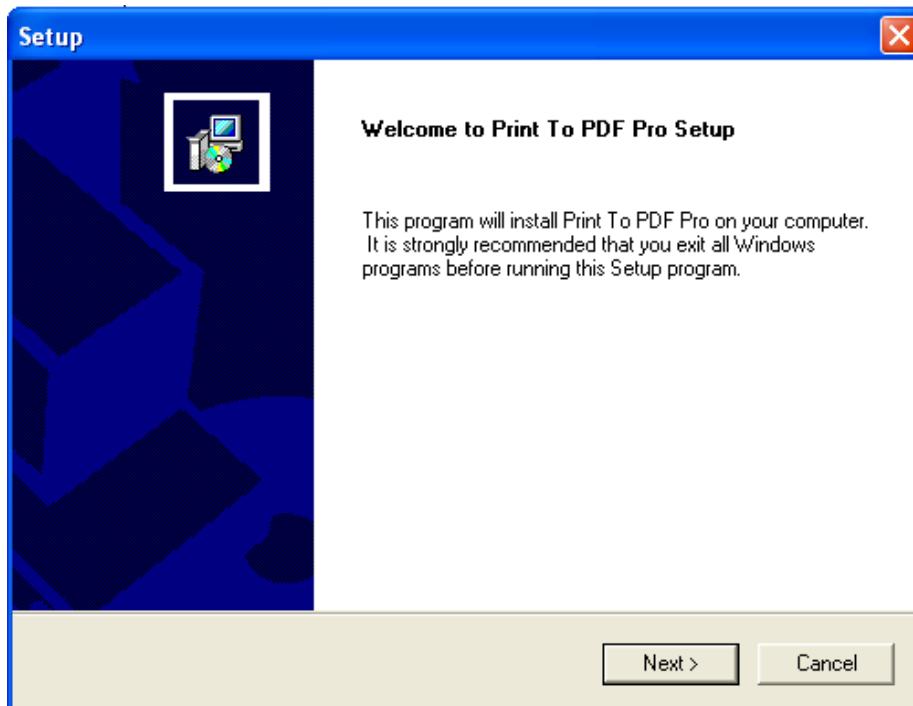


Figura A.10: Mensaje de bienvenida del programa de instalación de *Print to PDF Pro*

2. Tras presionar el botón *Next*, se tiene que aceptar el contrato de licencia para poder instalar *Print to PDF Pro* correctamente. El propio asistente recomienda también la lectura de dicho contrato antes de su aceptación.

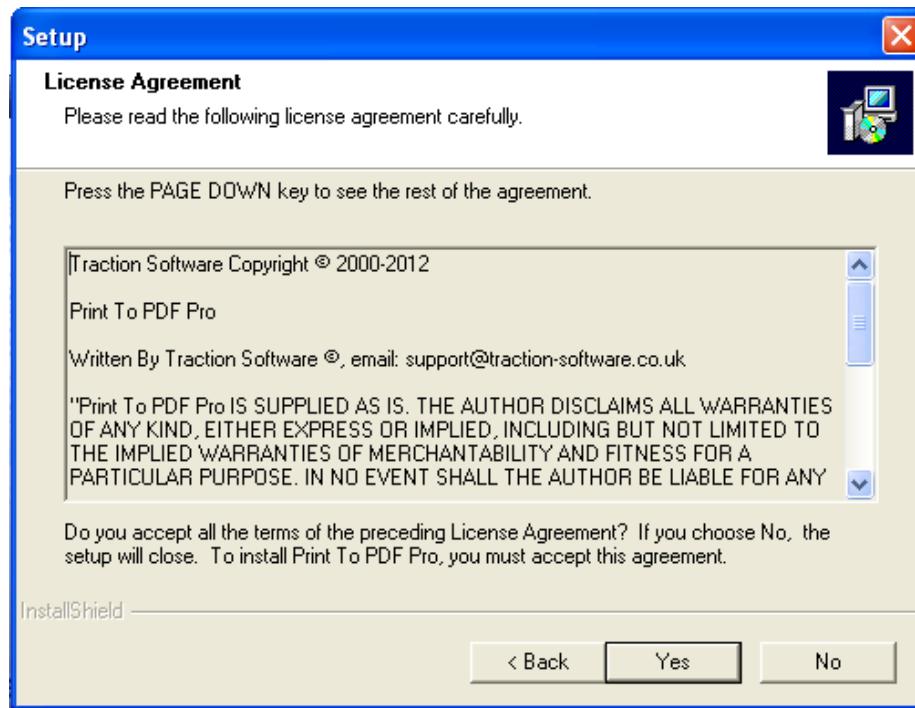


Figura A.11: Contrato de licencia de *Print to PDF Pro*, de obligada aceptación para la instalación del programa

3. Más adelante se procede a elegir el directorio donde se instalará el programa. Se puede optar por mantener el directorio por defecto, el cual es *C:/Program Files/Traction Software/Print to PDF Pro*, o en su defecto elegir otro directorio diferente donde instalarlo.

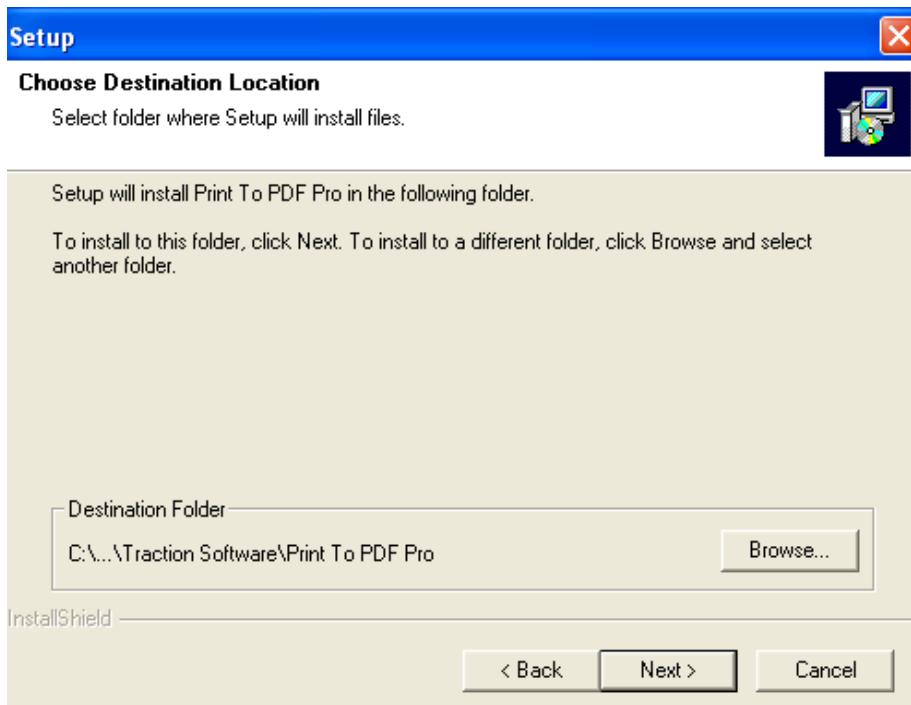


Figura A.12: Selección del directorio de instalación de *Print to PDF Pro*

4. Tras haber elegido el directorio de instalación se procede a la instalación del programa y, cuando ésta concluya, se da la opción al usuario la opción a registrar el producto, introduciendo su nombre, el nombre de la compañía donde trabaja y la dirección de correo electrónico.

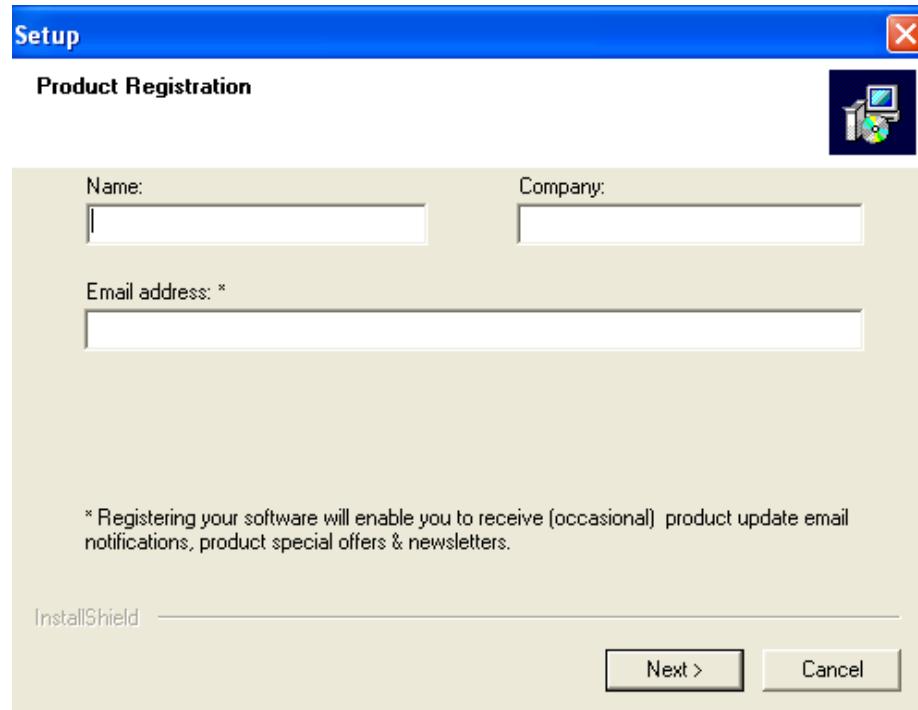


Figura A.13: Registro de *Print to PDF Pro*, de cumplimiento opcional.

5. Al finalizar el asistente de instalación se da la opción al usuario de imprimir un fichero de prueba en formato *.pdf* tras cerrar el programa de instalación. En caso de que se deje marcada la casilla, como se muestra a continuación, se procede a la creación del fichero:

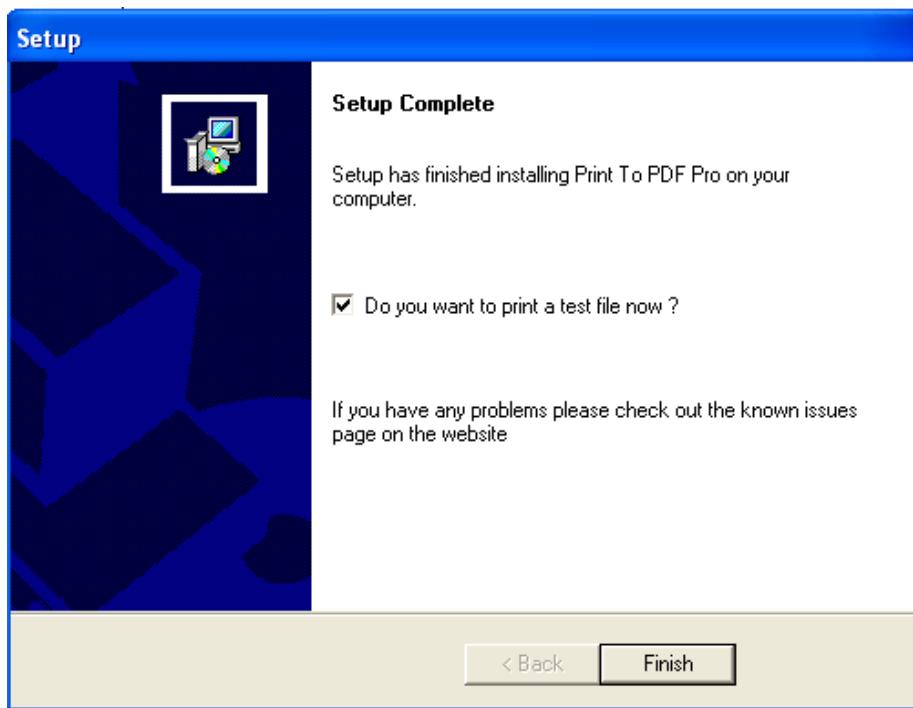


Figura A.14: Finalización del asistente de instalación de *Print to PDF Pro*

Cuando dicho fichero se haya terminado de crear, mostrará esta ventana emergente donde se solicita al usuario la visualización del fichero de prueba. En caso de seleccionar *Yes*, se mostrará el contenido de dicho fichero de prueba:

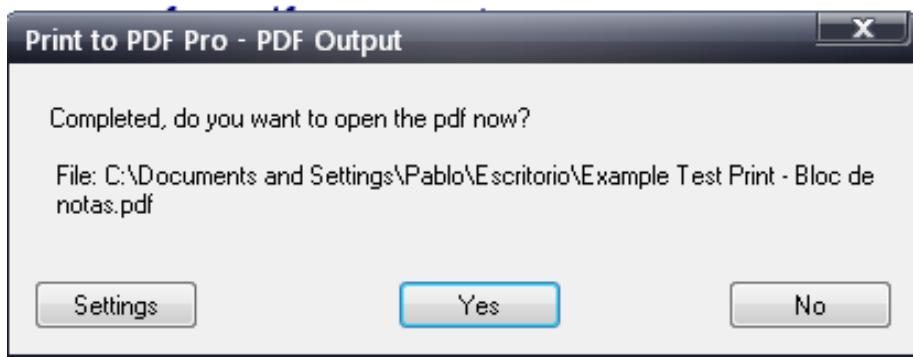


Figura A.15: Ventana emergente donde se pregunta si se desea visualizar el fichero recién generado

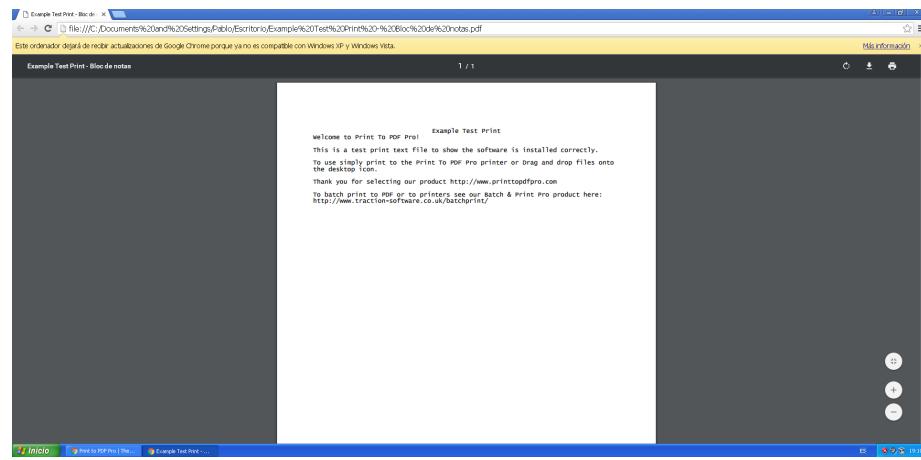


Figura A.16: Fichero de prueba de *Print to PDF Pro*

Por tanto, tras haber instalado este programa y haber comprobado que *Print to PDF Pro* es la impresora predefinida del sistema operativo, se procede a la impresión a un fichero .pdf del Informe Final, el cual se obtiene presionando el botón de la impresora de la parte inferior de la ventana de **Informe Final**

tbl_datos_centro

E-MAIL	
OTROS DATOS DE INTERÉS	Terapias para niños y orientación a padres
CONSULTOR/A EXTERNO/A	Miguel Gómez Gentil
ORGANIZACIÓN A LA QUE PERTENECE	Fundación Miradas (Burgos)
DIRECTOR/A DE LA ORGANIZACIÓN	
PROFESIONAL DE ATENCIÓN DIRECTA	
FAMILIAR	
OTROS PARTICIPANTES	
CONSTITUCIÓN DEL EQUIPO EVALUADOR	08/09/2022
VISITA AL CENTRO	14/11/2022
PRIMERA SESIÓN DE VALORACIÓN	16/01/2023
SEGUNDA SESIÓN DE VALORACIÓN	14/04/2023
TERCERA SESIÓN DE VALORACIÓN	14/07/2023
CUARTA SESIÓN DE VALORACIÓN	16/10/2023

Observaciones:

En este centro se tienen realizan terapias que estimulan el desarrollo cerebral de los niños. Dichas terapias constan de programas tanto fisiológicos y sensoriales como físicos e intelectuales. Para diseñar de forma individualizada los programas que cada niño necesita, el niño y su familia realizan una visita de dos días al centro. El primer día es dedicado a la evaluación, audiometría e historial del menor por la mañana y la realización de la revisión médica y el resumen de todo lo realizado durante el día por la tarde. Al día siguiente se realiza una conferencia de formación y se presenta el programa fisiológico y sensorial por la mañana y por la tarde se presentan los programas físico e intelectual. Dichas visitas son realizadas cada 6 meses como seguimiento de la evolución del niño.

Informe Final:

La dinámica del centro hacia los niños con Trastorno del Espectro Autista, al tratarse de visitas de dos días, cubren al completo todas las necesidades que tiene el niño a tratar, desde las fisiológicas y sensoriales hasta las físicas e intelectuales. Al tener dos días seguidos de visita, de los cuales se dedican bastantes horas a todo el proceso de evaluación, se realizan de manera personalizada los programas que el niño necesita y las pautas que deben seguir los padres y/o tutores del menor en todos los aspectos a tratar. Todos los aspectos de los indicadores están bien cubiertos con creces, a seguir así.

Guardar Cancelar

OTEA, José Luis Cuesta Gómez, jlcgomez@ubu.es

Figura A.17: Informe final con el botón de la impresora seleccionado

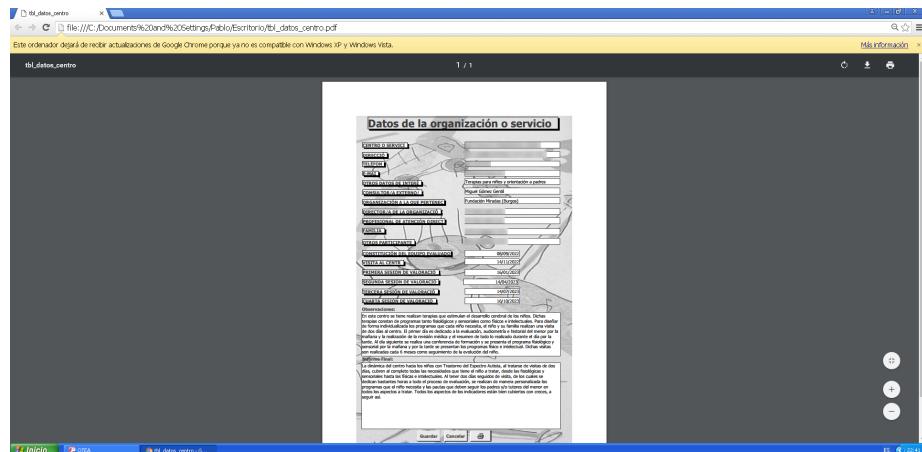


Figura A.18: Informe en PDF generado por el programa en *Microsoft Access*

Requisitos de uso del programa

Para poder ejecutar este programa en un equipo moderno, se ha tenido que instalar una máquina virtual de *Oracle VirtualBox* cuyo sistema operativo sea *Windows XP*, todo ello debido a la antigüedad de dicho programa. También se ha tenido que instalar una versión antigua del paquete de aplicaciones de ofimática *Microsoft Office*, en concreto se ha instalado *Microsoft Office 2007*, ya que tiene un mejor soporte para la aplicación de *OTEA*.

Diferencias entre *Azure* y las aplicaciones de bases de datos locales

Las diferencias que tiene *Azure* con respecto a las aplicaciones de gestión de bases de datos a nivel local, como es el caso de *Microsoft Access* o *OpenOffice Database*, son las siguientes:

- En las aplicaciones de bases de datos locales es preciso utilizar un ordenador con un fichero *.accdb* en *Microsoft Access* o un fichero *.odb* en *OpenOffice Database*, el cual aloje todos los registros de los indicadores y sus respectivas incidencias, cuya difusión depende de la cantidad de personas que tengan ese fichero. En cambio, con Microsoft Azure, la difusión es más sencilla puesto que no se necesita un fichero en cada uno de los dispositivos, ya que al alojarse los registros de los datos y de las correspondientes incidencias en la nube, permite de mejor manera la implementación en diferentes dispositivos, aparte de que solo el administrador tiene acceso a la base de datos.
- En una aplicación que se apoya en *Azure* es necesaria una conexión a internet para poder cargar los datos correctamente al servidor, al igual que para realizar todas las demás operaciones en las que esté involucrada la base de datos, por lo tanto si estuviese caído el servidor donde se ha implementado Azure, no se tendría el comportamiento esperado en la aplicación. En cambio, en el caso de la base de datos en *Microsoft Access* o en *OpenOffice Database*, como el comportamiento de la base de datos depende de que esté alojado su correspondiente fichero *.accdb* o *.odb* en una cantidad determinada de dispositivos, no se tendría esta problemática, salvo en el hipotético caso en que ninguno de los dispositivos que cuenten con el fichero de la base de datos se encuentre operativo.
- En el caso de implementar una base de datos en *Azure*, se puede implementar de mejor manera en una aplicación como la de la Fundación

Miradas, puesto que se espera que dicha aplicación reciba y envíe transacciones del cliente, que es el encargado de la Fundación Miradas que evalúa el correcto cumplimiento de los indicadores y de sus respectivas incidencias o el representante de la asociación de ayuda a la discapacidad que está siendo evaluada que comprueba los diferentes resultados realizados de las diferentes test, hacia el servidor que se encarga de alojar cada uno de los datos de las asociaciones evaluadas y de cada uno de los mencionados diagnósticos, proporcionando inmediatez y automatización en el proceso de muestra de resultados para todos los interesados mencionados con anterioridad. En cambio, esta tarea es más tediosa en la implementación original de la aplicación en Access, puesto que los resultados obtenidos sólo se mostrarían de forma inmediata en el equipo que corre dicha aplicación, por lo que sería necesario transportar el informe resultante manualmente, ya sea mediante un dispositivo de almacenamiento físico externo o por correo electrónico.

- *Azure*, al estar enfocada a alojar grandes cantidades de datos por parte de empresas, no dispone de una versión gratuita de forma permanente, sino que sólo unos cuantos servicios son gratuitos, mientras que hay ciertos que también son gratuitos, pero únicamente durante doce meses, mientras que otros se obtienen mediante las diferentes suscripciones de las que dispone *Azure*. En cambio, sucede lo contrario con las aplicaciones de bases de datos a nivel local, como *Microsoft Access*, cuya versión gratuita se puede encontrar en OneDrive como aplicación web, como con *OpenOffice Database*, el cual forma parte del paquete de ofimática de libre distribución *OpenOffice*. En el caso del paquete de *Microsoft Office*, donde se incluye *Microsoft Access*, también es un software de pago el cual dispone de licencias desde un mes hasta los doce.
- En *Azure* se garantiza la seguridad de los datos proporcionados gracias al cifrado de los datos en reposo, el cual se produce en tres niveles:
 - **A nivel de almacenamiento en el servidor** el servicio *Azure Storage*, el cual se encarga del almacenamiento de los datos, realiza un encriptado del servicio *SSE*, concretamente los datos se cifran y descifran de forma transparente mediante el cifrado *AES* de 256 bits, uno de los cifrados de bloques más sólidos que hay disponibles, y son compatibles con *FIPS 140-2*.
 - **A nivel de cliente**, la correspondiente biblioteca de *Azure Blob Storage* usa *AES* para cifrar los datos del usuario. Hay dos

versiones de cifrados de cliente disponibles en la biblioteca de cliente:

- La versión 2 utiliza el modo *Galois/Contador (GCM)* con *AES*.
- La versión 1 utiliza el modo *Cipher Block Chain (CBC)* con *AES*.
- **A nivel de disco duro del sistema operativo**, permite cifrar los discos del sistema operativo y los discos de datos usados por una máquina virtual *IaaS*. Dicho proceso se encarga de realizarlo *Azure Disk Encryption*, el cual ayuda a custodiar y proteger los datos con el objetivo de cumplir los compromisos de cumplimiento y seguridad. Usa la característica *DM-Crypt* de Linux para proporcionar cifrado de volumen tanto a los discos de datos como a los del sistema operativo de máquinas virtuales (VM) de *Azure* y se integra con *Azure Key Vault* para ayudarle a controlar y administrar las claves y los secretos del cifrado de disco.

En cambio, con las bases de datos locales, la seguridad depende de quien disponga el fichero correspondiente y del correcto uso que tenga del mismo.

A.5. Planificación temporal

Actividad	Período	Tiempo
Búsqueda de Trabajos de Final de Grado de Referencia	26/10/2022 - 16/11/2022	10 horas
Pruebas y detección de mejoras de la aplicación en Access	15/11/2022 - 17/01/2023	28 horas
Formación en Azure	9/6/2023 - 8/7/2024	200 horas
Análisis del entorno	25/01/2023 - 10/6/2023	40 horas
Diseño de la aplicación	25/01/2023 - 10/6/2023	90 horas
Pruebas de la aplicación	10/6/2023 - 8/7/2024	200 horas
Actualización de GitHub	12/12/2022 - 8/7/2024	12 horas
Desarrollo de la memoria y de los anexos	26/10/2022 - 8/7/2024	310 horas

Tabla A.1: Tiempo invertido en cada actividad

A.6. Estudio de viabilidad

Viabilidad económica

En cuanto a la viabilidad económica de este proyecto, cabe resaltar que esta se basa única y exclusivamente en los costes de mantenimiento y escalado del servidor. Podemos comprobar esa información en el propio portal de Azure, para ser más precisos, en el apartado de información general de la suscripción que se tiene activa por parte del usuario:

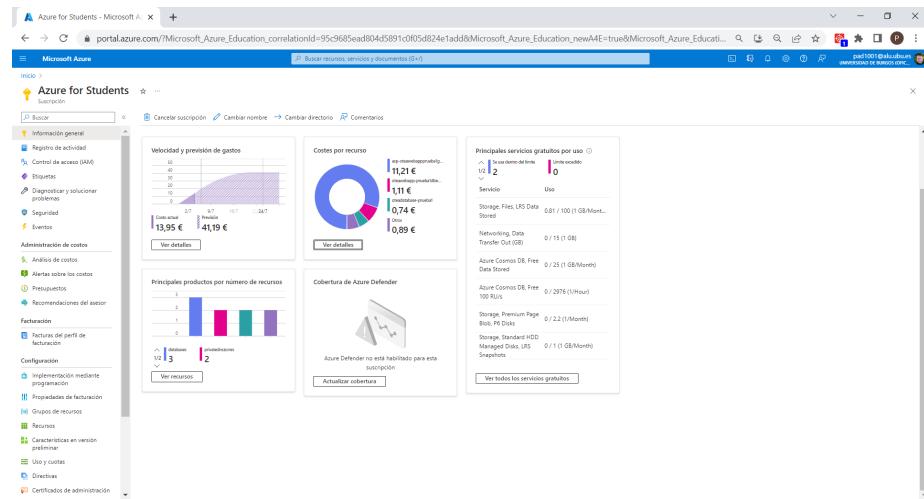


Figura A.19: Información general de la suscripción en Azure

Como se puede comprobar en esta pantalla, el análisis del coste puede medirse mediante diferentes métricas:

- **Velocidad y previsión de gastos:** Se trata de un gráfico acumulativo con los gastos totales que se han tenido en la suscripción. En la figura anterior aparecen costes muy bajos, ya que se está utilizando la base de datos más básica, la de la versión de prueba del servicio web, que se ha ido manteniendo con los 200\$ de crédito gratuito disponibles (*Véase apartado de técnicas y herramientas de la memoria*).
- **Coste por recurso:** Es un gráfico circular el cual muestra el porcentaje de gasto total para cada uno de los recursos utilizados.

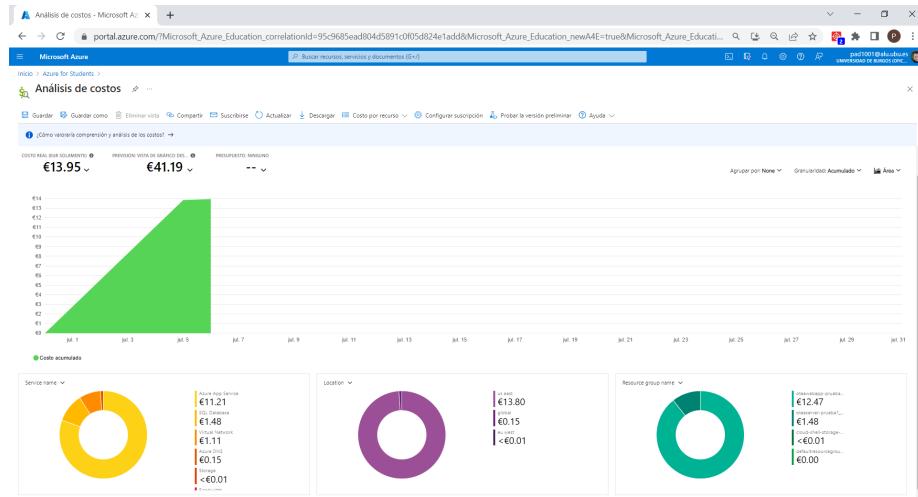


Figura A.20: Velocidad y previsión de gastos junto con los costes por cada recurso

Como se ha podido comprobar, la previsión de costes ayuda a conocer la tendencia del gasto a partir del gasto acumulado total, mientras que los recursos del servidor de la base de datos y la propia base de datos son los que más consumen. Estos datos son de una versión básica, pero con mayor escalabilidad y prestaciones, el gasto se amplía y a su vez se mantienen esos porcentajes.

Cabe resaltar también que Azure dispone de diferentes herramientas las cuales se utilizan para realizar buenas prácticas en estudios de viabilidad a nivel económico. La propia suscripción da la posibilidad de programar alertas si se supera un cierto nivel de gasto, al igual que también dar la posibilidad de generar presupuestos que ayuden a manejar dichos costes. Todas estas herramientas pertenecen al servicio de *Cost Management* de Azure, disponibles para cualquier tipo de suscripción.

Esta burocracia en el caso de *Fundación Miradas* es evitada, ya que *Fundación Miradas* cuenta con un acuerdo de patrocinio con Microsoft, del cual recibe un montante anual suficiente para mantener todos los servicios de Azure que este entorno necesita para funcionar.

Por lo tanto, un análisis coste-beneficio en nuestro caso no sería necesario, puesto que con el dinero que recibe *Fundación Miradas* para el mantenimiento de todo el entorno de Azure, no supondría ningún sobrecoste para dicha organización, por lo que es difícil hacer una estimación del beneficio económico que supondría el proyecto, más allá de una posible comercialización en Google Play como aplicación de pago, posibilidad bastante importante.

Viabilidad legal

En cuanto a la viabilidad legal, hay que tener en cuenta que la base de datos trabaja con datos catalogados como muy sensibles, como es el caso de números telefónicos o contraseñas. Todas las organizaciones que tratan con grandes volúmenes de datos están sujetas a la ley en materia de protección de datos de cada país, siendo el caso de España la conocida Ley Orgánica de Protección de Datos.

La LOPD es una ley que tiene como objetivo adaptarse al *Reglamento Oficial de Protección de Datos* de la Unión Europea, en la cual se pretende garantizar que la libre circulación de los datos esté protegida. Dicha ley exige que se acepte que la empresa, en este caso la *Fundación Miradas*, esté obligada a solicitar el permiso a los usuarios para que los datos de usuarios y organizaciones para poder ser tratados con fines estadísticos, dentro de lo marcado por la LOPD o las leyes equivalentes en otros países en esta materia:



Miguel

Gomez Gentil

mgg0174@ubu.es

.....

687878787

Usuario de la Fundación Miradas



Autismo Gamonal



Acepto que la Fundación Miradas guarde registro
sobre mis datos de acuerdo con la Ley Orgánica
de Protección de Datos

REGISTRARSE

En esta captura, cuyos datos no son verdaderos, al tener activada la casilla de aceptación, da la posibilidad de registrarse al usuario sin ningún problema.

Apéndice B

Especificación de Requisitos

B.1. Introducción

En este apéndice se desarrollan los aspectos relacionados con la obtención de los requisitos, tanto funcionales como no funcionales, y su posterior especificación mediante los casos de uso. Cada requisito representa una acción o actividad esperada por parte de la aplicación para un correcto funcionamiento de la misma, siendo relacionado a posteriori con un caso de uso. Al tener todos los casos de uso, se relacionan posteriormente con los actores correspondientes.

También se desarrolla en este apéndice, los diferentes tipos de actores o usuarios que están involucrados en la utilización de la aplicación, los cuales son relacionados a posteriori con los diferentes casos de uso de la aplicación, con el fin de conocer quién es el encargado de realizar cada actividad relacionada con cada caso de uso.

B.2. Objetivos generales

El análisis de requisitos, actores y casos de uso para la aplicación tiene como finalidad realizar un estudio sobre qué actividades se espera que realice la aplicación, quién tiene los permisos necesarios para realizarlas y qué resultados se esperan de su realización.

En cuanto a la obtención de requisitos, se tienen que tener en cuenta tanto los requisitos funcionales, los cuales indican los objetivos que debería marcar la propia aplicación, como los requisitos no funcionales, que se encargan

de marcar las limitaciones que tiene la aplicación. Estos requisitos deben utilizarse posteriormente para la creación de los diferentes casos de uso que abarca la aplicación, con el objetivo de tener que cubrir todo lo esperable dentro del funcionamiento de la misma.

Los requisitos utilizados en este proyecto aparecen en el apartado **Catálogo de requisitos** de este apéndice.

En cuanto a los casos de uso posteriores, se tienen que identificar los actores que están involucrados en el uso de la aplicación, siendo en el caso de esta aplicación usuarios repartidos en cuatro tipos diferentes, teniendo cada uno de ellos diferentes funciones y permisos, con cuatro tipos distintos de usuario con diferentes permisos dentro de la aplicación:

- **Administrador:** Es el usuario que posee más permisos en cuanto a funcionalidades de la aplicación se refiere. Dicho actor será el director de *Fundación Miradas*, quien tendrá la potestad de manejar quiénes pueden acceder a la organización y qué organizaciones pueden acceder a ella, además de poder gestionar aquellas operaciones relacionadas con las evaluaciones de indicadores, llenando los valores de las evidencias y calculando el valor total de cada indicador a partir de las mismas, también calculando el valor total obtenido mediante el valor total de cada indicador, operaciones que ejerce el sistema experto del lado del servidor.
- **Director de organización externa:** Este actor actúa como una especie de .^ayudante" para el actor *Administrador*, puesto que tiene la potestad de añadir los equipos evaluadores y los centros de su organización, aparte de modificarlos y gestionarlos, con tal de no depender de *Fundación Miradas* para poder realizar esos cometidos.
- **Usuario de organización:** Es un usuario cuya organización a la que pertenece posee no permisos de evaluación, por lo tanto, dicho usuario pertenece a una organización que está a disposición de ser evaluada. Además de recibir la evaluación por parte de la organización evaluadora, también puede observar los resultados de las diferentes evaluaciones, ya sea mediante tablas o mediante gráficos.

La especificación de los requisitos funcionales en forma de casos de uso aparece en el apartado **Especificación de requisitos** de este apéndice.

B.3. Catálogo de requisitos

Como se ha mencionado en los apartados anteriores de este apéndice, el catálogo de requisitos resume el comportamiento que se desea obtener por parte de la aplicación, a partir de todas las actividades que se desean implementar en la aplicación.

Hay que considerar también que a medida que se va desarrollando la aplicación, se van introduciendo mejoras y a la par realizando cambios debido a que la aplicación debe adaptarse en la medida de lo posible al objetivo de poder ser utilizada por la *Fundación Miradas*, por las organizaciones a las que evalúa y por los administradores de la base de datos, por lo que durante todo el tiempo de desarrollo se ha tomado conciencia de lo que se pretende implementar, empleando para ello los siguientes requisitos funcionales:

- **RF-01:** El administrador puede añadir test de indicadores.
- **RF-02:** El administrador puede continuar test de indicadores.
- **RF-03:** El administrador puede ver resultados.
- **RF-04:** El administrador puede descargar informes.
- **RF-05:** El administrador puede añadir organizaciones.
- **RF-06:** El administrador puede eliminar organizaciones.
- **RF-07:** El administrador puede eliminar usuarios.
- **RF-08:** El administrador puede editar su propio usuario.
- **RF-09:** El administrador puede gestionar solicitudes de registro.
- **RF-10:** El administrador puede iniciar sesión.
- **RF-11:** El administrador puede cerrar sesión.
- **RF-12:** El director de organización puede añadir equipos evaluadores.
- **RF-13:** El director de organización puede añadir centros o servicios de sus propios equipos evaluadores.
- **RF-14:** El director de organización puede ver resultados.
- **RF-15:** El director de organización puede descargar informes.
- **RF-16:** El director de organización puede editar organizaciones.

- **RF-17:** El director de organización puede editar centros.
- **RF-18:** El director de organización puede editar equipos evaluadores.
- **RF-19:** El director de organización puede eliminar centros.
- **RF-20:** El director de organización puede eliminar equipos evaluadores.
- **RF-21:** El director de organización puede editar su propio usuario.
- **RF-22:** El director de organización puede iniciar sesión.
- **RF-23:** El director de organización puede cerrar sesión.
- **RF-24:** El director de organización puede darse de alta bajo supervisión del administrador.
- **RF-25:** El usuario de organización puede ver resultados.
- **RF-26:** El usuario de organización puede descargar informes.
- **RF-27:** El usuario de organización puede editar su propio usuario.
- **RF-28:** El usuario de organización puede iniciar sesión.
- **RF-29:** El usuario de organización puede cerrar sesión.
- **RF-30:** El usuario de organización puede darse de alta bajo supervisión del administrador.

B.4. Especificación de requisitos

Con los siguientes casos de uso, se han construido los diagramas de caso de uso para cada actor:

CU-1	Realizar test de indicadores
Versión	1.0
Autor	Pablo Ahíta del Barrio
Requisitos asociados	RF-01 y RF-02
Descripción	Realización de la evaluación de indicadores
Precondición	Estar registrado en la aplicación como administrador y tener la sesión activa
Acciones	<ol style="list-style-type: none"> 1. Asignar tipo de evaluación, si aún no se ha empezado 2. Asignar organización 3. Asignar centro o servicio 4. Asignar equipo evaluador 5. Elegir evaluación de indicadores, si ya se ha empezado 6. Ir al indicador anterior 7. Ir al indicador siguiente 8. Rellenar evidencias 9. Rellenar oportunidades de mejora 10. Guardar cambios 11. Rellenar conclusiones 12. Calcular resultados
Postcondición	Ninguna
Excepciones	Ninguna
Importancia	Alta

Tabla B.1: CU-1 Realizar test de indicadores

CU-2	Muestra de resultados
Versión	1.0
Autor	Pablo Ahita del Barrio
Requisitos asociados	RF-03, RF-04, RF-14, RF-15, RF-25 y RF-26
Descripción	Muestra de tabulación de datos, puntuación total, oportunidades de mejora y puntos fuertes
Precondición	Estar registrado en la aplicación y tener la sesión activa
Acciones	<ol style="list-style-type: none"> 1. Ver tabla de indicadores 2. Ver tabla de puntuación 3. Descargar informes
Postcondición	Ninguna
Excepciones	Ninguna
Importancia	Alta

Tabla B.2: CU-2 Muestra de resultados

CU-3	Inicio de sesión
Versión	1.0
Autor	Pablo Ahíta del Barrio
Requisitos asociados	RF-10, RF-11, RF-22, RF-23, RF-28, RF-29
Descripción	Gestión de inicio y cierre de sesión de usuarios
Precondición	Estar registrado en la aplicación y tener la sesión activa
Acciones	<ol style="list-style-type: none"> 1. Iniciar sesión 2. Cerrar sesión
Postcondición	Ninguna
Excepciones	Ninguna
Importancia	Alta

Tabla B.3: CU-3 Inicio de sesión en la app

CU-4	Gestión de organizaciones
Versión	1.0
Autor	Pablo Ahíta del Barrio
Requisitos asociados	RF-05, RF-06
Descripción	Gestión de organizaciones
Precondición	Estar registrado en la aplicación como administrador y tener la sesión activa
Acciones	<ol style="list-style-type: none"> 1. Rellenar cuestionario de registro 2. Añadir fotografía 3. Aceptar LOPD
Postcondición	Ninguna
Excepciones	Ninguna
Importancia	Media

Tabla B.4: CU-4 Gestión de organizaciones

CU-5	Edición de la organización
Versión	1.0
Autor	Pablo Ahíta del Barrio
Requisitos asociados	RF-16
Descripción	Edición de la información de la organización
Precondición	Estar registrado en la aplicación como director de organización, que la organización sea la propia a la que pertenece y tener la sesión activa
Acciones	<ol style="list-style-type: none"> 1. Rellenar cuestionario de registro 2. Añadir fotografía
Postcondición	Ninguna
Excepciones	Ninguna
Importancia	Media

Tabla B.5: CU-5 Gestión de organizaciones

CU-6	Gestión de registros de usuarios
Versión	1.0
Autor	Pablo Ahíta del Barrio
Requisitos asociados	RF-07, RF-09
Descripción	Proceso de registro de nuevos usuarios
Precondición	Estar registrado en la aplicación como administrador y tener la sesión activa
Acciones	<ul style="list-style-type: none"> 1. Aceptar solicitudes 2. Rechazar solicitudes 3. Eliminar usuarios
Postcondición	Ninguna
Excepciones	Ninguna
Importancia	Alta

Tabla B.6: CU-6 Gestión de registros de usuarios

CU-7	Editar su usuario
Versión	1.0
Autor	Pablo Ahíta del Barrio
Requisitos asociados	RF-08, RF-21, RF-27
Descripción	Edición de la información del propio usuario
Precondición	Estar registrado en la aplicación y tener la sesión activa
Acciones	<ul style="list-style-type: none"> 1. Cambiar datos 2. Cambiar foto de perfil
Postcondición	Ninguna
Excepciones	Ninguna
Importancia	Media

Tabla B.7: CU-7 Gestión de registros de usuarios

CU-8	Registro del usuario
Versión	1.0
Autor	Pablo Ahíta del Barrio
Requisitos asociados	RF-24, RF-30
Descripción	El usuario se registra
Precondición	El administrador debe haber aceptado la solicitud de registro
Acciones	<ol style="list-style-type: none"> 1. Rellenar datos 2. Agregar foto de perfil 3. Aceptar LOPD
Postcondición	Ninguna
Excepciones	Ninguna
Importancia	Alta

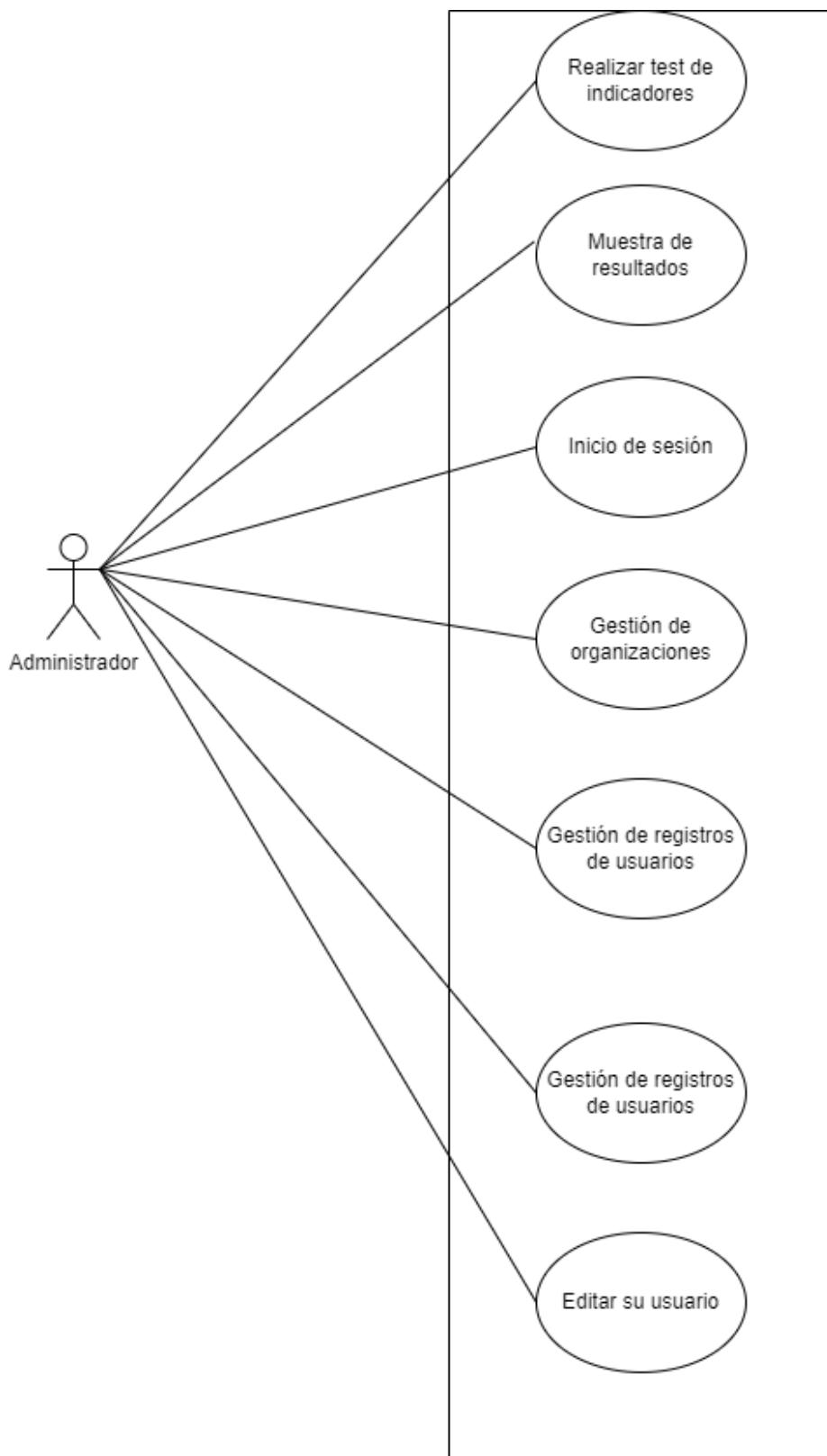
Tabla B.8: CU-8 Gestión de registros de usuarios

CU-9	Gestión de equipos evaluadores
Versión	1.0
Autor	Pablo Ahíta del Barrio
Requisitos asociados	RF-12, RF-18, RF-20
Descripción	Gestión de equipos evaluadores
Precondición	Estar registrado en la aplicación como director de organización evaluada y tener la sesión activa
Acciones	<ol style="list-style-type: none"> 1. Añadir equipo evaluador 2. Editar equipo evaluador 3. Eliminar equipo evaluador
Postcondición	Ninguna
Excepciones	Ninguna
Importancia	Alta

Tabla B.9: CU-9 Gestión de equipos evaluadores

CU-10	Gestión de centros o servicios
Versión	1.0
Autor	Pablo Ahíta del Barrio
Requisitos asociados	RF-13, RF-17, RF-19
Descripción	Gestión de centros o servicios
Precondición	Estar registrado en la aplicación como director de organización evaluada y tener la sesión activa
Acciones	<ol style="list-style-type: none"> 1. Añadir centro 2. Editar centro 3. Eliminar centro
Postcondición	Ninguna
Excepciones	Ninguna
Importancia	Alta

Tabla B.10: CU-10 Gestión de centros o servicios

Figura B.1: Diagrama de casos de uso para el actor *Administrador*

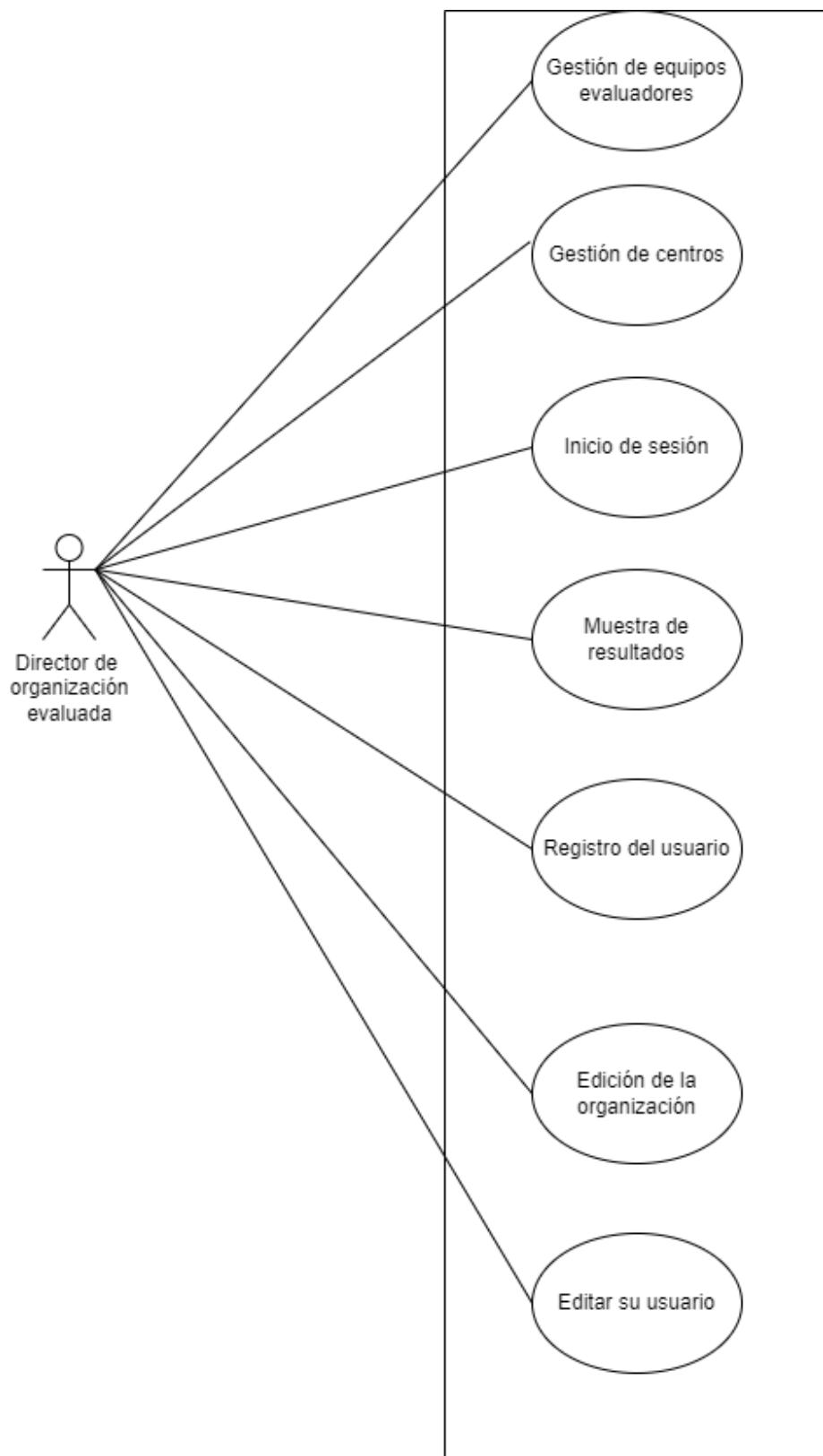


Figura B.2: Diagrama de casos de uso para el actor *Director de organización evaluada*

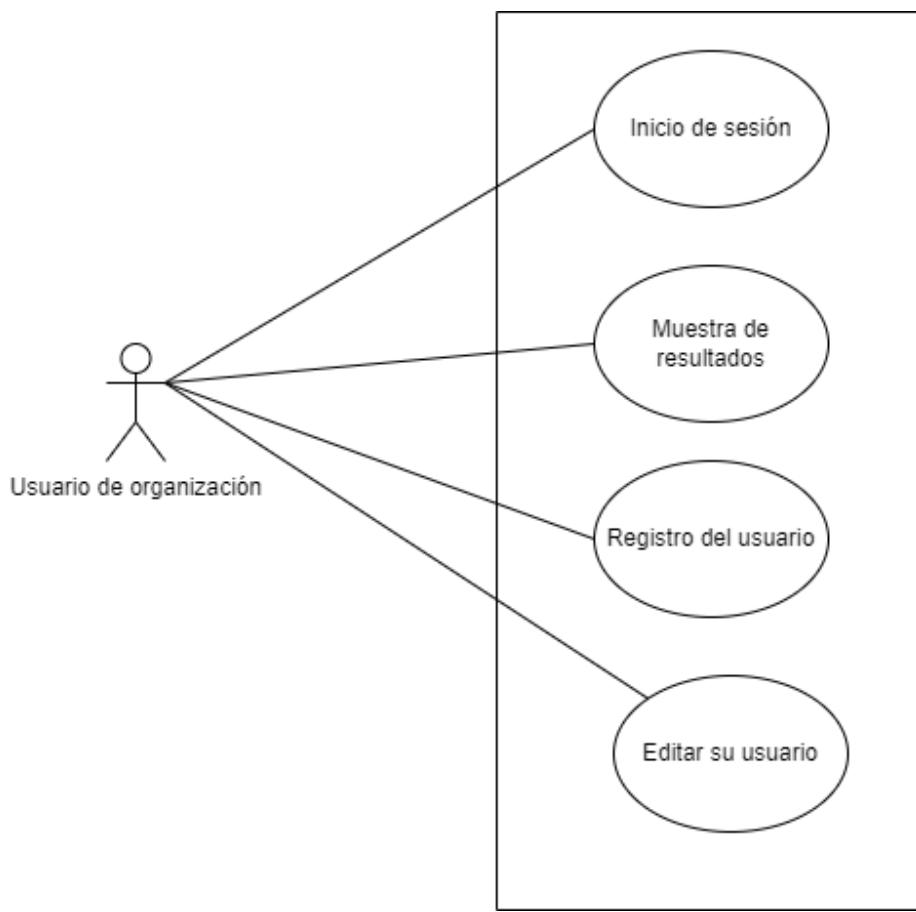


Figura B.3: Diagrama de casos de uso para el actor *Usuario de organización*

Apéndice C

Especificación de diseño

C.1. Introducción

C.2. Diseño de datos

Diseño de las entidades de la base de datos

En cuanto al diseño de los datos, se han utilizado las siguientes clases o entidades, que son idénticas tanto en la base de datos, como en el servidor, como en el cliente:

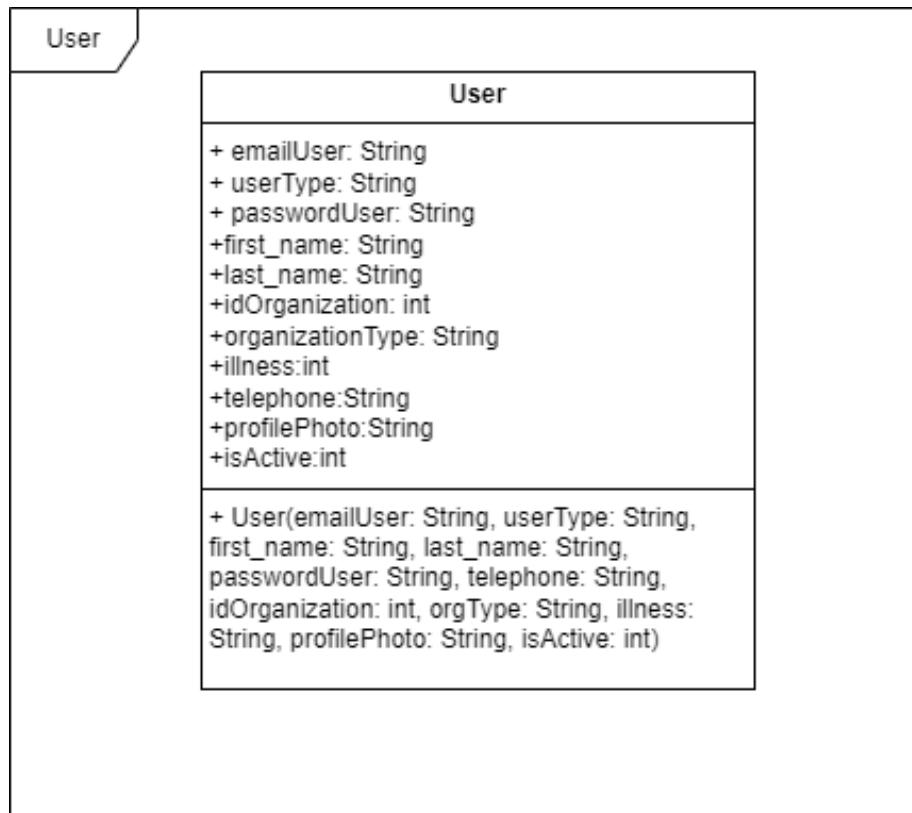


Figura C.1: Paquete user

- **User:** Esta entidad se encarga de almacenar información sobre los usuarios registrados en la base de datos. Dicha entidad consta de los siguientes campos:
 - **emailUser:** Almacena la dirección de correo electrónico del usuario. Es de tipo **String** o **VARCHAR(500)**
 - **userType:** Indica el tipo de usuario, que puede ser **ADMIN**, **DIRECTOR** o **ORGANIZATION**. Es de tipo **String** o **VARCHAR(50)**
 - **passwordUser:** Almacena la contraseña del usuario, hasheada en **SHA-256**. Es de tipo **String** o **VARCHAR(500)**.
 - **first_name:** Almacena el nombre del usuario. Es de tipo **String** o **VARCHAR(500)**.
 - **last_name:** Almacena el apellido del usuario. Es de tipo **String** o **VARCHAR(500)**.

- **idOrganization:** Almacena el identificador de la organización a la que pertenece el usuario. Es de tipo INT
- **organizationType:** Indica el tipo de la organización, que puede ser ".EV ALUATED." o ".EV ALUATOR". Es de tipo String o VARCHAR(50).
- **illness:** Almacena el tipo de enfermedad relacionada con la organización. Es de tipo String o VARCHAR(50).
- **telephone:** Almacena el número de teléfono del usuario, incluyendo el prefijo telefónico de su país. Es de tipo String o VARCHAR
- **profilePhoto:** Almacena el nombre del fichero de la fotografía de perfil del usuario, para que pueda acceder Azure Blob Storage. Es de tipo String o VARCHAR(MAX).
- **isActive:** Esta variable interna determina si el usuario está o no activo. En caso de que dicho valor sea igual a -1, se trata de un usuario cuya solicitud no ha sido respondida. En caso de que dicho valor sea igual a 0, significa que el usuario está autorizado a registrarse. Si dicho valor es igual a 1, significa que el usuario está activo. Dicho valor es de tipo entero.
- **PRIMARY KEY:** La clave primaria de esta tabla es el campo ".emailUser", considerando que no se pueden añadir dos emails iguales.
- **CHECK:** Define una serie de condiciones que los datos deben cumplir para ser válidos, comprobando léxicamente la clave primaria.

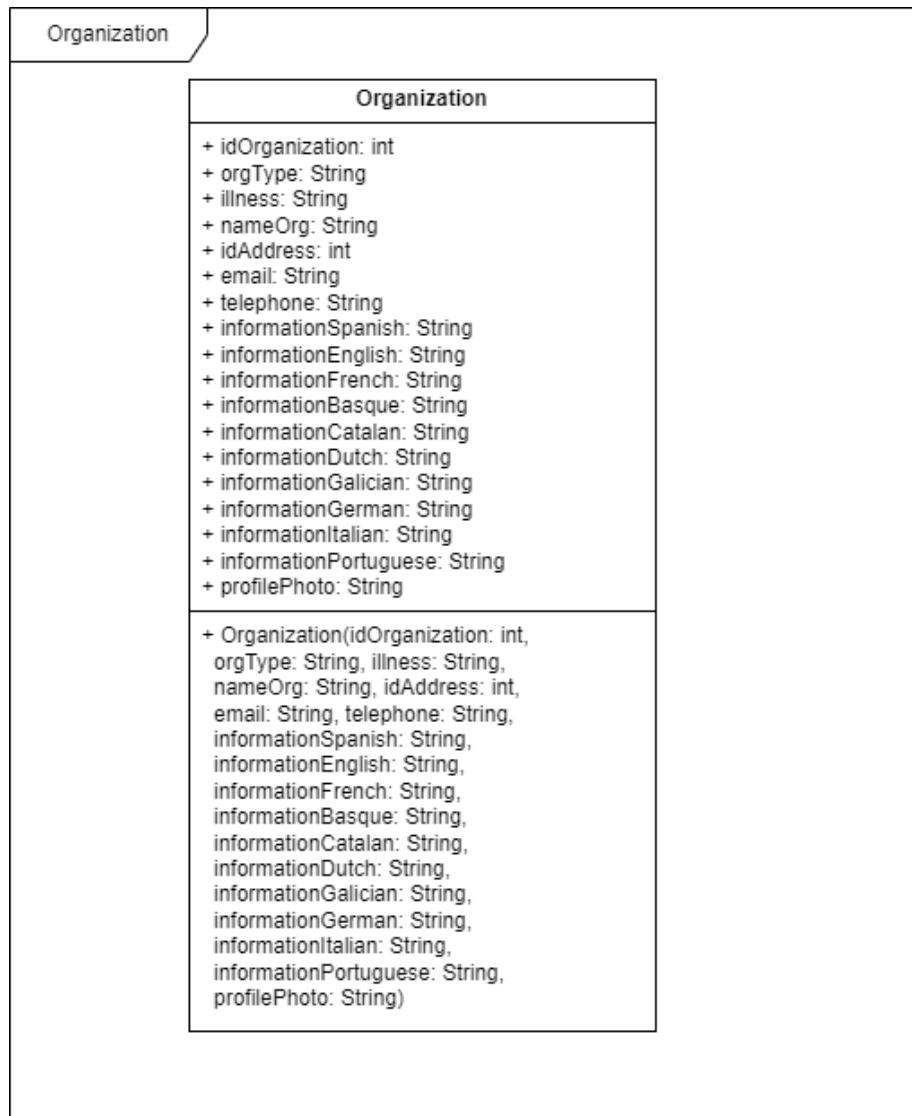
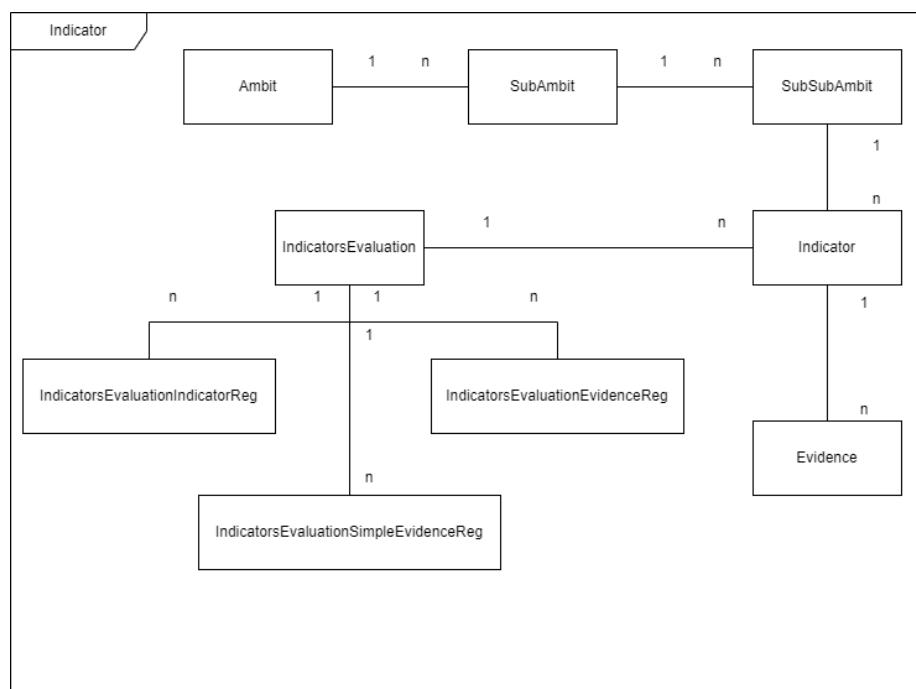
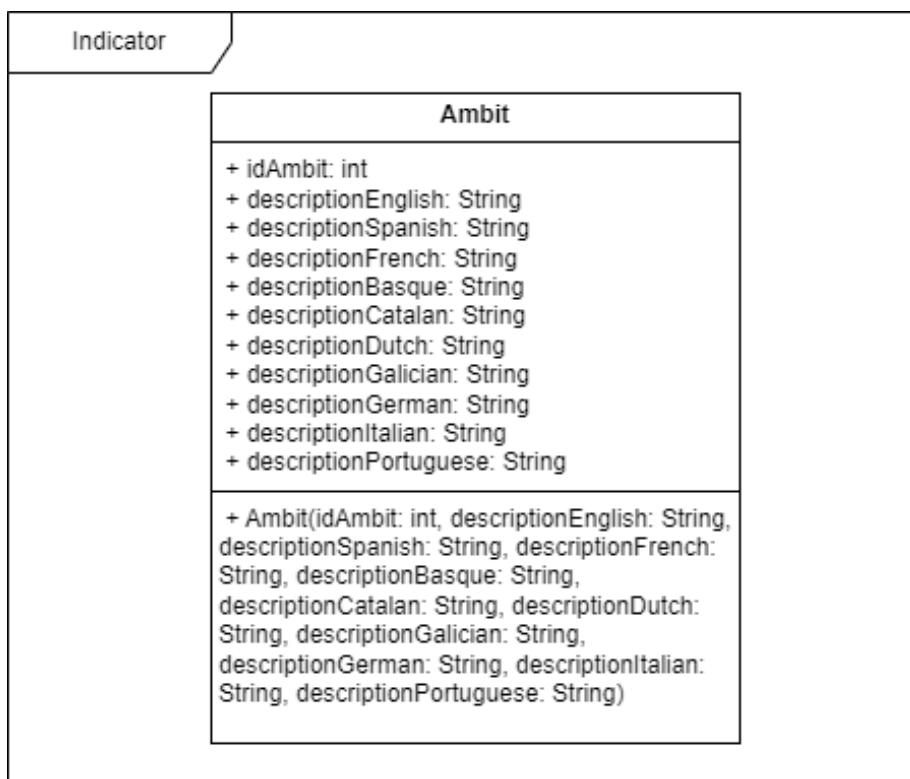


Figura C.2: Paquete organization

- **Organization:** Esta tabla se encarga de almacenar la información sobre las organizaciones registradas de todos los tipos, utilizando para ello los siguientes campos:
 - **idOrganization:** Almacena el identificador de la organización. Es de tipo INT.
 - **orgType:** Indica el tipo de la organización. Es de tipo String o VARCHAR(50).

- **illness:** Almacena el tipo de enfermedad relacionada con la organización. Es de tipo **String** o **VARCHAR(50)**.
- **nameOrg:** Almacena el nombre de la organización. Es de tipo **String** o **VARCHAR(500)**.
- **idAddress:** Almacena el identificador de la dirección de la organización. Es de tipo **INT**.
- **informationSpanish:** Almacena información sobre la organización en español. Es de tipo **String** o **VARCHAR(5000)**.
- **informationEnglish:** Almacena información sobre la organización en inglés. Es de tipo **String** o **VARCHAR(5000)**.
- **informationFrench:** Almacena información sobre la organización en francés. Es de tipo **String** o **VARCHAR(5000)**.
- **informationBasque:** Almacena información sobre la organización en euskera. Es de tipo **String** o **VARCHAR(5000)**.
- **informationCatalan:** Almacena información sobre la organización en catalán. Es de tipo **String** o **VARCHAR(5000)**.
- **informationDutch:** Almacena información sobre la organización en neerlandés. Es de tipo **String** o **VARCHAR(5000)**.
- **informationGalician:** Almacena información sobre la organización en gallego. Es de tipo **String** o **VARCHAR(5000)**.
- **informationGerman:** Almacena información sobre la organización en alemán. Es de tipo **String** o **VARCHAR(5000)**.
- **informationItalian:** Almacena información sobre la organización en italiano. Es de tipo **String** o **VARCHAR(5000)**.
- **informationPortuguese:** Almacena información sobre la organización en portugués. Es de tipo **String** o **VARCHAR(5000)**.
- **email:** Almacena la dirección de correo electrónico de la organización. Es de tipo **String** o **VARCHAR(500)**.
- **telephone:** Almacena el número de teléfono de la organización. Es de tipo **String** o **VARCHAR(50)**.
- **profilePhoto:** Almacena el nombre del fichero de la fotografía de perfil de la organización, para que pueda acceder Azure Blob Storage. Es de tipo **String** o **VARCHAR(MAX)**.
- **PRIMARY KEY:** La clave principal de esta tabla está compuesta por los campos **IdOrganization**, **orgType** e **illness**.
- **FOREIGN KEY:** Establece una relación con la entidad **Address** a través del campo **idAddress**.

Figura C.3: Paquete **indicator**

Figura C.4: Clase **Ambit**

- **Ambit:** Esta tabla se encarga de almacenar información sobre los ámbitos almacenados en la base de datos, utilizando para ello los siguientes campos:
 - **idAmbit:** Almacena el identificador del ámbito. Es de tipo INT.
 - **descriptionSpanish:** Almacena la descripción en español del ámbito. Es de tipo String o VARCHAR(5000).
 - **descriptionEnglish:** Almacena la descripción en inglés del ámbito. Es de tipo String o VARCHAR(5000).
 - **descriptionFrench:** Almacena la descripción en francés del ámbito. Es de tipo String o VARCHAR(5000).
 - **descriptionBasque:** Almacena la descripción en euskera del ámbito. Es de tipo String o VARCHAR(5000).
 - **descriptionCatalan:** Almacena la descripción en catalán del ámbito. Es de tipo String o VARCHAR(5000).
 - **descriptionDutch:** Almacena la descripción en neerlandés del ámbito. Es de tipo String o VARCHAR(5000).
 - **descriptionGalician:** Almacena la descripción en gallego del ámbito. Es de tipo String o VARCHAR(5000).
 - **descriptionGerman:** Almacena la descripción en alemán del ámbito. Es de tipo String o VARCHAR(5000).
 - **descriptionItalian:** Almacena la descripción en italiano del ámbito. Es de tipo String o VARCHAR(5000).
 - **descriptionPortuguese:** Almacena la descripción en portugués del ámbito. Es de tipo String o VARCHAR(5000).
 - **indicatorType:** Indica el tipo de indicador. Es de tipo String o VARCHAR(50).
 - **PRIMARY KEY:** La clave principal de esta tabla está compuesta por los campos **idAmbit** y **indicatorType**.

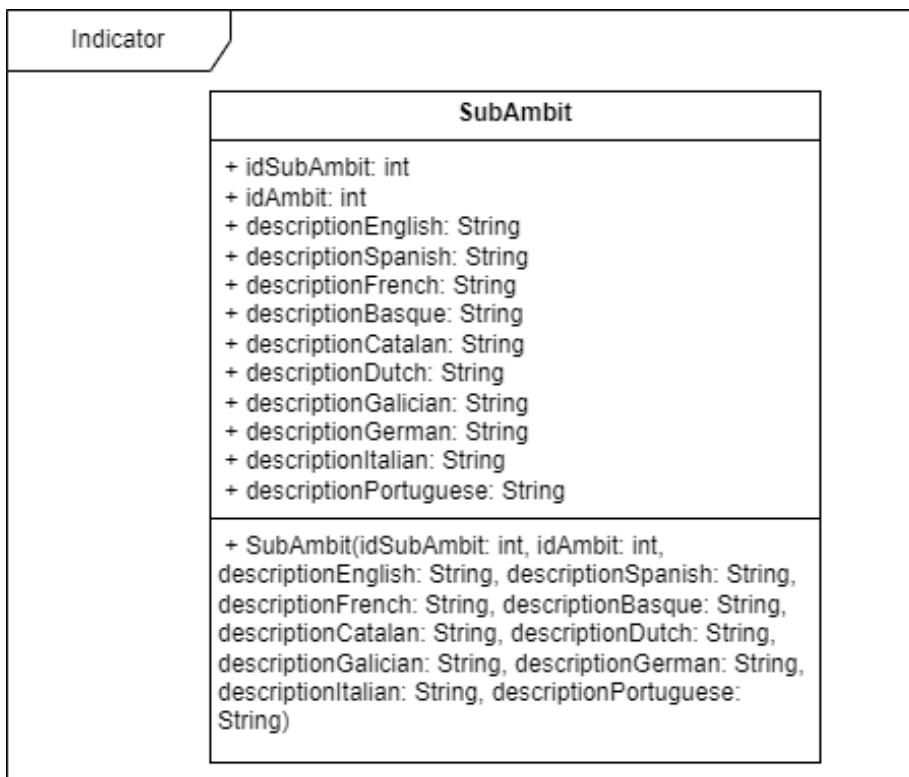


Figura C.5: Clase SubAmbit

- **SubAmbit:** Esta tabla se encarga de almacenar información sobre los subámbitos (primera división de ámbitos) almacenados en la base de datos, utilizando para ello los siguientes campos:
 - **idSubAmbit:** Almacena el identificador del subámbito. Es de tipo INT.
 - **idAmbit:** Almacena el identificador del ámbito. Es de tipo INT.
 - **descriptionSpanish:** Almacena la descripción en español del subámbito. Es de tipo String o VARCHAR(5000).
 - **descriptionEnglish:** Almacena la descripción en inglés del subámbito. Es de tipo String o VARCHAR(5000).
 - **descriptionFrench:** Almacena la descripción en francés del subámbito. Es de tipo String o VARCHAR(5000).
 - **descriptionBasque:** Almacena la descripción en euskera del subámbito. Es de tipo String o VARCHAR(5000).

- **descriptionCatalan:** Almacena la descripción en catalán del subámbito.Es de tipo **String** o **VARCHAR(5000)**.
- **descriptionDutch:** Almacena la descripción en neerlandés del subámbito.Es de tipo **String** o **VARCHAR(5000)**.
- **descriptionGalician:** Almacena la descripción en gallego del subámbito.Es de tipo **String** o **VARCHAR(5000)**.
- **descriptionGerman:** Almacena la descripción en alemán del subámbito.Es de tipo **String** o **VARCHAR(5000)**.
- **descriptionItalian:** Almacena la descripción en italiano del subámbito.Es de tipo **String** o **VARCHAR(5000)**.
- **descriptionPortuguese:** Almacena la descripción en portugués del subámbito.Es de tipo **String** o **VARCHAR(5000)**.
- **indicatorType:** Indica el tipo de indicador.Es de tipo **String** o **VARCHAR(50)**.
- **PRIMARY KEY:** La clave principal de esta tabla está compuesta por los campos **idSubAmbit**, **idAmbit**, **indicatorType**.
- **FOREIGN KEY:** Establece una relación con la entidad **Ambit** a través de los campos **indicatorType**, **idSubAmbit** e **idAmbit**.

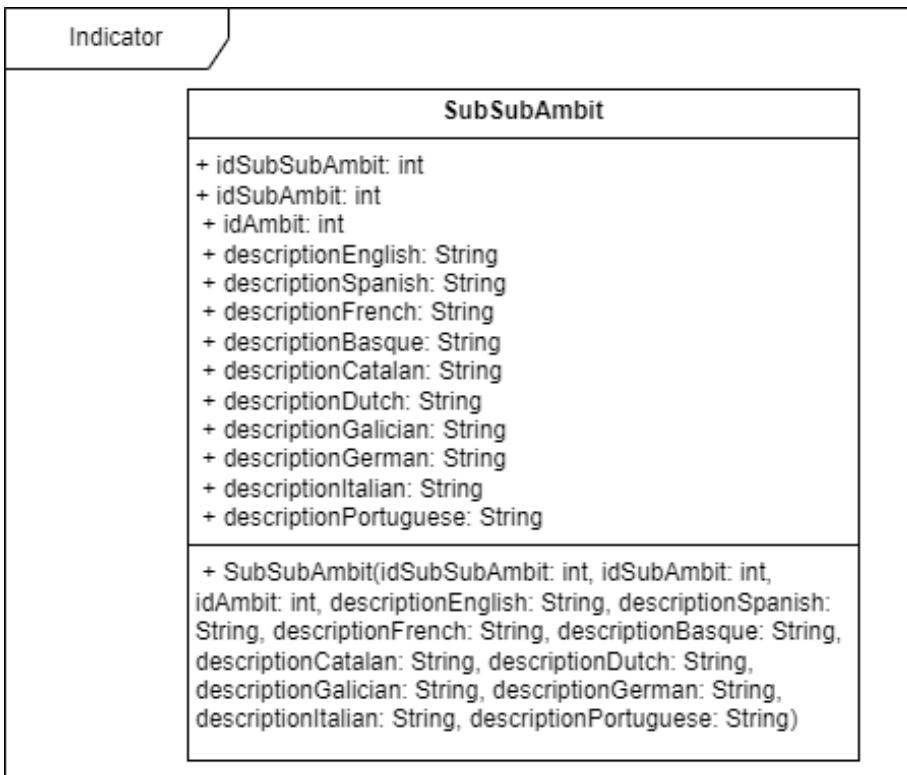


Figura C.6: Clase SubSubAmbit

- **SubSubAmbit:** Esta tabla se encarga de almacenar información sobre los subsubámbitos (segunda división de ámbitos) almacenados en la base de datos, utilizando para ello los siguientes campos:
 - **idSubSubAmbit:** Almacena el identificador del subsubámbito. Es de tipo INT.
 - **idSubAmbit:** Almacena el identificador del subámbito. Es de tipo INT.
 - **idAmbit:** Almacena el identificador del ámbito. Es de tipo INT.
 - **descriptionSpanish:** Almacena la descripción en español del subsubámbito. Es de tipo String o VARCHAR(5000).
 - **descriptionEnglish:** Almacena la descripción en inglés del subsubámbito. Es de tipo String o VARCHAR(5000).
 - **descriptionFrench:** Almacena la descripción en francés del subsubámbito. Es de tipo String o VARCHAR(5000).

- **descriptionBasque:** Almacena la descripción en euskera del subsubámbito.Es de tipo String o VARCHAR(5000).
- **descriptionCatalan:** Almacena la descripción en catalán del subsubámbito.Es de tipo String o VARCHAR(5000).
- **descriptionDutch:** Almacena la descripción en neerlandés del subsubámbito.Es de tipo String o VARCHAR(5000).
- **descriptionGalician:** Almacena la descripción en gallego del subsubámbito.Es de tipo String o VARCHAR(5000).
- **descriptionGerman:** Almacena la descripción en alemán del subsubámbito.Es de tipo String o VARCHAR(5000).
- **descriptionItalian:** Almacena la descripción en italiano del subsubámbito.Es de tipo String o VARCHAR(5000).
- **descriptionPortuguese:** Almacena la descripción en portugués del subsubámbito.Es de tipo String o VARCHAR(5000).
- **indicatorType:** Indica el tipo de indicador.Es de tipo String o VARCHAR(50).
- **PRIMARY KEY:** La clave principal de esta tabla está compuesta por los campos idSubSubAmbit, idSubAmbit, idAmbit, indicatorType.
- **FOREIGN KEY:** Establece una relación con la entidad **SubAmbit** a través de los campos indicatorType, idSubAmbit e idAmbit.

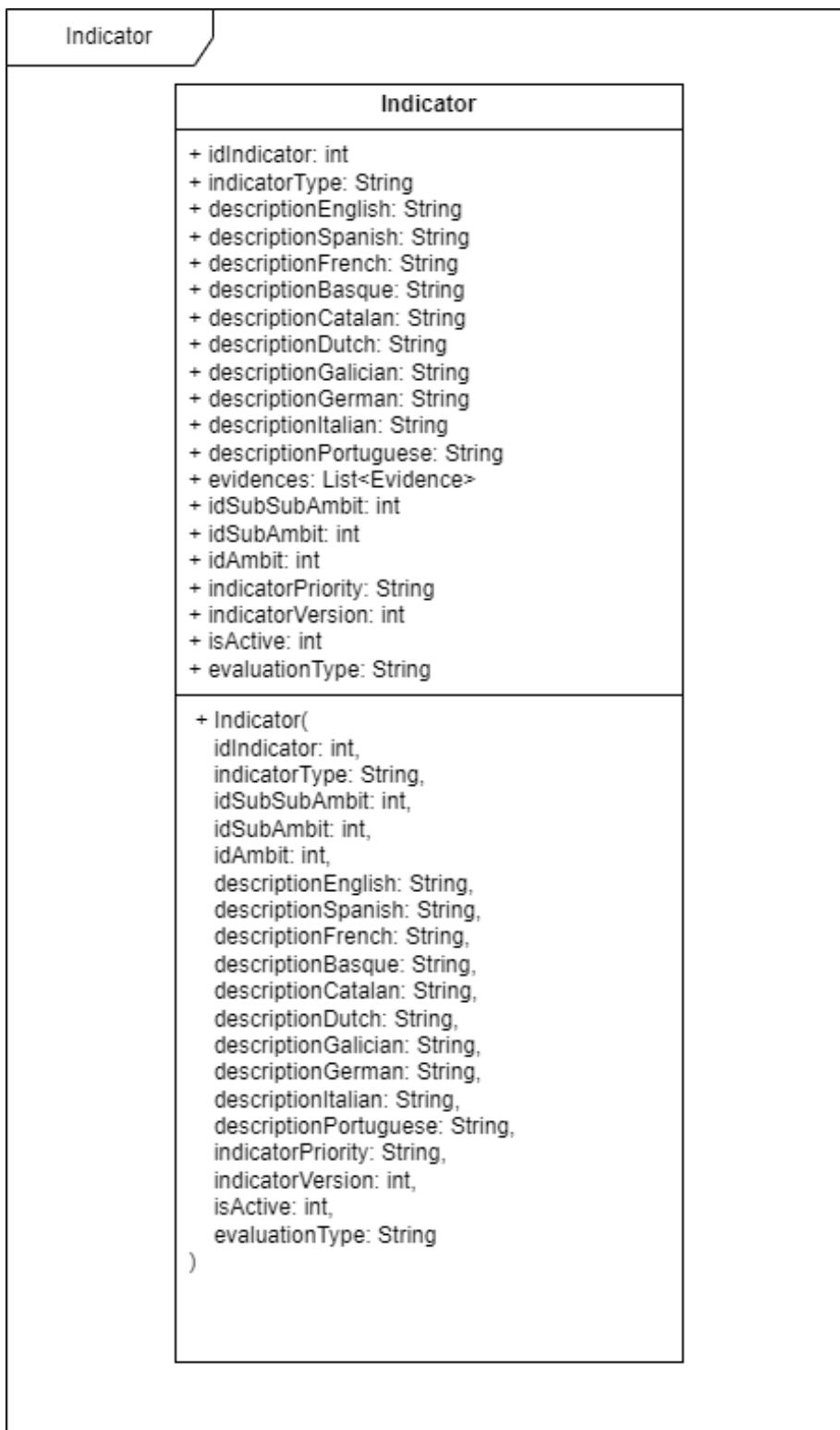


Figura C.7: Clase Indicator

- **Indicator:** Esta tabla se encarga de almacenar información sobre los indicadores almacenados en la base de datos, utilizando para ello los siguientes campos:
 - **idIndicator:** Almacena el identificador del indicador. Es de tipo INT.
 - **indicatorType:** Indica el tipo de indicador. Es de tipo String o VARCHAR(50).
 - **idSubSubAmbit:** Almacena el identificador del subsubámbito. Es de tipo INT.
 - **idSubAmbit:** Almacena el identificador del subámbito. Es de tipo INT.
 - **idAmbit:** Almacena el identificador del ámbito. Es de tipo INT.
 - **descriptionSpanish:** Almacena la descripción en español del indicador. Es de tipo String o VARCHAR(5000).
 - **descriptionEnglish:** Almacena la descripción en inglés del indicador. Es de tipo String o VARCHAR(5000).
 - **descriptionFrench:** Almacena la descripción en francés del indicador. Es de tipo String o VARCHAR(5000).
 - **descriptionBasque:** Almacena la descripción en euskera del indicador. Es de tipo String o VARCHAR(5000).
 - **descriptionCatalan:** Almacena la descripción en catalán del indicador. Es de tipo String o VARCHAR(5000).
 - **descriptionDutch:** Almacena la descripción en neerlandés del indicador. Es de tipo String o VARCHAR(5000).
 - **descriptionGalician:** Almacena la descripción en gallego del indicador. Es de tipo String o VARCHAR(5000).
 - **descriptionGerman:** Almacena la descripción en alemán del indicador. Es de tipo String o VARCHAR(5000).
 - **descriptionItalian:** Almacena la descripción en italiano del indicador. Es de tipo String o VARCHAR(5000).
 - **descriptionPortuguese:** Almacena la descripción en portugués del indicador. Es de tipo String o VARCHAR(5000).
 - **indicatorVersion:** Almacena la versión del indicador. Es de tipo INT.
 - **isActive:** Indica si el indicador está en uso o no. Es de tipo INT.

- **evaluationType**: Almacena si el indicador pertenece a la evaluación completa o a la evaluación simple. Es de tipo **String** o **VARCHAR(50)**.
- **indicatorPriority**: Almacena la prioridad del indicador. Es de tipo **String** o **VARCHAR(50)**.
- **PRIMARY KEY**: La clave principal de esta tabla está compuesta por los campos **idIndicator**, **indicatorType**, **idSubSubAmbit**, **idSubAmbit**, **idAmbit**, **indicatorVersion** y **evaluationType**.
- **FOREIGN KEY**: Establece una relación con la entidad **SubSubAmbit** a través de los campos **indicatorType**, **idSubSubAmbit**, **idSubAmbit**, **idAmbit** e **indicatorType**.

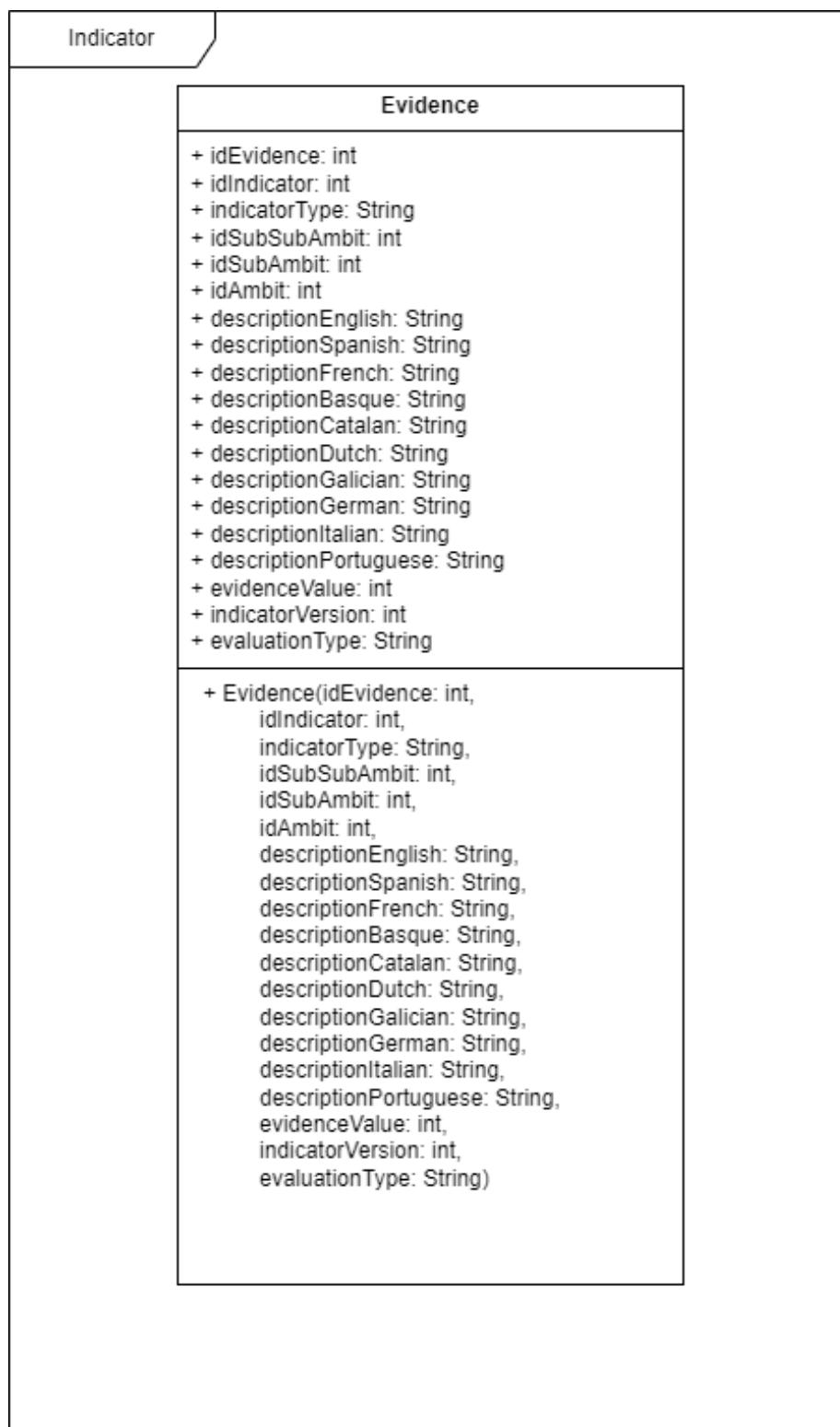


Figura C.8: Clase Evidence

- **Evidence:** Esta entidad se encarga de almacenar información sobre todas las evidencias de cada uno de los indicadores, utilizando para ello los siguientes campos:
 - **idEvidence:** Almacena el identificador de la evidencia. Es de tipo INT.
 - **idIndicator:** Almacena el identificador del indicador al que pertenece la evidencia. Es de tipo INT.
 - **indicatorType:** Indica el tipo de indicador. Es de tipo String o VARCHAR(50).
 - **idSubSubAmbit:** Almacena el identificador del subsubámbito. Es de tipo INT.
 - **idSubAmbit:** Almacena el identificador del subámbito. Es de tipo INT.
 - **idAmbit:** Almacena el identificador del ámbito. Es de tipo INT.
 - **indicatorVersion:** Almacena la versión del indicador. Es de tipo INT.
 - **descriptionSpanish:** Almacena la descripción en español de la evidencia. Es de tipo String o VARCHAR(5000).
 - **descriptionEnglish:** Almacena la descripción en inglés de la evidencia. Es de tipo String o VARCHAR(5000).
 - **descriptionFrench:** Almacena la descripción en francés de la evidencia. Es de tipo String o VARCHAR(5000).
 - **descriptionBasque:** Almacena la descripción en euskera de la evidencia. Es de tipo String o VARCHAR(5000).
 - **descriptionCatalan:** Almacena la descripción en catalán de la evidencia. Es de tipo String o VARCHAR(5000).
 - **descriptionDutch:** Almacena la descripción en neerlandés de la evidencia. Es de tipo String o VARCHAR(5000).
 - **descriptionGalician:** Almacena la descripción en gallego de la evidencia. Es de tipo String o VARCHAR(5000).
 - **descriptionGerman:** Almacena la descripción en alemán de la evidencia. Es de tipo String o VARCHAR(5000).
 - **descriptionItalian:** Almacena la descripción en italiano de la evidencia. Es de tipo String o VARCHAR(5000).
 - **descriptionPortuguese:** Almacena la descripción en portugués de la evidencia. Es de tipo String o VARCHAR(5000).

- **evidenceValue**: Almacena el valor de la evidencia. Es de tipo INT.
 - **evaluationType**: Almacena si la evidencia pertenece a la evaluación completa o a la evaluación simple. Es de tipo String o VARCHAR(50).
 - PRIMARY KEY: La clave principal de esta tabla está compuesta por los campos **idEvidence**, **idIndicator**, **indicatorType**, **idSubSubAmbit**, **idSubAmbit**, **idAmbit**, **indicatorVersion** y **evaluationType**.
 - FOREIGN KEY: Establece una relación con la entidad **Indicator** a través de los campos **idIndicator**, **indicatorType**, **idSubSubAmbit**, **idSubAmbit**, **idAmbit**, **indicatorVersion** y **evaluationType**.



Figura C.9: Clase IndicatorsEvaluation

- **IndicatorsEvaluation:** Esta entidad se encarga de almacenar la información sobre los diferentes test de indicadores, utilizando para ello los siguientes campos:

- **evaluationDate**: Almacena la fecha de evaluación de los indicadores.Es de tipo `long` o `BIGINT`.
- **idEvaluatedOrganization**: Almacena el identificador de la organización evaluada.Es de tipo `INT`.
- **orgTypeEvaluated**: Indica el tipo de la organización evaluada.Es de tipo `String` o de tipo
- **idEvaluatorTeam**: Almacena el identificador del equipo evaluador que realizó la evaluación.Es de tipo `INT`.
- **idEvaluatorOrganization**: Almacena el identificador de la organización a la que pertenece el equipo evaluador.Es de tipo `INT`.
- **orgTypeEvaluator**: Indica el tipo de la organización evaluadora.
- **idCenter**: Almacena el identificador del centro.Es de tipo `INT`.
- **illness**: Almacena el tipo de enfermedad relacionada con la organización evaluada.
- **totalScore**: Almacena la puntuación total del test de indicadores.Es de tipo `INT`.
- **conclusionsSpanish**: Almacena las conclusiones del test de indicadores en español. Es de tipo `String` o `VARCHAR(5000)`.
- **conclusionsEnglish**: Almacena las conclusiones del test de indicadores en inglés. Es de tipo `String` o `VARCHAR(5000)`.
- **conclusionsFrench**: Almacena las conclusiones del test de indicadores en francés. Es de tipo `String` o `VARCHAR(5000)`.
- **conclusionsBasque**: Almacena las conclusiones del test de indicadores en euskera. Es de tipo `String` o `VARCHAR(5000)`.
- **conclusionsCatalan**: Almacena las conclusiones del test de indicadores en catalán. Es de tipo `String` o `VARCHAR(5000)`.
- **conclusionsDutch**: Almacena las conclusiones del test de indicadores en neerlandés. Es de tipo `String` o `VARCHAR(5000)`.
- **conclusionsGalician**: Almacena las conclusiones del test de indicadores en gallego. Es de tipo `String` o `VARCHAR(5000)`.
- **conclusionsGerman**: Almacena las conclusiones del test de indicadores en alemán. Es de tipo `String` o `VARCHAR(5000)`.
- **conclusionsItalian**: Almacena las conclusiones del test de indicadores en italiano. Es de tipo `String` o `VARCHAR(5000)`.

- **conclusionsPortuguese:** Almacena las conclusiones del test de indicadores en portugués. Es de tipo String o VARCHAR(5000).
- **isFinished:** Almacena si el test de indicadores ha finalizado (1) o no ha finalizado (0). Es de tipo INT.
- **evaluationType:** Almacena si el indicador pertenece a la evaluación completa o a la evaluación simple. Es de tipo String o VARCHAR(50).
- **scorePriorityZeroColourRed:** Almacena el puntaje para los indicadores de estado IN_START el interés LOW_INTEREST. Es de tipo INT.
- **scorePriorityZeroColourYellow:** Almacena el puntaje para los indicadores de estado IN_PROCESS el interés LOW_INTEREST. Es de tipo INT.
- **scorePriorityZeroColourGreen:** Almacena el puntaje para los indicadores de estado REACHED el interés LOW_INTEREST. Es de tipo INT.
- **scorePriorityOneColourRed:** Almacena el puntaje para los indicadores de estado IN_START el interés MEDIUM_INTEREST. Es de tipo INT.
- **scorePriorityOneColourYellow:** Almacena el puntaje para los indicadores de estado IN_PROCESS el interés MEDIUM_INTEREST. Es de tipo INT.
- **scorePriorityOneColourGreen:** Almacena el puntaje para los indicadores de estado REACHED el interés MEDIUM_INTEREST. Es de tipo INT.
- **scorePriorityTwoColourRed:** Almacena el puntaje para los indicadores de estado IN_START el interés HIGH_INTEREST. Es de tipo INT.
- **scorePriorityTwoColourYellow:** Almacena el puntaje para los indicadores de estado IN_PROCESS el interés HIGH_INTEREST. Es de tipo INT.
- **scorePriorityTwoColourGreen:** Almacena el puntaje para los indicadores de estado REACHED el interés HIGH_INTEREST. Es de tipo INT.
- **scorePriorityThreeColourRed:** Almacena el puntaje para los indicadores de estado IN_START el interés FUNDAMENTAL_INTEREST. Es de tipo INT.

- **scorePriorityThreeColourYellow:** Almacena el puntaje para los indicadores de estado IN_PROCESS el interés FUNDAMENTAL_INTEREST.Es de tipo INT.
- **scorePriorityThreeColourGreen:** Almacena el puntaje para los indicadores de estado REACHED el interés FUNDAMENTAL_INTEREST.Es de tipo INT.
- **level:** Almacena el nivel del indicador, obtenido del sistema experto. El valor puede ser IN_START, IN_PROCESS o REACHED. Es de tipo String o VARCHAR(50).
- **PRIMARY KEY:** La clave principal de esta tabla está compuesta por los campos evaluationDate, idEvaluatorTeam, idEvaluatorOrganization, orgTypeEvaluator, idEvaluatedOrganization, orgTypeEvaluated, idCenter, illness y evaluationType.
- **FOREIGN KEY:** Establece una relación con la entidad **Evaluator-Team** a través de los campos idEvaluatorTeam, idEvaluatorOrganization, orgTypeEvaluator, idEvaluatedOrganization, orgTypeEvaluated, idCenter e illness.

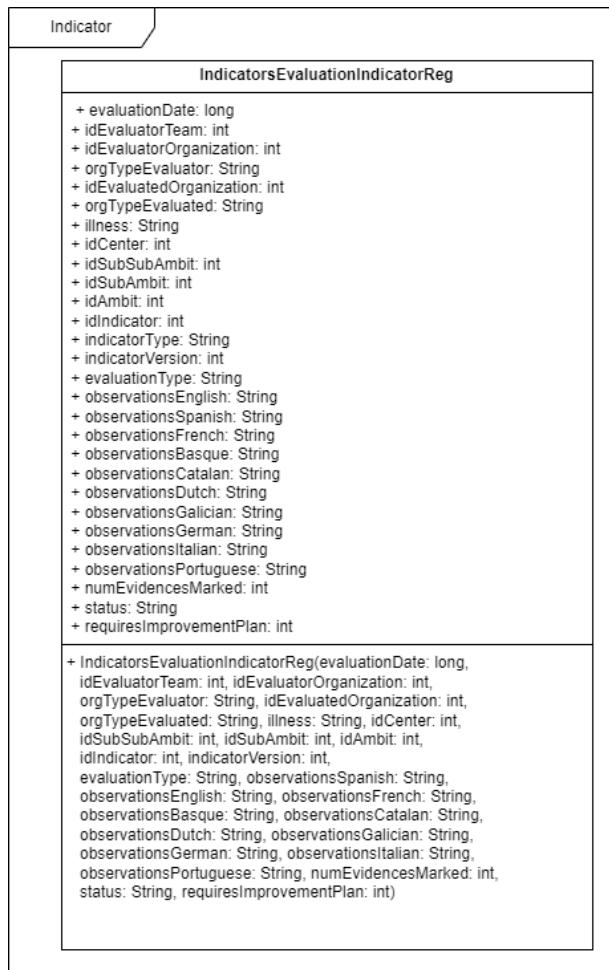
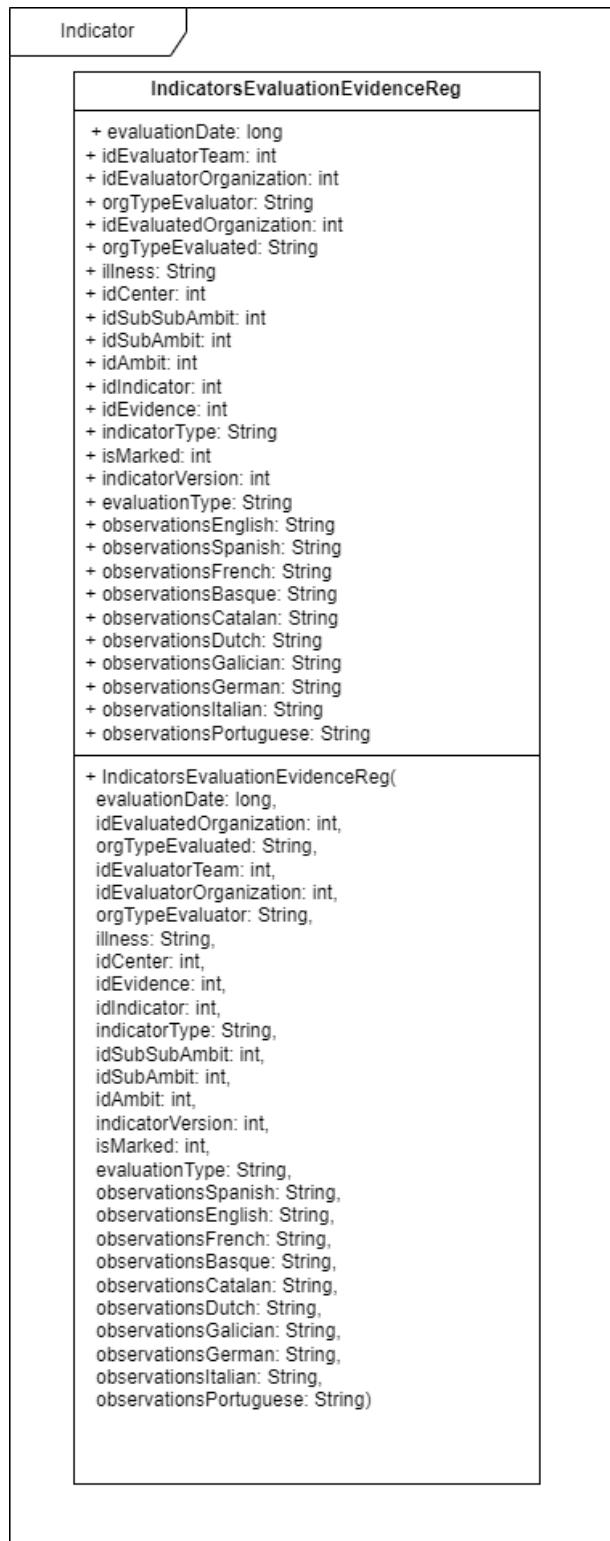


Figura C.10: Clase IndicatorsEvaluationIndicatorReg

- **IndicatorsEvaluationIndicatorReg:** Esta entidad se encarga de almacenar los registros de evidencias (para evaluaciones completas) de cada uno de los test de indicadores realizados, existiendo uno para cada evidencia rellena, utilizando para ello los siguientes campos:
 - **evaluationDate:** Almacena la fecha de la evaluación de los indicadores.
 - **idEvaluatedOrganization:** Almacena el identificador de la organización evaluada. Es de tipo INT.
 - **orgTypeEvaluated:** Indica el tipo de la organización evaluada.
 - **idEvaluatorTeam:** Almacena el identificador del equipo evaluador que realizó la evaluación. Es de tipo INT.

- **idEvaluatorOrganization:** Almacena el identificador de la organización a la que pertenece el equipo evaluador. Es de tipo INT.
- **orgTypeEvaluator:** Indica el tipo de la organización evaluadora.
- **illness:** Almacena el tipo de enfermedad relacionada con la organización evaluada.
- **idCenter:** Almacena el identificador del centro. Es de tipo INT.
- **idIndicator:** Almacena el identificador del indicador evaluado. Es de tipo INT.
- **indicatorType:** Indica el tipo de indicador. Es de tipo String o VARCHAR(50).
- **idSubSubAmbit:** Almacena el identificador del subsubámbito. Es de tipo INT.
- **idSubAmbit:** Almacena el identificador del subámbito. Es de tipo INT.
- **idAmbit:** Almacena el identificador del ámbito. Es de tipo INT.
- **indicatorVersion:** Almacena la versión del indicador evaluado. Es de tipo INT.
- **evaluationType:** Almacena si el indicador pertenece a la evaluación completa o a la evaluación simple. Es de tipo String o VARCHAR(50).
- **observationsSpanish:** Almacena las observaciones del registro en español. Es de tipo String o VARCHAR(MAX).
- **observationsEnglish:** Almacena las observaciones del registro en inglés. Es de tipo String o VARCHAR(MAX).
- **observationsFrench:** Almacena las observaciones del registro en francés. Es de tipo String o VARCHAR(MAX).
- **observationsBasque:** Almacena las observaciones del registro en euskera. Es de tipo String o VARCHAR(MAX).
- **observationsCatalan:** Almacena las observaciones del registro en catalán. Es de tipo String o VARCHAR(MAX).
- **observationsDutch:** Almacena las observaciones del registro en neerlandés. Es de tipo String o VARCHAR(MAX).
- **observationsGalician:** Almacena las observaciones del registro en gallego. Es de tipo String o VARCHAR(MAX).

- **observationsGerman**: Almacena las observaciones del registro en alemán. Es de tipo String o VARCHAR(MAX).
- **observationsItalian**: Almacena las observaciones del registro en italiano. Es de tipo String o VARCHAR(MAX).
- **observationsPortuguese**: Almacena las observaciones del registro en portugués. Es de tipo String o VARCHAR(MAX).
- **numEvidencesMarked**: Almacena el número de evidencias marcadas para ese indicador. Es de tipo INT.
- **status**: Almacena el estado del indicador, que puede ser IN_START, IN_PROCESS o REACHED, valores obtenidos por el sistema experto a partir de numEvidencesMarked. Es de tipo String o VARCHAR(50)
- **requiresImprovementPlan**: Almacena si el indicador requiere estar en el plan de mejora (1) o no (0), se obtiene a partir del sistema experto con el valor de **status**. Es de tipo INT.
- **PRIMARY KEY**: La clave principal de esta tabla está compuesta por los campos evaluationDate, idEvaluatorTeam, idEvaluatorOrganization, orgTypeEvaluator, idEvaluatedOrganization, orgTypeEvaluated, idCenter, illness, idIndicator, indicatorType, idSubSubAmbit, idSubAmbit, idAmbit, indicatorVersion y evaluationType.
- **FOREIGN KEY**: Establece una relación con la entidad **Indicator** a través de los campos idIndicator, indicatorType, idSubSubAmbit, idSubAmbit, idAmbit, indicatorVersion y evaluationType.
- **FOREIGN KEY**: Establece una relación con la entidad **IndicatorsEvaluation** a través de los campos evaluationDate, idEvaluatedOrganization, orgTypeEvaluated, illness, indicatorVersion y evaluationType.

Figura C.11: Clase `IndicatorsEvaluationEvidenceReg`

- **IndicatorsEvaluationEvidenceReg:** Esta entidad se encarga de almacenar los registros de evidencias (para evaluaciones completas) de cada uno de los test de indicadores realizados, existiendo uno para cada evidencia rellenada, utilizando para ello los siguientes campos:
 - **evaluationDate:** Almacena la fecha de la evaluación de los indicadores.
 - **idEvaluatedOrganization:** Almacena el identificador de la organización evaluada.Es de tipo INT.
 - **orgTypeEvaluated:** Indica el tipo de la organización evaluada.
 - **idEvaluatorTeam:** Almacena el identificador del equipo evaluador que realizó la evaluación.Es de tipo INT.
 - **idEvaluatorOrganization:** Almacena el identificador de la organización a la que pertenece el equipo evaluador.Es de tipo INT.
 - **orgTypeEvaluator:** Indica el tipo de la organización evaluadora.
 - **illness:** Almacena el tipo de enfermedad relacionada con la organización evaluada.
 - **idCenter:** Almacena el identificador del centro.Es de tipo INT.
 - **idEvidence:** Almacena el identificador de la evidencia relacionada con el indicador.Es de tipo INT.
 - **idIndicator:** Almacena el identificador del indicador evaluado. Es de tipo INT.
 - **indicatorType:** Indica el tipo de indicador.Es de tipo String o VARCHAR(50).
 - **idSubSubAmbit:** Almacena el identificador del subsubámbito. Es de tipo INT.
 - **idSubAmbit:** Almacena el identificador del subámbito. Es de tipo INT.
 - **idAmbit:** Almacena el identificador del ámbito. Es de tipo INT.
 - **indicatorVersion:** Almacena la versión del indicador evaluado. Es de tipo INT.
 - **isMarked:** Es un campo entero no nulo que indica si el indicador está marcado o no. Solo puede tener valores 0 o 1, lo que se verifica con una restricción CHECK.Es de tipo INT.

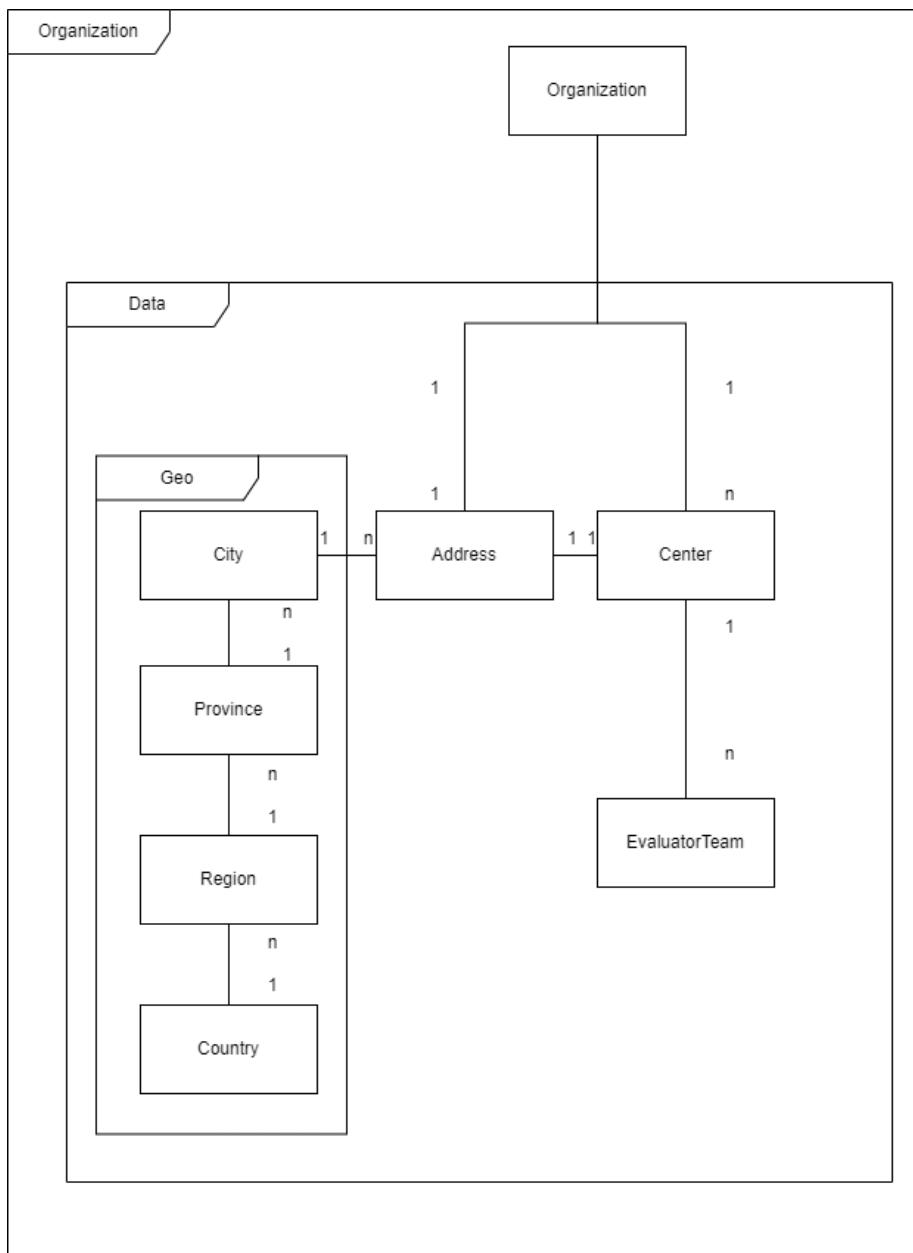
- **evaluationType:** Almacena si el indicador pertenece a la evaluación completa o a la evaluación simple. Es de tipo **String** o **VARCHAR(50)**.
- **observationsSpanish:** Almacena las observaciones del registro en español. Es de tipo **String** o **VARCHAR(MAX)**.
- **observationsEnglish:** Almacena las observaciones del registro en inglés. Es de tipo **String** o **VARCHAR(MAX)**.
- **observationsFrench:** Almacena las observaciones del registro en francés. Es de tipo **String** o **VARCHAR(MAX)**.
- **observationsBasque:** Almacena las observaciones del registro en euskera. Es de tipo **String** o **VARCHAR(MAX)**.
- **observationsCatalan:** Almacena las observaciones del registro en catalán. Es de tipo **String** o **VARCHAR(MAX)**.
- **observationsDutch:** Almacena las observaciones del registro en neerlandés. Es de tipo **String** o **VARCHAR(MAX)**.
- **observationsGalician:** Almacena las observaciones del registro en gallego. Es de tipo **String** o **VARCHAR(MAX)**.
- **observationsGerman:** Almacena las observaciones del registro en alemán. Es de tipo **String** o **VARCHAR(MAX)**.
- **observationsItalian:** Almacena las observaciones del registro en italiano. Es de tipo **String** o **VARCHAR(MAX)**.
- **observationsPortuguese:** Almacena las observaciones del registro en portugués. Es de tipo **String** o **VARCHAR(MAX)**.
- **PRIMARY KEY:** La clave principal de esta tabla está compuesta por los campos **evaluationDate**, **idEvaluatorTeam**, **idEvaluatorOrganization**, **orgTypeEvaluator**, **idEvaluatedOrganization**, **orgTypeEvaluated**, **idCenter**, **illness**, **idEvidence**, **idIndicator**, **indicatorType**, **idSubSubAmbit**, **idSubAmbit**, **idAmbit**, **indicatorVersion** y **evaluationType**.
- **FOREIGN KEY:** Establece una relación con la entidad **Evidence** a través de los campos **idEvidence**, **idIndicator**, **indicatorType**, **idSubSubAmbit**, **idSubAmbit**, **idAmbit**, **indicatorVersion** y **evaluationType**.
- **FOREIGN KEY:** Establece una relación con la entidad **IndicatorsEvaluation** a través de los campos **evaluationDate**, **idEvaluatedOrganization**, **orgTypeEvaluated**, **illness**, **indicatorVersion** y **evaluationType**.

Figura C.12: Clase `IndicatorsEvaluationSimpleEvidenceReg`

- **IndicatorsEvaluationSimpleEvidenceReg:** Esta entidad se encarga de almacenar los registros de evidencias (para evaluaciones simples) de cada uno de los test de indicadores realizados, existiendo uno para cada evidencia rellenada, utilizando para ello los siguientes campos:
 - **evaluationDate:** Almacena la fecha de la evaluación de los indicadores.
 - **idEvaluatedOrganization:** Almacena el identificador de la organización evaluada. Es de tipo INT.
 - **orgTypeEvaluated:** Indica el tipo de la organización evaluada.
 - **idEvaluatorTeam:** Almacena el identificador del equipo evaluador que realizó la evaluación. Es de tipo INT.
 - **idEvaluatorOrganization:** Almacena el identificador de la organización a la que pertenece el equipo evaluador. Es de tipo INT.
 - **orgTypeEvaluator:** Indica el tipo de la organización evaluadora.
 - **illness:** Almacena el tipo de enfermedad relacionada con la organización evaluada.
 - **idCenter:** Almacena el identificador del centro. Es de tipo INT.
 - **idIndicator:** Almacena el identificador del indicador evaluado. Es de tipo INT.
 - **idEvidence:** Almacena el identificador de la evidencia relacionada con el indicador. Es de tipo INT.
 - **descriptionSpanish:** Almacena las descripción de la evidencia en español. Es de tipo String o VARCHAR(MAX).
 - **descriptionEnglish:** Almacena las descripción de la evidencia en inglés. Es de tipo String o VARCHAR(MAX).
 - **descriptionFrench:** Almacena las descripción de la evidencia en francés. Es de tipo String o VARCHAR(MAX).
 - **descriptionBasque:** Almacena las descripción de la evidencia en euskera. Es de tipo String o VARCHAR(MAX).
 - **descriptionCatalan:** Almacena las descripción de la evidencia en catalán. Es de tipo String o VARCHAR(MAX).
 - **descriptionDutch:** Almacena las descripción de la evidencia en neerlandés. Es de tipo String o VARCHAR(MAX).
 - **descriptionGalician:** Almacena las descripción de la evidencia en gallego. Es de tipo String o VARCHAR(MAX).

- **descriptionGerman**: Almacena las descripción de la evidencia en alemán. Es de tipo **String** o **VARCHAR(MAX)**.
- **descriptionItalian**: Almacena las descripción de la evidencia en italiano. Es de tipo **String** o **VARCHAR(MAX)**.
- **descriptionPortuguese**: Almacena las descripción de la evidencia en portugués. Es de tipo **String** o **VARCHAR(MAX)**.
- **idSubSubAmbit**: Almacena el identificador del subsubámbito. Es de tipo **INT**.
- **idSubAmbit**: Almacena el identificador del subámbito. Es de tipo **INT**.
- **idAmbit**: Almacena el identificador del ámbito. Es de tipo **INT**.
- **isMarked**: Es un campo entero no nulo que indica si el indicador está marcado o no. Solo puede tener valores 0 o 1, lo que se verifica con una restricción **CHECK**.Es de tipo **INT**.
- **indicatorVersion**: Almacena la versión del indicador evaluado. Es de tipo **INT**.
- **evaluationType**: Almacena si el indicador pertenece a la evaluación completa o a la evaluación simple. Es de tipo **String** o **VARCHAR(50)**.
- **observationsSpanish**: Almacena las observaciones del registro en español. Es de tipo **String** o **VARCHAR(MAX)**.
- **observationsEnglish**: Almacena las observaciones del registro en inglés. Es de tipo **String** o **VARCHAR(MAX)**.
- **observationsFrench**: Almacena las observaciones del registro en francés. Es de tipo **String** o **VARCHAR(MAX)**.
- **observationsBasque**: Almacena las observaciones del registro en euskera. Es de tipo **String** o **VARCHAR(MAX)**.
- **observationsCatalan**: Almacena las observaciones del registro en catalán. Es de tipo **String** o **VARCHAR(MAX)**.
- **observationsDutch**: Almacena las observaciones del registro en neerlandés. Es de tipo **String** o **VARCHAR(MAX)**.
- **observationsGalician**: Almacena las observaciones del registro en gallego. Es de tipo **String** o **VARCHAR(MAX)**.
- **observationsGerman**: Almacena las observaciones del registro en alemán. Es de tipo **String** o **VARCHAR(MAX)**.

- **observationsItalian:** Almacena las observaciones del registro en italiano. Es de tipo **String** o **VARCHAR(MAX)**.
- **observationsPortuguese:** Almacena las observaciones del registro en portugués. Es de tipo **String** o **VARCHAR(MAX)**.
- **PRIMARY KEY:** La clave principal de esta tabla está compuesta por los campos **evaluationDate**, **idEvaluatorTeam**, **idEvaluatorOrganization**, **orgTypeEvaluator**, **idEvaluatedOrganization**, **orgTypeEvaluated**, **idCenter**, **illness**, **idEvidence**, **idIndicator**, **indicatorType**, **idSubSubAmbit**, **idSubAmbit**, **idAmbit**, **indicatorVersion** y **evaluationType**.
- **FOREIGN KEY:** Establece una relación con la entidad **Indicator** a través de los campos **idIndicator**, **illness**, **idSubSubAmbit**, **idSubAmbit**, **idAmbit**, **indicatorVersion** y **evaluationType**.
- **FOREIGN KEY:** Establece una relación con la entidad **IndicatorsEvaluation** a través de los campos **evaluationDate**, **idEvaluatedOrganization**, **orgTypeEvaluated**, **illness**, **indicatorVersion** y **evaluationType**.

Figura C.13: Paquete `orgData`

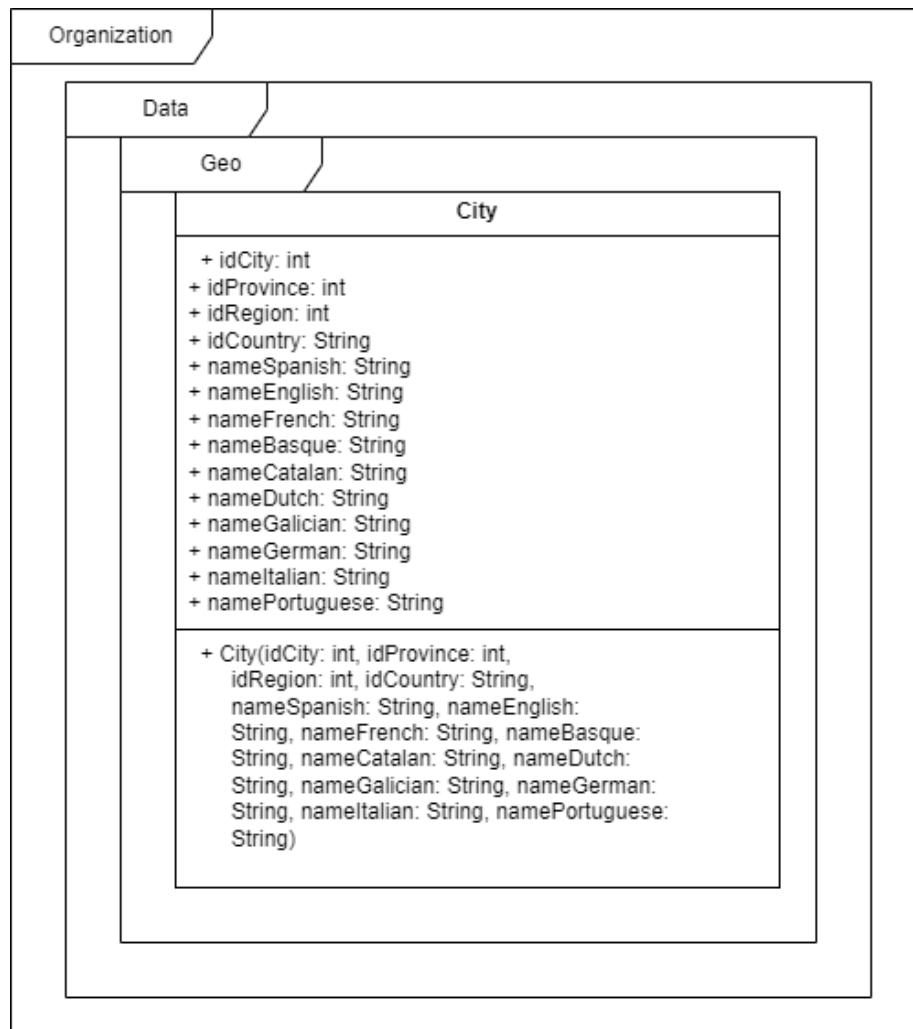


Figura C.14: Clase City

- **City:** Esta entidad se utiliza como carga de las ciudades precargadas, utilizando para ello los siguientes campos:
 - **idCity:** Es el identificador de la ciudad. Es de tipo INT.
 - **idProvince:** Es el identificador de la provincia a la que pertenece la ciudad. Es de tipo INT.
 - **idRegion:** Es el identificador de la región o comunidad autónoma a la que pertenece la ciudad. Es de tipo INT.
 - **idCountry:** Es el identificador del país al que pertenece la ciudad. Es de tipo String o VARCHAR(50)
 - **nameSpanish:** Es el nombre de la ciudad en español. Es de tipo String o VARCHAR(500).
 - **nameEnglish:** Es el nombre de la ciudad en inglés. Es de tipo String o VARCHAR(500).
 - **nameFrench:** Es el nombre de la ciudad en francés. Es de tipo String o VARCHAR(500).
 - **nameBasque:** Es el nombre de la ciudad en euskera. Es de tipo String o VARCHAR(500).
 - **nameCatalan:** Es el nombre de la ciudad en catalán. Es de tipo String o VARCHAR(500).
 - **nameDutch:** Es el nombre de la ciudad en neerlandés. Es de tipo String o VARCHAR(500).
 - **nameGalician:** Es el nombre de la ciudad en gallego. Es de tipo String o VARCHAR(500).
 - **nameGerman:** Es el nombre de la ciudad en alemán. Es de tipo String o VARCHAR(500).
 - **nameItalian:** Es el nombre de la ciudad en italiano. Es de tipo String o VARCHAR(500).
 - **namePortuguese:** Es el nombre de la ciudad en portugués. Es de tipo String o VARCHAR(500).
- **PRIMARY KEY:** La clave primaria está conformada por las columnas **idCity**, **idProvince**, **idRegion** e **idCountry**.
- **FOREIGN KEY:** Los campos **idProvince**, **idRegion** e **idCountry** establecen una relación directa con la entidad **Province**.

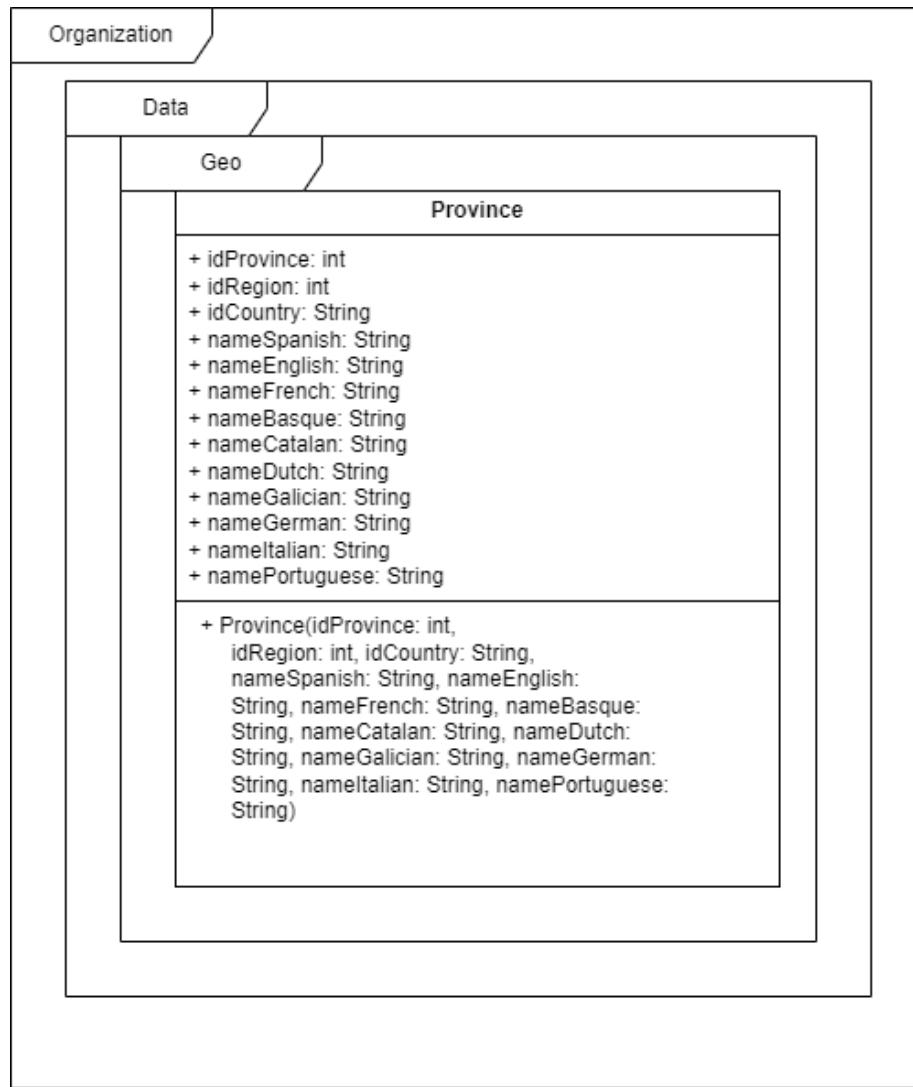
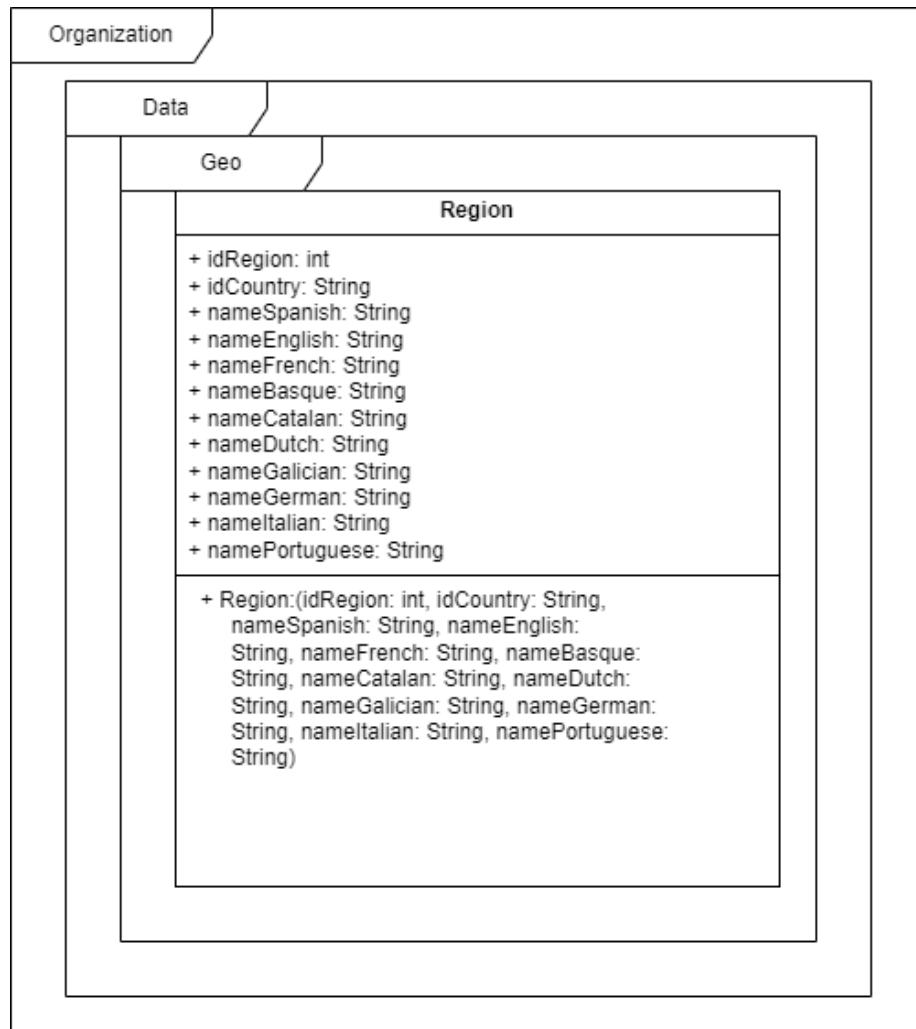


Figura C.15: Clase Province

- **Province:** :Esta entidad se utiliza como carga de las provincias pre-cargadas, utilizando para ello los siguientes campos:
 - **idProvince:** Almacena el identificador de la provincia. Es de tipo INT.
 - **idRegion:** Almacena el identificador de la región a la que pertenece la provincia.Es de tipo INT.
 - **idCountry:** Almacena el identificador del país al que pertenece la provincia. Es de tipo String o VARCHAR(50)

- **nameSpanish:** Es el nombre de la provincia en español.Es de tipo String o VARCHAR(500).
- **nameEnglish:** Es el nombre de la provincia en inglés.Es de tipo String o VARCHAR(500).
- **nameFrench:** Es el nombre de la provincia en francés.Es de tipo String o VARCHAR(500).
- **nameBasque:** Es el nombre de la provincia en euskera.Es de tipo String o VARCHAR(500).
- **nameCatalan:** Es el nombre de la provincia en catalán.Es de tipo String o VARCHAR(500).
- **nameDutch:** Es el nombre de la provincia en neerlandés.Es de tipo String o VARCHAR(500).
- **nameGalician:** Es el nombre de la provincia en gallego.Es de tipo String o VARCHAR(500).
- **nameGerman:** Es el nombre de la provincia en alemán.Es de tipo String o VARCHAR(500).
- **nameItalian:** Es el nombre de la provincia en italiano.Es de tipo String o VARCHAR(500).
- **namePortuguese:** Es el nombre de la provincia en portugués.Es de tipo String o VARCHAR(500).
- **PRIMARY KEY:** La clave principal de esta tabla está compuesta por los campos **idProvince**, **idRegion** e **idCountry**.
- **FOREIGN KEY:** Establece una relación con la tabla regions.^a través de los campos **idRegion** e **idCountry**.

Figura C.16: Clase **Region**

- **Region:** Esta entidad se utiliza como carga de las regiones precargadas, utilizando para ello los siguientes campos:
 - **idRegion:** Almacena el identificador de la región. Es de tipo **INT**.
 - **idCountry:** Almacena el identificador del país al que pertenece la región. Es de tipo **String** o **VARCHAR(50)**.
 - **nameSpanish:** Es el nombre de la región en español. Es de tipo **String** o **VARCHAR(500)**.
 - **nameEnglish:** Es el nombre de la región en inglés. Es de tipo **String** o **VARCHAR(500)**.

- **nameFrench:** Es el nombre de la región en francés.Es de tipo String o VARCHAR(500).
- **nameBasque:** Es el nombre de la región en euskera.Es de tipo String o VARCHAR(500).
- **nameCatalan:** Es el nombre de la región en catalán.Es de tipo String o VARCHAR(500).
- **nameDutch:** Es el nombre de la región en neerlandés.Es de tipo String o VARCHAR(500).
- **nameGalician:** Es el nombre de la región en gallego.Es de tipo String o VARCHAR(500).
- **nameGerman:** Es el nombre de la región en alemán.Es de tipo String o VARCHAR(500).
- **nameItalian:** Es el nombre de la región en italiano.Es de tipo String o VARCHAR(500).
- **namePortuguese:** Es el nombre de la región en portugués.Es de tipo String o VARCHAR(500).
- **PRIMARY KEY:** La clave principal de esta tabla está compuesta por los campos **idRegion** e **idCountry**.
- **FOREIGN KEY:** Establece una relación con la entidad **Country** a través del campo **idCountry**.

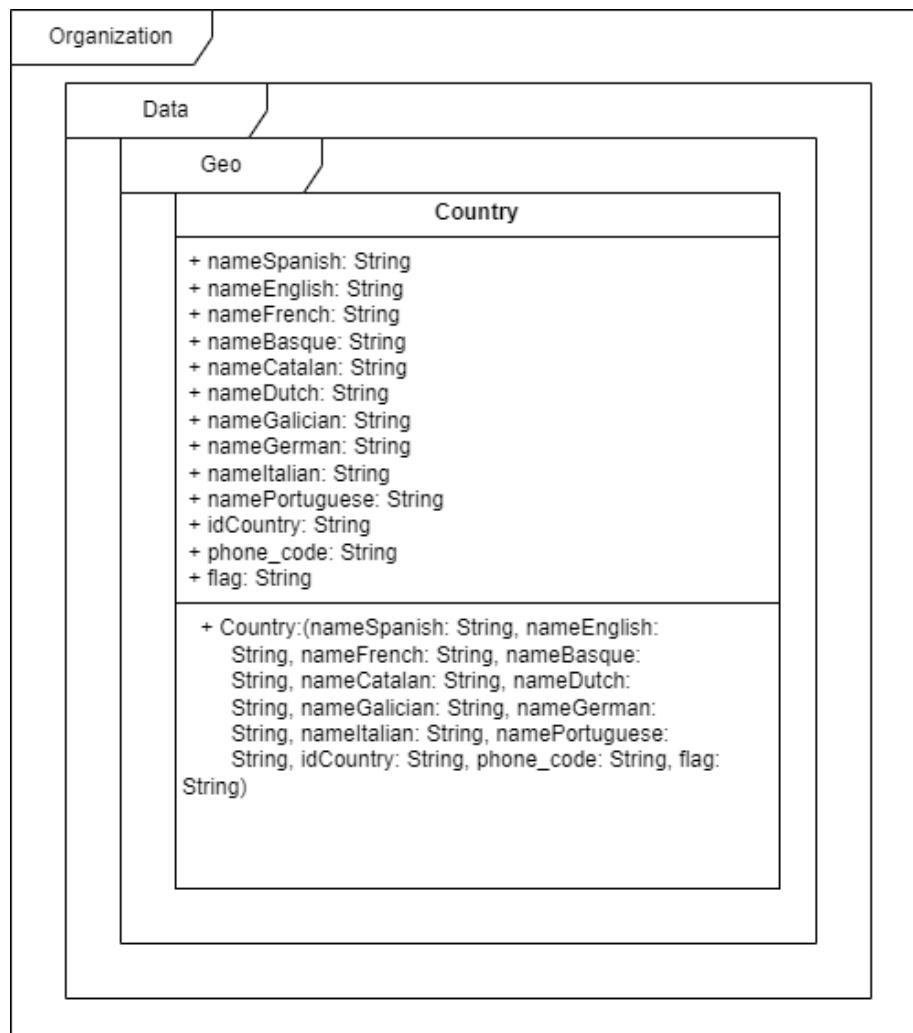
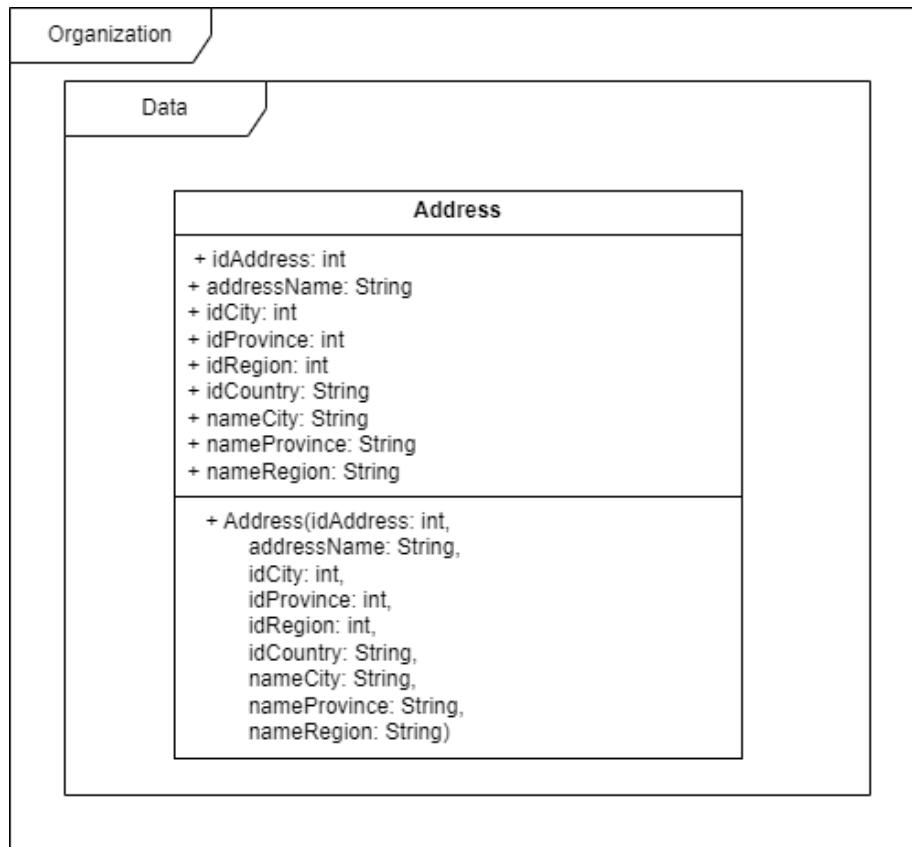


Figura C.17: Clase Country

- **Country:** Esta entidad se utiliza como carga de los diferentes países, utilizando para ello los siguientes campos:
 - **idCountry:** Es el identificador del país. Es de tipo **String** o **VARCHAR(500)**.
 - **nameEnglish:** Es el nombre del país en inglés. Es de tipo **String** o **VARCHAR(500)**.
 - **nameSpanish:** Es el nombre del país en español. Es de tipo **String** o **VARCHAR(500)**.

- **nameFrench:** Es el nombre del país en inglés. Es de tipo **String** o **VARCHAR(500)**.
- **nameBasque:** Es el nombre del país en euskera. Es de tipo **String** o **VARCHAR(500)**.
- **nameCatalan:** Es el nombre del país en catalán. Es de tipo **String** o **VARCHAR(500)**.
- **nameDutch:** Es el nombre del país en neerlandés. Es de tipo **String** o **VARCHAR(500)**.
- **nameGalician:** Es el nombre del país en gallego. Es de tipo **String** o **VARCHAR(500)**.
- **nameGerman:** Es el nombre del país en alemán. Es de tipo **String** o **VARCHAR(500)**.
- **nameItalian:** Es el nombre del país en italiano. Es de tipo **String** o **VARCHAR(500)**.
- **namePortuguese:** Es el nombre del país en portugués. Es de tipo **String** o **VARCHAR(500)**.
- **phone_code:** Es el código telefónico del país, el cual se utiliza para facilitar el registro de números de teléfono. Es de tipo **String** o **VARCHAR(50)**.
- **flag:** Es el código unicode del emoji de la bandera del país, únicamente por motivos visuales. Es de tipo **String** o **NVARCHAR(50)**, ya que es un tipo SQL compatible con Unicode.
- **PRIMARY KEY:** La clave primaria de esta entidad es el campo **idCountry**

Figura C.18: Clase **Address**

■ **Address:** :Esta entidad se utiliza para almacenar información sobre las diferentes direcciones que se almacenan para organizaciones y centros, utilizando para ello los siguientes campos:

- **idAddress:** Almacena el identificador de la dirección. Es de tipo INT.
- **addressName:** Almacena el nombre de la dirección. Es de tipo String o VARCHAR(500).
- **zipCode:** Almacena el código postal de la dirección. Es de tipo INT.
- **idCity:** Almacena el identificador de la ciudad. Es de tipo INT.
- **idProvince:** Almacena el identificador de la provincia. Es de tipo INT.
- **idRegion:** Almacena el identificador de la región. Es de tipo INT.

- **idCountry:** Almacena el identificador del país al que pertenece la dirección.Es de tipo **String** o **VARCHAR(50)**.
- **nameCity:** Almacena el nombre de la ciudad (solo para ciudades no precargadas).Es de tipo **String** o **VARCHAR(500)**.
- **nameProvince:** Almacena el nombre de la provincia (solo para provincias no precargadas).Es de tipo **String** o **VARCHAR(500)**.
- **nameRegion:** Almacena el nombre de la región (solo para regiones no precargadas).Es de tipo **String** o **VARCHAR(500)**.
- **PRIMARY KEY:** La clave primaria de esta entidad es el campo **idAddress**
- **FOREIGN KEY:** Establece una relación con la entidad **Country** a través del campo **idCountry**.
- **FOREIGN KEY:** Establece una relación con la entidad **City** a través de los campos **idCity**, **idProvince**, **idRegion** e **idCountry** (solo si es una ciudad española).

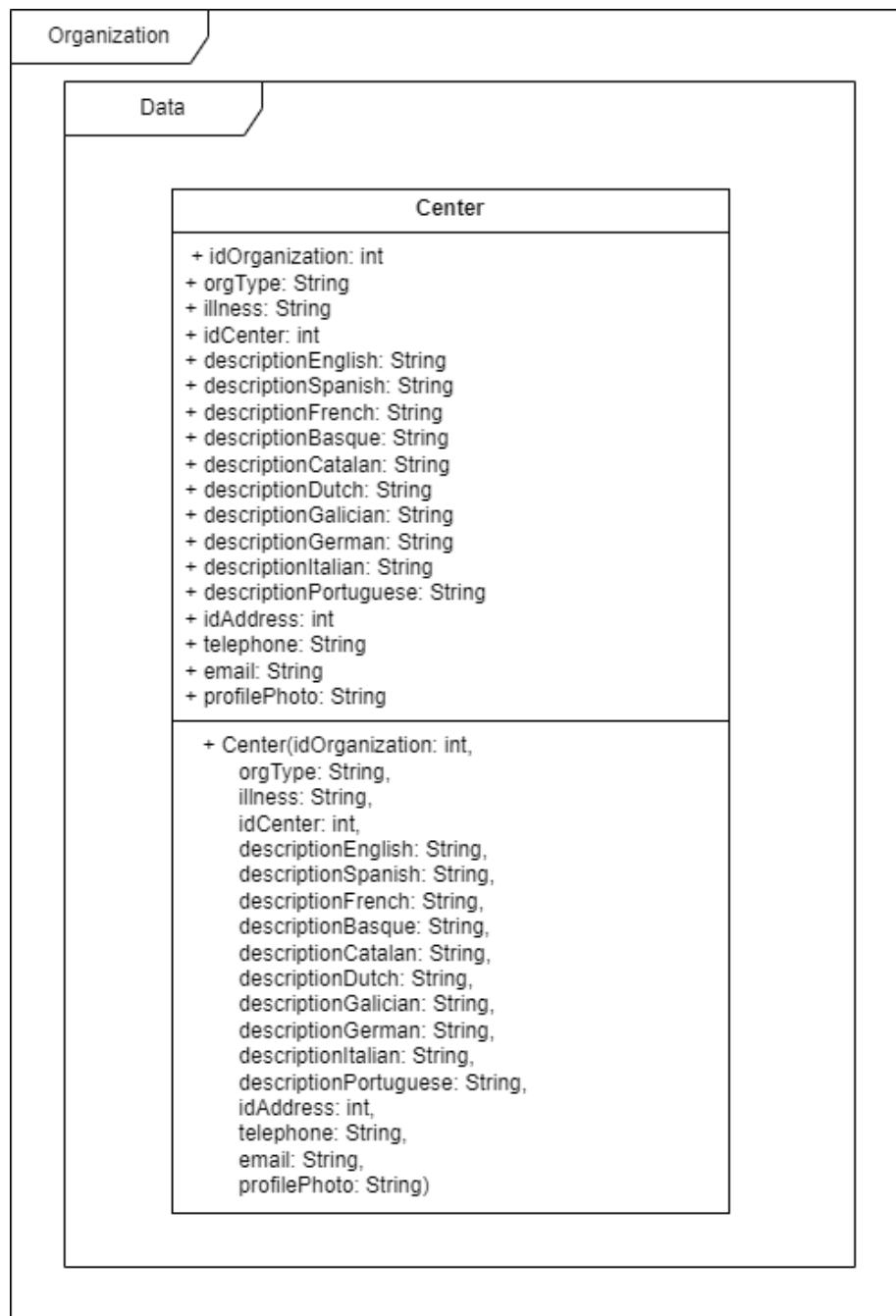


Figura C.19: Clase Center

- **Center:** Esta entidad se encarga de almacenar los centros o servicios que tiene cada organización, utilizando para ello las siguientes columnas:

- **idOrganization:** Almacena el identificador de la organización. Es de tipo INT.
- **orgType:** Indica el tipo de la organización del centro. Es de tipo String o VARCHAR(50).
- **illness:** Almacena el tipo de enfermedad relacionada con la organización del centro. Es de tipo String o VARCHAR(50).
- **idCenter:** Almacena el identificador del centro. Es de tipo INT.
- **descriptionSpanish:** Almacena la descripción en español del centro o servicio. Es de tipo String o VARCHAR(5000).
- **descriptionEnglish:** Almacena la descripción en inglés del centro o servicio. Es de tipo String o VARCHAR(5000).
- **descriptionFrench:** Almacena la descripción en francés del centro o servicio. Es de tipo String o VARCHAR(5000).
- **descriptionBasque:** Almacena la descripción en euskera del centro o servicio. Es de tipo String o VARCHAR(5000).
- **descriptionCatalan:** Almacena la descripción en catalán del centro o servicio. Es de tipo String o VARCHAR(5000).
- **descriptionDutch:** Almacena la descripción en neerlandés del centro o servicio. Es de tipo String o VARCHAR(5000).
- **descriptionGalician:** Almacena la descripción en gallego del centro o servicio. Es de tipo String o VARCHAR(5000).
- **descriptionGerman:** Almacena la descripción en alemán del centro o servicio. Es de tipo String o VARCHAR(5000).
- **descriptionItalian:** Almacena la descripción en italiano del centro o servicio. Es de tipo String o VARCHAR(5000).
- **descriptionPortuguese:** Almacena la descripción en portugués del centro o servicio. Es de tipo String o VARCHAR(5000).
- **idAddress:** Almacena el identificador de la dirección del centro. Es de tipo INT.
- **email:** Almacena el email del centro. Es de tipo String o VARCHAR(50).
- **telephone:** Almacena el número de teléfono del centro. Es de tipo long o BIGINT.
- **profilePhoto:** Almacena el nombre del fichero de la fotografía de perfil del centro o servicio, para que pueda acceder Azure Blob Storage. Es de tipo String o VARCHAR(MAX).

- **PRIMARY KEY:** La clave principal de esta tabla está compuesta por los campos `IdOrganization`, `orgType`, `illness` e `idCenter`.
- **FOREIGN KEY:** Establece una relación con la entidad **Organization** a través de los campos `idOrganization`, `orgType` e `illness`.
- **FOREIGN KEY:** Establece una relación con la tabla **Address** a través del campo `idAddress`.

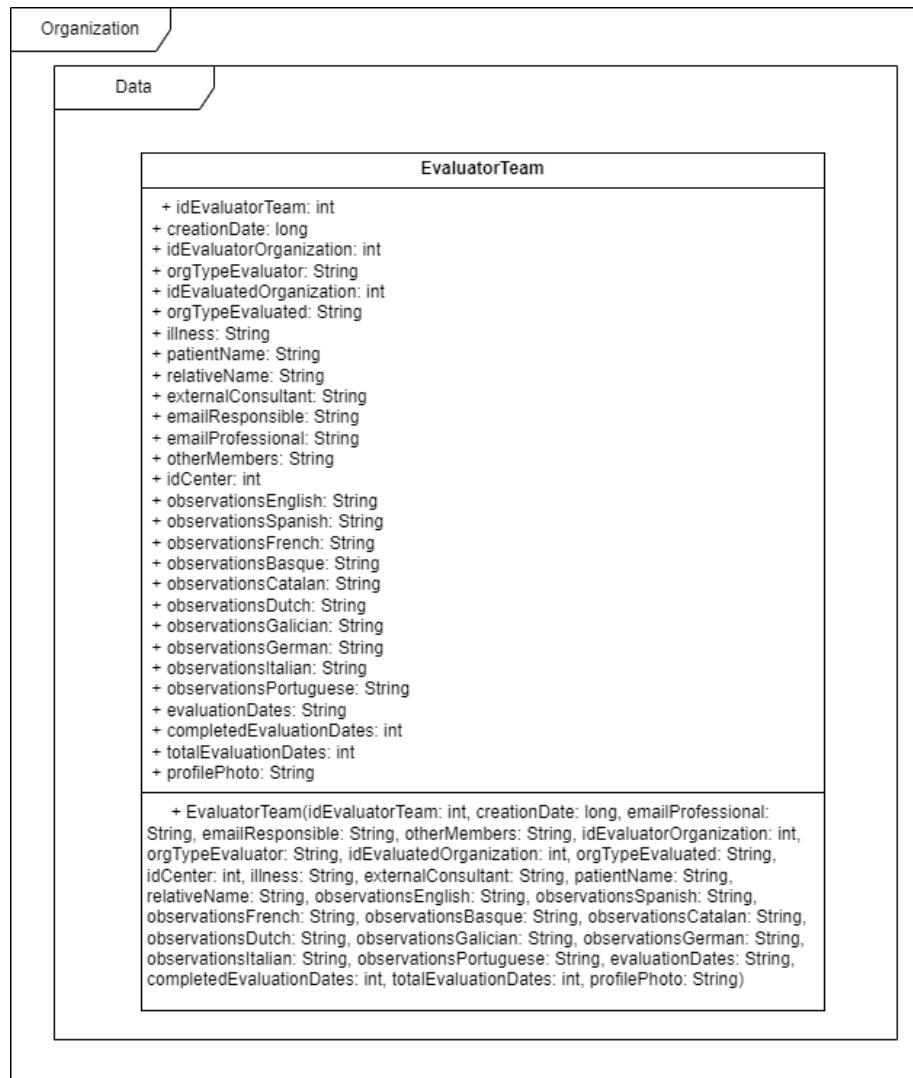


Figura C.20: Clase EvaluatorTeam

- **EvaluatorTeam:** Esta entidad se encarga de almacenar información sobre los equipos evaluadores, utilizando para ello los siguientes campos:
 - **idEvaluatorTeam:** Almacena el identificador del equipo evaluador.Es de tipo INT.
 - **creationDate:** Almacena la fecha de creación del equipo.Es de tipo long o BIGINT.
 - **externalConsultant:** Almacena el nombre del consultor del equipo.Es de tipo String o VARCHAR(500).
 - **emailResponsible:** Almacena la dirección de correo electrónico del responsable del equipo.Es de tipo String o VARCHAR(500).
 - **emailProfessional:** Almacena la dirección de correo electrónico del profesional del equipo.Es de tipo String o VARCHAR(500).
 - **otherMembers:** Almacena el nombre o nombres de otros miembros dentro del equipo evaluador. Es de tipo String o VARCHAR(500).
 - **idEvaluatorOrganization:** Almacena el identificador de la organización evaluadora a la que pertenece el equipo.Es de tipo INT.
 - **orgTypeEvaluator:** Indica el tipo de la organización del equipo, siempre de valor EVALUATOR.Es de tipo String o VARCHAR(50).
 - **idEvaluatedOrganization:** Almacena el identificador de la organización evaluada a la que pertenece el equipo.Es de tipo INT.
 - **orgTypeEvaluated:** Indica el tipo de la organización del equipo, siempre de valor EVALUATED..Es de tipo String o VARCHAR(50).
 - **idCenter:** Almacena el identificador del centro o servicio de la organización evaluada a la que pertenece el equipo.Es de tipo INT.
 - **illness:** Almacena el tipo de enfermedad relacionada con la organización del equipo.Es de tipo String o VARCHAR(50).
 - **patientName:** Almacena el nombre del paciente evaluado por el equipo.Es de tipo String o VARCHAR(500).
 - **relativeName:** Almacena el nombre del familiar del paciente evaluado por el equipo.Es de tipo String o VARCHAR(500).
 - **observationsSpanish:** Almacena las observaciones del equipo evaluador en español. Es de tipo String o VARCHAR(5000).

- **observationsEnglish:** Almacena las observaciones del equipo evaluador en inglés. Es de tipo **String** o **VARCHAR(5000)**.
- **observationsFrench:** Almacena las observaciones del equipo evaluador en francés. Es de tipo **String** o **VARCHAR(5000)**.
- **observationsBasque:** Almacena las observaciones del equipo evaluador en euskera. Es de tipo **String** o **VARCHAR(5000)**.
- **observationsCatalan:** Almacena las observaciones del equipo evaluador en catalán. Es de tipo **String** o **VARCHAR(5000)**.
- **observationsDutch:** Almacena las observaciones del equipo evaluador en neerlandés. Es de tipo **String** o **VARCHAR(5000)**.
- **observationsGalician:** Almacena las observaciones del equipo evaluador en gallego. Es de tipo **String** o **VARCHAR(5000)**.
- **observationsGerman:** Almacena las observaciones del equipo evaluador en alemán. Es de tipo **String** o **VARCHAR(5000)**.
- **observationsItalian:** Almacena las observaciones del equipo evaluador en italiano. Es de tipo **String** o **VARCHAR(5000)**.
- **observationsPortuguese:** Almacena las observaciones del equipo evaluador en portugués. Es de tipo **String** o **VARCHAR(5000)**.
- **evaluationDates:** Almacena las diferentes fechas de evaluación del equipo evaluador, en formato timestamp y separadas por comas. Es de tipo **String** o **VARCHAR(MAX)**.
- **completedEvaluationDates:** Almacena la cantidad de fechas de evaluación completadas. Es de tipo entero.
- **totalEvaluationDates:** Almacena la cantidad de fechas de evaluación totales. Es de tipo entero.
- **profilePhoto:** Almacena el nombre del fichero de la fotografía de perfil del equipo evaluador, para que pueda acceder Azure Blob Storage. Es de tipo **String** o **VARCHAR(MAX)**.
- **PRIMARY KEY:** La clave principal de esta tabla está compuesta por el campo **idEvaluatorTeam**, **idEvaluatorOrganization**, **orgTypeEvaluator**, **idEvaluatedOrganization**, **orgTypeEvaluated**, **idCenter** e **illness**.
- **FOREIGN KEY:** Establece una relación con la tabla **Organization** a través de los campos **idEvaluatorOrganization**, **orgTypeEvaluator**, **illness**.

- **FOREIGN KEY:** Establece una relación con la tabla **Center** a través de los campos `idEvaluatedOrganization`, `orgTypeEvaluated`, `idCenter` e `illness`.
- **FOREIGN KEY:** Establece una relación con la tabla **User** a través del campo `emailResponsible`.
- **FOREIGN KEY:** Establece una relación con la tabla **User** a través del campo `emailProfessional`.

Diseño del sistema experto

En cuanto al diseño experto, como se ha mencionado anteriormente en la memoria, se ha utilizado *NRules*. Pero como se ha mencionado con anterioridad, las reglas no se lanzan en un determinado orden, por lo que es necesario tener un control muy estricto sobre las reglas que se necesitan marcar. Para ello, en el lado del servidor, se ha creado un paquete **ExpertSystem**, el cual está compuesto por todas las reglas y una clase singleton que se encarga de lanzarlas.

- **Clase singleton:** La clase singleton se llama **ExpertSystem**, la cual es una clase Singleton que se encarga de gestionar las reglas a voluntad:

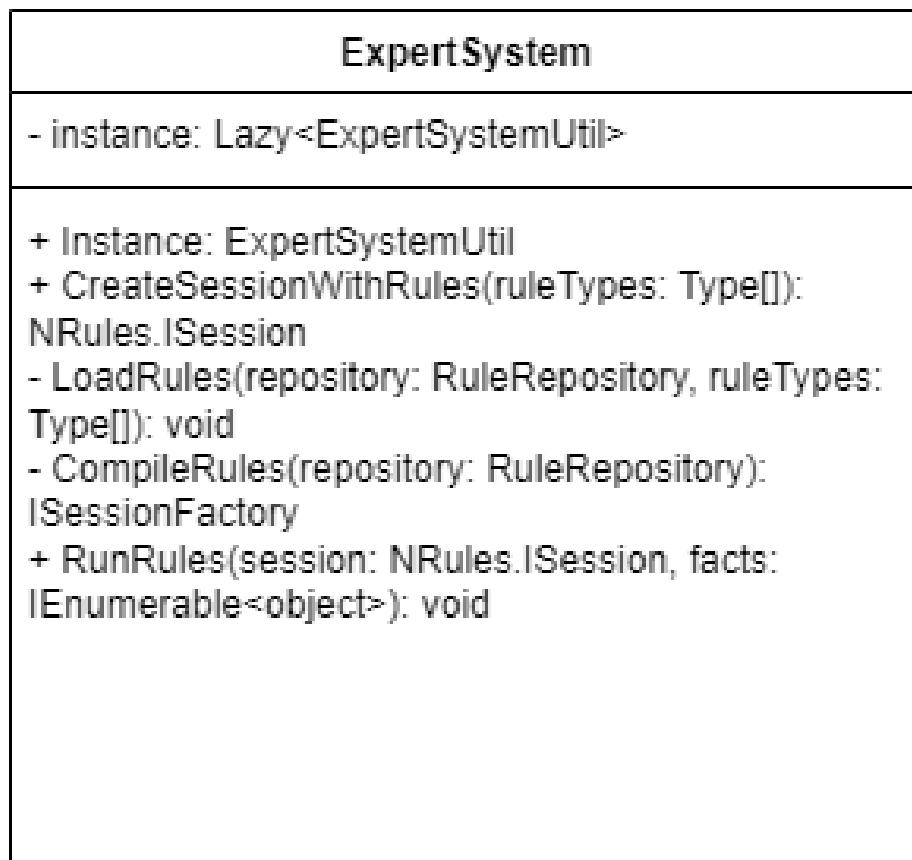


Figura C.21: Clase ExpertSystem

0.9

- **Instance:** Es la instancia singleton que es cargada cada vez que se necesita.
- **CreateSessionWithRules:** Es el método que se encarga de crear una sesión con unas reglas pasadas por parámetro.
- **LoadRules:** Se cargan las reglas que se encuentran disponibles en el repositorio pasado por parámetro, además de las clases correspondientes a dichas reglas.
- **CompileRules:** Se encarga de compilar las reglas que tiene el repositorio pasado por parámetro.
- **RunRules:** Se encarga de correr las reglas de una determinada sesión con unos hechos pasados por parámetro.

- **Estructura de las reglas:** Las clases de las dieciocho reglas del sistema experto se estructuran en dos métodos:

- Método **Define**: Es el método que se encarga de definir las condiciones que están detrás de una determinada regla y de obtener los hechos necesarios para poder comprobar dicha condición para que la regla pueda ser lanzada.
- Método **Set**: Se encarga de establecer una variable o conjunto de variables a aquellos hechos que cumplan con la regla. Se le pasa todo lo necesario a partir de **Define** y se establece una etiqueta o un valor.

```

1      namespace OTEAServer.ExpertSystem
2      {
3          public class
4              ↪ RuleExcellentIndicatorEvaluation
5                  : Rule
6          {
7              public override void Define()
8              {
9                  IndicatorsEvaluation indEval =
10                     ↪ default;
11
12                  When()
13                      .Match<IndicatorsEvaluation>(())
14                          ↪ => indEval, indEval => (
15                              ↪ indEval.evaluationType ==
16                                  "COMPLETE" && indEval.
17                                  totalScore >= 200) || (
18                                      ↪ indEval.evaluationType ==
19                                          "SIMPLE" && indEval.
20                                          totalScore >= 143));
21
22                  Then()
23                      .Do(ctx=>SetExcellent(
24                          ↪ indEval));
25              }
26
27              private static void SetExcellent(
28                  ↪ IndicatorsEvaluation indEval)
29                  {
30                      indEval.level = "EXCELLENT";
31                  }
32      }
```

```

19          }
20      }

```

Por lo tanto, para poder ejecutar las reglas, se tiene que llamar siempre a la instancia de `ExpertSystem`, como por ejemplo, a la hora de asignar, primero el estado de los indicadores y después si necesitan o no necesitan estar en el plan de mejora:

```

1     [HttpPost("fromList")]
2     [Authorize(Policy = "Administrator")]
3     public IActionResult CreateRegs([FromBody] List<
4         ↪ IndicatorsEvaluationIndicatorReg> regs, [
5         ↪ FromHeader] string Authorization)
6     {
7         if (regs == null)
8         {
9             return BadRequest();
10        }
11        try
12        {
13            var session=ExpertSystemUtil.Instance.
14                ↪ CreateSessionWithRules(typeof(
15                    ↪ RuleIndicatorInProcess), typeof(
16                    ↪ RuleIndicatorInStart), typeof(
17                    ↪ RuleIndicatorReached));
18            List<object> facts=new List<object>();
19            facts.Add(regs);
20            ExpertSystemUtil.Instance.RunRules(session,
21                ↪ facts);
22            session = ExpertSystemUtil.Instance.
23                ↪ CreateSessionWithRules(typeof(
24                    ↪ RuleDoesntRequireImprovementPlan),
25                    ↪ typeof(RuleRequiresImprovementPlan));
26            facts = new List<object>();
27            facts.Add(regs);
28            ExpertSystemUtil.Instance.RunRules(session,
29                ↪ facts);
30
31            foreach (IndicatorsEvaluationIndicatorReg
32                ↪ reg in regs)

```

```

24
25     if (reg == null) { continue; }
26     IndicatorsEvaluationIndicatorReg aux =
27         ↪ _context.
28         ↪ IndicatorsEvaluationsIndicatorsRegs
29         ↪ .FirstOrDefault(r => r.
30             ↪ evaluationDate == reg.
31             ↪ evaluationDate && r.
32             ↪ idEvaluatorTeam == reg.
33             ↪ idEvaluatorTeam && r.
34             ↪ idEvaluatorOrganization == reg.
35             ↪ idEvaluatorOrganization && r.
36             ↪ orgTypeEvaluator == reg.
37             ↪ orgTypeEvaluator && r.
38             ↪ idEvaluatedOrganization == reg.
39             ↪ idEvaluatedOrganization && r.
40             ↪ orgTypeEvaluated == reg.
41             ↪ orgTypeEvaluated && r.illness ==
42             ↪ reg.illness && r.idCenter == reg.
43             ↪ idCenter && r.idSubSubAmbit ==
44             ↪ reg.idSubSubAmbit && r.idSubAmbit
45             ↪ == reg.idSubAmbit && r.idAmbit
46             ↪ == reg.idAmbit && r.idIndicator
47             ↪ == reg.idIndicator && r.
48             ↪ indicatorVersion == reg.
49             ↪ indicatorVersion && r.
50             ↪ evaluationType == reg.
51             ↪ evaluationType);
52
53     if (aux == null)
54     {
55         ↪ _context.
56         ↪ IndicatorsEvaluationsIndicatorsRegs
57         ↪ .Add(reg);
58     }
59     else {
60         aux.evaluationDate = reg.
61             ↪ evaluationDate;
62         aux.idEvaluatorTeam = reg.
63             ↪ idEvaluatorTeam;
64         aux.idEvaluatorOrganization = reg.
65             ↪ idEvaluatorOrganization;

```

```

36     aux.orgTypeEvaluator = reg.
37         ↳ orgTypeEvaluator;
38     aux.idEvaluatedOrganization = reg.
39         ↳ idEvaluatedOrganization;
40     aux.orgTypeEvaluated = reg.
41         ↳ orgTypeEvaluated;
42     aux.illness = reg.illness;
43     aux.idCenter = reg.idCenter;
44     aux.idSubSubAmbit = reg.
45         ↳ idSubSubAmbit;
46     aux.idSubAmbit = reg.idSubAmbit;
47     aux.idAmbit = reg.idAmbit;
48     aux.idIndicator = reg.idIndicator;
49     aux.indicatorVersion = reg.
50         ↳ indicatorVersion;
51     aux.evaluationType = reg.
52         ↳ evaluationType;

53     aux.observationsSpanish = reg.
54         ↳ observationsSpanish;
55     aux.observationsEnglish = reg.
56         ↳ observationsEnglish;
57     aux.observationsFrench = reg.
58         ↳ observationsFrench;
59     aux.observationsBasque = reg.
60         ↳ observationsBasque;
61     aux.observationsCatalan = reg.
62         ↳ observationsCatalan;
63     aux.observationsDutch = reg.
64         ↳ observationsDutch;
65     aux.observationsGalician = reg.
66         ↳ observationsGalician;
67     aux.observationsGerman = reg.
68         ↳ observationsGerman;
69     aux.observationsItalian = reg.
70         ↳ observationsItalian;
71     aux.observationsPortuguese = reg.
72         ↳ observationsPortuguese;
73     aux.numEvidencesMarked = reg.
74         ↳ numEvidencesMarked;
75     aux.status = reg.status;
76     aux.requiresImprovementPlan = reg.
77         ↳ requiresImprovementPlan;

```

```
61          }
62      }
63      _context.SaveChanges();
64      return Ok(regs);
65  }
66  catch (Exception ex)
67  {
68      return BadRequest(ex.Message);
69  }
70 }
71 }
```

Como se puede comprobar, los pasos que sigue en este caso el sistema experto son los siguientes:

1. En primer lugar se crea la sesión llamada `session` con `ExpertSystemUtil.Instance.CreateSession` pasando las clases de las reglas `RuleIndicatorInStart`, `RuleIndicatorInProcess` y `RuleIndicatorReached`.
2. Posteriormente, se añaden como hechos los diferentes registros de indicadores.
3. Más adelante, mediante `ExpertSystemUtil.Instance.RunRules`, pasamos el objeto de sesión y la lista de hechos para lanzar las diferentes reglas.
4. Tras haber lanzado las reglas, repetimos una vez más el procedimiento, pero esta vez con las reglas `RuleDoesntRequireImprovementPlan` y `RuleRequiresImprovementPlan`.

C.3. Diseño procedimental

En cuanto al diseño procedimental de la aplicación, se tiene que tener en cuenta qué se espera de cada una de las características que tiene la aplicación. El objetivo fundamental de realizar este tipo de diseño es saber los pasos a seguir para realizar cada una de las funcionalidades más importantes para cada uno de los algoritmos a seguir. Para realizar el diseño procedimental se ha establecido como base la realización de simples diagramas de flujo, que indiquen cómo tiene que realizarse paso a paso estos algoritmos. No se requieren de algoritmos bastante sofisticados para la realización de esta aplicación, por lo que se antoja bastante más sencillo conocer los pasos a seguir. En nuestro caso, tenemos varias funcionalidades diferentes:

- **Iniciar sesión:** Para el inicio de sesión se ha decidido implementar este diagrama de flujo:

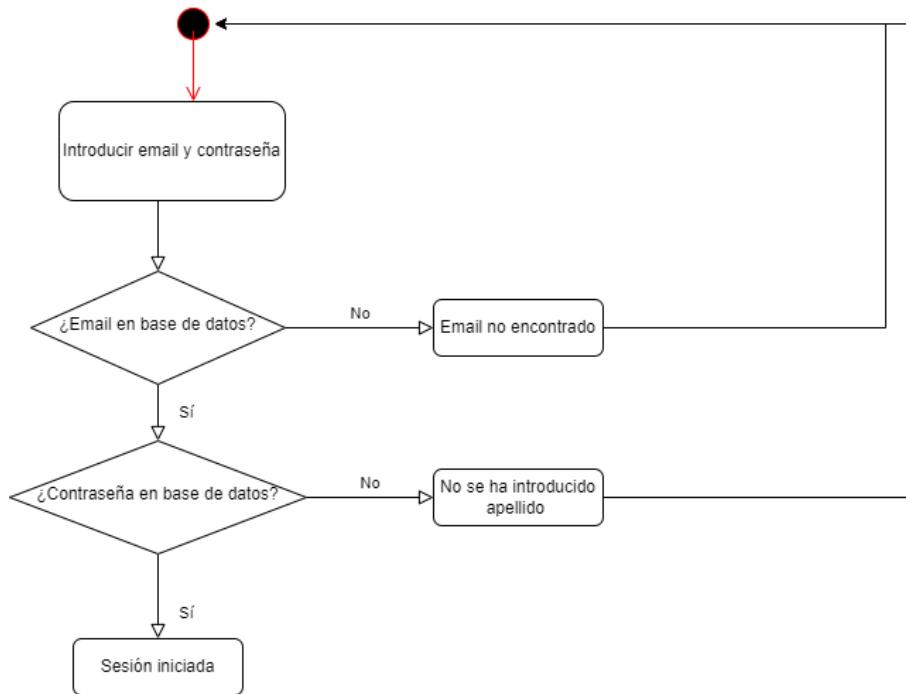


Figura C.22: Diagrama de flujo para el inicio de sesión

En este caso se inicializa con la introducción de los campos de usuario y de contraseña. Posteriormente, tras el intento de inicio de sesión, se intenta buscar el usuario con las credenciales de la base de datos. En primer lugar busca si el email está correctamente introducido, en caso de que el email no esté bien introducido, lanza el error y vuelve a empezar. En caso contrario, comprueba la contraseña. Si no es correcta, lanza el error y vuelve a empezar, y en caso contrario se consigue un inicio de sesión satisfactorio.

- **Registrar usuario:** Para el registro de usuario se ha decidido implementar este diagrama de flujo:

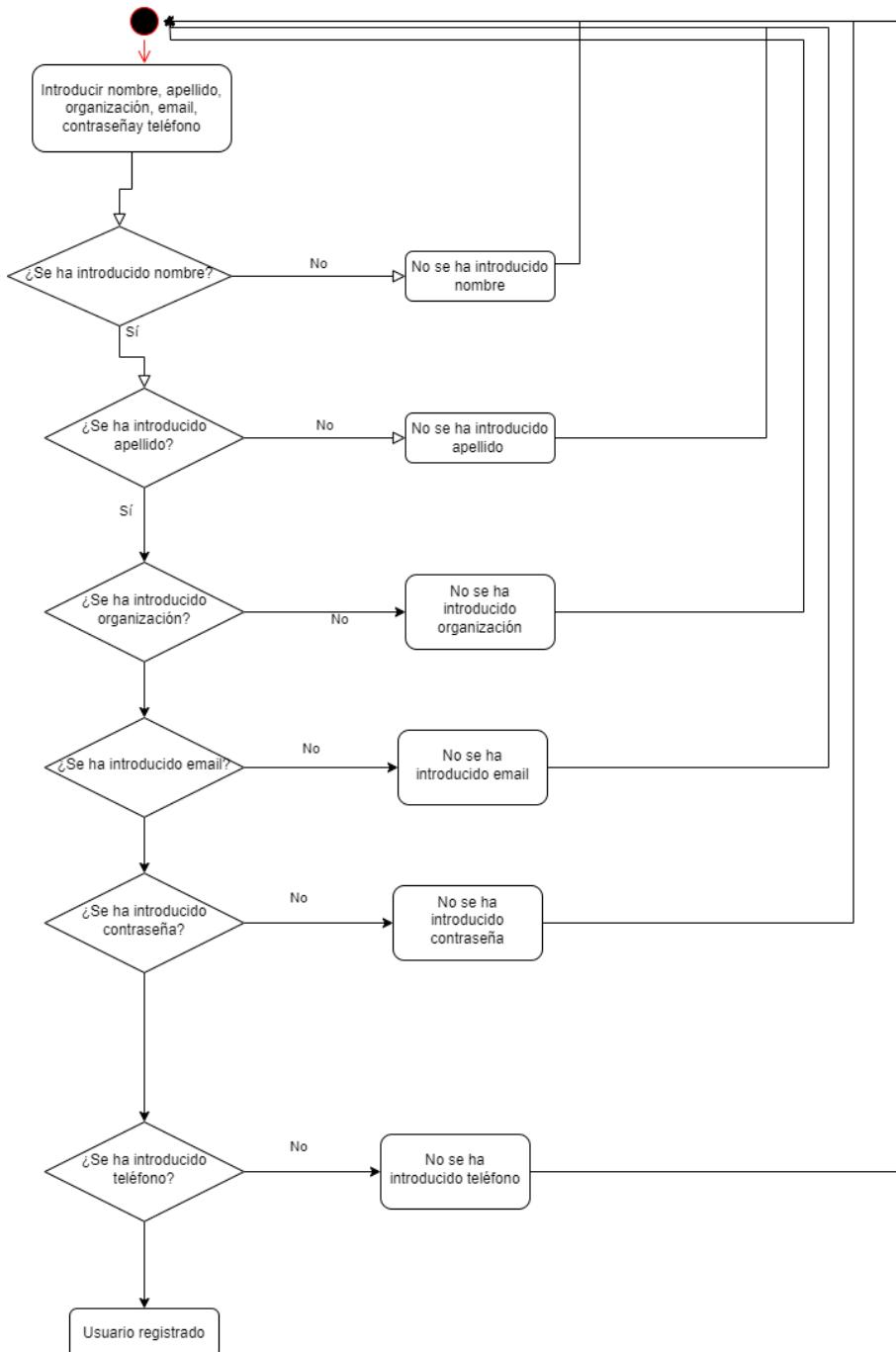
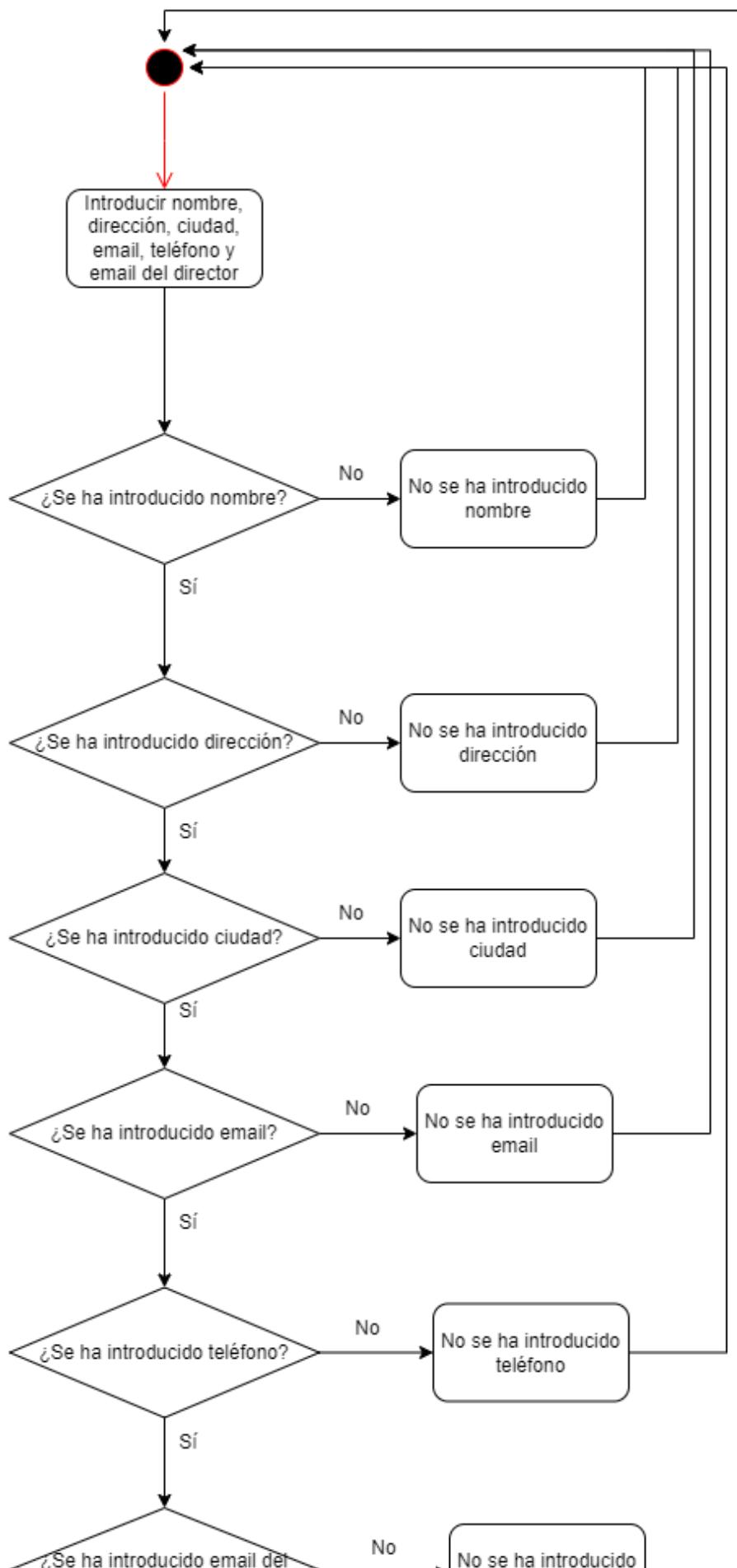


Figura C.23: Diagrama de flujo para el registro del usuario

Al igual que en el caso anterior, la dinámica es muy similar, con la diferencia que se necesitan introducir más campos de forma obligatoria.

En primer lugar se comprueba si se ha introducido el nombre del usuario. Si no se ha introducido vuelve al punto de arranque y en caso contrario pasa a comprobar el apellido. En caso de que no se haya introducido ningún apellido vuelve al punto de arranque y en caso contrario pasa a comprobar la organización. En caso de que no se haya introducido ninguna organización vuelve al punto de arranque y en caso contrario pasa a comprobar el email. Si no se ha introducido el email vuelve al punto de arranque y en caso contrario pasa a comprobar la contraseña. Si no se ha introducido la contraseña vuelve al punto de arranque y en caso contrario pasa a comprobar el teléfono. Si no se ha introducido ningún teléfono, vuelve al punto de arranque y en caso contrario se considera el registro del usuario como realizado.

- **Registrar organización:** Para el registro de la organización se cumple la misma dinámica que en la actividad anterior. Cabe resaltar que el campo de *información* no se ha introducido en este diagrama debido a que no es un campo que se considere de obligatorio llenado, más allá que para comentar algún elemento adicional. Se ha seguido el siguiente diagrama de flujo:



Como se puede apreciar, en primer lugar se comprueba si se ha introducido el nombre de la organización. Si no se ha introducido vuelve al punto de arranque y en caso contrario pasa a comprobar la dirección. En caso de que no se haya introducido ninguna dirección vuelve al punto de arranque y en caso contrario pasa a comprobar la ciudad. Si no se ha introducido ninguna ciudad vuelve al punto de arranque y en caso contrario pasa a comprobar el email. Si no se ha introducido el email vuelve al punto de arranque y en caso contrario pasa a comprobar el teléfono. Si no se ha introducido ningún teléfono, vuelve al punto de arranque y en caso contrario se pasa a comprobar el director de la organización. Si no se ha introducido ningún director, se vuelve al punto de arranque y en caso contrario, se considera el registro de la organización como realizado.

- **Registrar equipo evaluador:** Para el registro del equipo evaluador se sigue una dinámica muy similar al del caso anterior, siguiendo para ello el siguiente diagrama de flujo:

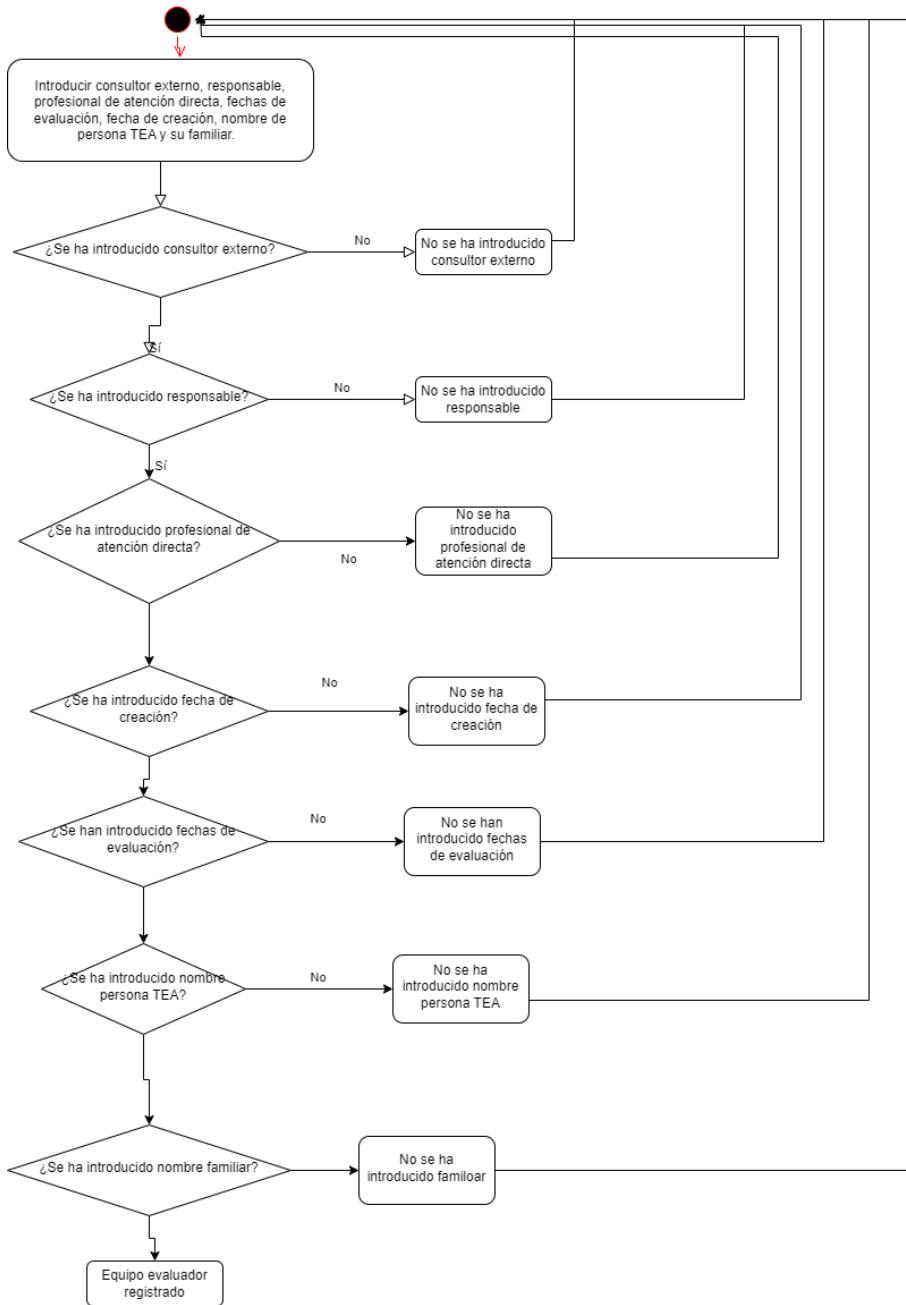


Figura C.25: Diagrama de flujo para el registro del equipo evaluador

Como se puede apreciar, tras la introducción del consultor externo, el responsable, el profesional de atención directa, las fechas de evaluación, la fecha de creación, el nombre de la persona con TEA y el nombre

de su familiar, se busca realizar la comprobación de dichos campos. En primer lugar si no se ha introducido un consultor externo vuelve al punto de arranque y en caso contrario comprueba el responsable. Si no se ha introducido ningún responsable, se vuelve al punto de arranque y en caso contrario se pasa a comprobar el profesional de atención directa. Si no se ha introducido ningún profesional de atención directa, se vuelve al punto de arranque y en caso contrario se pasan a comprobar las fechas de evaluación y de creación del equipo evaluador. Si no se ha introducido alguna de esas fechas, se vuelve al punto de arranque y en caso contrario se pasa a comprobar los nombres tanto de la persona con TEA como el de su familiar. En caso de que no se haya introducido alguno de los nombres, se vuelve al punto de arranque y en caso contrario se considera como registrado el equipo evaluador.

- **Realizar test de indicadores:** La realización del test de indicadores básica se considera como un proceso iterativo, el cual sigue el siguiente diagrama de flujo:

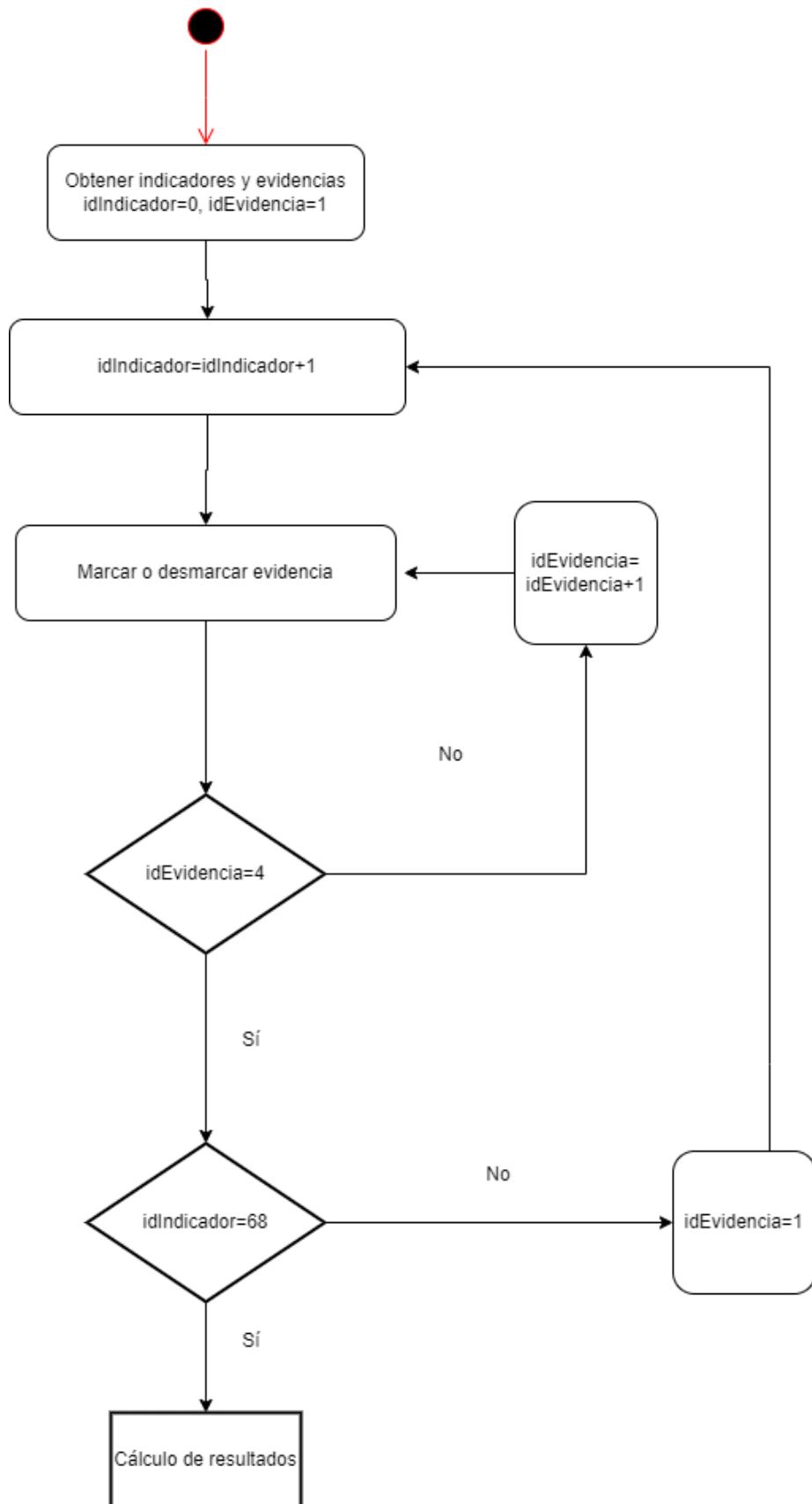


Figura C.26: Diagrama de flujo para la realización del test de indicadores

. Como se puede comprobar, tras la obtención de los indicadores y evidencias con sus respectivos contadores, ambos se van actualizando a medida que se van avanzando en los test de indicadores. Si se ha superado el número de evidencias, refresca el número del indicador y retorna al valor 1 como valor de evidencia. En caso contrario ya habríamos terminado con el test y se obtendrían los resultados.

C.4. Diseño arquitectónico

Como se ha mencionado con anterioridad, la arquitectura utilizada para el desarrollo de esta aplicación es una arquitectura de tipo *Modelo-Vista-Controlador*:

- En el lado del servidor tenemos tres tipos de clase diferentes segregadas por el paquete al que pertenecen. Dichos paquetes son:
 - **Models:** Este paquete incluye todos los modelos de las clases que son incluidos en el servidor. Ejemplo:

```

1           using Newtonsoft.Json;
2
3 namespace OTEAServer.Models
4 {
5     public class Indicator
6     {
7         public Indicator(int idIndicator,
8             ↳ string indicatorType, string
9                 ↳ descriptionEnglish, string
10                ↳ descriptionSpanish, string
11                  ↳ descriptionFrench, int
12                    ↳ indicatorPriority, int
13                      ↳ indicatorVersion) {
14             this.indicatorId = idIndicator;
15             this.indicatorType = indicatorType;
16             this.descriptionEnglish =
17                 ↳ descriptionEnglish;
18             this.descriptionSpanish =
19                 ↳ descriptionSpanish;
20             this.descriptionFrench =
21                 ↳ descriptionFrench;
22             this.indicatorPriority =
23                 ↳ indicatorPriority;

```

```

14         this.indicatorVersion =
15             ↪ indicatorVersion;
16     }
17
18     [JsonProperty("indicatorId")]
19     public int indicatorId { get; set; }
20
21     [JsonProperty("indicatorType")]
22     public string indicatorType { get; set;
23             ↪ }
24
25     [JsonProperty("descriptionEnglish")]
26     public string descriptionEnglish { get;
27             ↪ set; }
28
29     [JsonProperty("descriptionSpanish")]
30     public string descriptionSpanish { get;
31             ↪ set; }
32
33     [JsonProperty("descriptionFrench")]
34     public string descriptionFrench { get;
35             ↪ set; }
36
37     [JsonProperty("indicatorPriority")]
38     public int indicatorPriority { get; set
39             ↪ ; }

```

Como se comprueba en el siguiente ejemplo, todos los campos del modelo de la clase `Indicator` tienen la propiedad de *Newtonsoft.Json* denominada `JsonProperty`, la cual se utiliza para la serialización de los datos de este tipo para enviarlos en formato JSON.

- **Controllers:** Este paquete incluye todas las clases que estructuran los endpoints de la web app, utilizándose para realizar las operaciones en el servidor. Ejemplo:

```
1         using Microsoft.AspNetCore.
2             ↪ Authorization;
3 using Microsoft.AspNetCore.Http;
4 using Microsoft.AspNetCore.Mvc;
5 using OTEAServer.Misc;
6 using OTEAServer.Models;
7 using System.Security.Policy;
8 using System.Text.Json;
9
10 namespace OTEAServer.Controllers
11 {
12     /// <summary>
13     /// Controller class for indicators
14     /// Author: Pablo Ahita del Barrio
15     /// Version: 1
16     /// </summary>
17
18     [ApiController]
19     [Route("Indicators")]
20     public class IndicatorsController :
21         ↪ ControllerBase
22     {
23         /// <summary>
24         /// Database context
25         /// </summary>
26         private readonly DatabaseContext
27             ↪ _context;
28
29         public IndicatorsController(
30             ↪ DatabaseContext context)
31         {
32             _context = context;
33         }
34
35         /// <summary>
36         /// Method that obtains all the
37             ↪ indicators
38         /// </summary>
39         /// <param name="evaluationType">
40             ↪ Evaluation type</param>
41         /// <returns>Indicators list</returns>
```

```

38         [HttpGet("all")]
39         [Authorize]
40         public IActionResult GetAll([FromQuery]
41             ↪ string evaluationType, [
42             ↪ FromHeader] string Authorization)
43     {
44         try
45     {
46         List<JsonDocument> allData=new
47             ↪ List<JsonDocument>();
48         var indicators = _context.
49             ↪ Indicators
50         .Where(i => i.
51             ↪ evaluationType ==
52             ↪ evaluationType && i.
53             ↪ isActive == 1)
54         .OrderBy(i => i.idIndicator
55             ↪ )
56         .ToList();
57         foreach (var indicator in
58             ↪ indicators)
59     {
60         String ind= "{\"idIndicator":
61             ↪ \"\":\"" + indicator.
62             ↪ idIndicator + "\","
63             ↪ "\"indicatorType\":\""
64             ↪ + indicator.
65             ↪ indicatorType + "
66             ↪ \","
67             ↪ "\"idSubSubAmbit\":\""
68             ↪ + indicator.
69             ↪ idSubSubAmbit + "
70             ↪ \","
71             ↪ "\"idSubAmbit\":\""
72             ↪ + indicator.
73             ↪ idSubAmbit + "\",
74             ↪ "
75             ↪ "\"idAmbit\":\""
76             ↪ + indicator.idAmbit
77             ↪ + "\","
78             ↪ "
79             ↪ "\"descriptionSpanish"
80             ↪ \"\":\"" +
81             ↪ indicator.

```

```

      ↳ descriptionSpanish
      ↳ + "\," +
57   "=\""descriptionEnglish
      ↳ \":\\"" +
      ↳ indicator.
      ↳ descriptionEnglish
      ↳ + "\," +
      ↳ descriptionFrench
      ↳ \":\\"" +
      ↳ indicator.
      ↳ descriptionFrench
      ↳ + "\," +
58   "=\""descriptionBasque
      ↳ \":\\"" +
      ↳ indicator.
      ↳ descriptionBasque
      ↳ + "\," +
59   "=\""descriptionCatalan
      ↳ \":\\"" +
      ↳ indicator.
      ↳ descriptionCatalan
      ↳ + "\," +
60   "=\""descriptionDutch
      ↳ \":\\"" +
      ↳ indicator.
      ↳ descriptionDutch
      ↳ + "\," +
61   "=\""descriptionGalician
      ↳ \":\\"" +
      ↳ indicator.
      ↳ descriptionGalician
      ↳ + "\," +
62   "=\""descriptionGerman
      ↳ \":\\"" +
      ↳ indicator.
      ↳ descriptionGerman
      ↳ + "\," +
63   "=\""descriptionItalian
      ↳ \":\\"" +
      ↳ indicator.
      ↳ descriptionItalian
      ↳ + "\," +
64

```

```

65           " \\
66             ↪ descriptionPortuguese
66             ↪ \":\" + 
66             ↪ indicator.
66             ↪ descriptionPortuguese
66             ↪ + "\\",," +
66           "\"indicatorPriority
66             ↪ \":\" + 
66             ↪ indicator.
66             ↪ indicatorPriority
66             ↪ + "\\",," +
67           "\"indicatorVersion
67             ↪ \":\" + 
67             ↪ indicator.
67             ↪ indicatorVersion
67             ↪ + "\\",," +
68           "\"isActive\":\" :\" + 
68             ↪ indicator.
68             ↪ indicatorVersion
68             ↪ + "\\",," +
69           "\"evaluationType\":\" :\" "
69             ↪ + indicator.
69             ↪ evaluationType +
69             ↪ "\")";
70           allData.Add(JsonDocument.
70             ↪ Parse(ind));
71         }
72
73         int tamEvidences = 0;
74
75         if (evaluationType == "COMPLETE
75             ↪ ") {
76           var evidences = _context.
76             ↪ Evidences
77             .Where(e => e.
77               ↪ evaluationType ==
77               ↪ evaluationType)
78             .OrderBy(e => e.
78               ↪ idIndicator)
79             .ThenBy(e => e.
79               ↪ idEvidence)
80             .ToList();

```

```

81         foreach (var evidence in
82             ↪ evidences)
83 {
84     String ev = "{\""
85         ↪ idEvidence\":\""
86         ↪ + evidence.
87         ↪ idEvidence + "\","
88         ↪ +
89         ↪ " \
80         ↪ idIndicator
81         ↪ \":\""
82         ↪ + evidence.
83         ↪ idIndicator
84         ↪ + "\","
85         ↪ +
86         ↪ " \
87         ↪ indicatorType
88         ↪ \":\""
89         ↪ + evidence.
90         ↪ indicatorType
91         ↪ + "\","
92         ↪ +
93         ↪ " \
94         ↪ idSubSubAmbit
95         ↪ \":\""
96         ↪ + evidence.
97         ↪ idSubSubAmbit
98         ↪ + "\","
99         ↪ +
100         ↪ " \
101         ↪ idSubAmbit
102         ↪ \":\""
103         ↪ + evidence.
104         ↪ idSubAmbit
105         ↪ + "\","
106         ↪ +
107         ↪ " \
108         ↪ idAmbit\":\""
109         ↪ +
110         ↪ evidence.
111         ↪ idAmbit +
112         ↪ "\","
113         ↪ +
114         ↪ " \
115         ↪ descriptionSpanish
116         ↪ \":\""
117         ↪ + evidence.

```

```

90
    ↪ descriptionSpanish
    ↪ + "\", "
    ↪ +
    " \
    ↪ descriptionEnglish
    ↪ "\" : \" " +
    ↪ evidence.
    ↪ descriptionEnglish
    ↪ + "\", "
    ↪ +
    " \
91
    ↪ descriptionFrench
    ↪ "\" : \" " +
    ↪ evidence.
    ↪ descriptionFrench
    ↪ + "\", "
    ↪ +
    " \
92
    ↪ descriptionBasque
    ↪ "\" : \" " +
    ↪ evidence.
    ↪ descriptionBasque
    ↪ + "\", "
    ↪ +
    " \
93
    ↪ descriptionCatalan
    ↪ "\" : \" " +
    ↪ evidence.
    ↪ descriptionCatalan
    ↪ + "\", "
    ↪ +
    " \
94
    ↪ descriptionDutch
    ↪ "\" : \" " +
    ↪ evidence.
    ↪ descriptionDutch
    ↪ + "\", "
    ↪ +
    " \
95
    ↪ descriptionGalician
    ↪ "\" : \" " +
    ↪ evidence.
    ↪ descriptionGalician

```

```

    ↪ + "\\", "
    ↪ +
  " \
    ↪ descriptionGerman
    ↪ ":"\" + "
    ↪ evidence.
    ↪ descriptionGerman
    ↪ + "\\", "
    ↪ +
  " \
    ↪ descriptionItalian
    ↪ ":"\" + "
    ↪ evidence.
    ↪ descriptionItalian
    ↪ + "\\", "
    ↪ +
  " \
    ↪ descriptionPortuguese
    ↪ ":"\" + "
    ↪ evidence.
    ↪ descriptionPortuguese
    ↪ + "\\", "
    ↪ +
  " \
    ↪ evidenceValue
    ↪ ":"\" + "
    ↪ evidence.
    ↪ evidenceValue
    ↪ + "\\", "
    ↪ +
  " \
    ↪ indicatorVersion
    ↪ ":"\" + "
    ↪ evidence.
    ↪ indicatorVersion
    ↪ + "\\", "
    ↪ +
  " \
    ↪ evaluationType
    ↪ ":"\" + "
    ↪ evidence.
    ↪ evaluationType
    ↪ + "\"}";

```

```

102                         allData.Add(
103                             ↪ JsonDocument.
104                             ↪ Parse(ev));
105                         }
106
107
108         var ambits = _context.Ambits.
109             ↪ ToList();
110         foreach (var ambit in ambits) {
111             String amb= "{$idAmbit
112                 ↪ \":\\" + ambit.
113                 ↪ idAmbit + "\","
114                 ↪ "\"
115                 ↪ descriptionSpanish
116                 ↪ \":\\" +
117                 ↪ ambit.
118                 ↪ descriptionSpanish
119                 ↪ + "\\","
120                 ↪ "\"
121                 ↪ descriptionEnglish
122                 ↪ \":\\" +
123                 ↪ ambit.
124                 ↪ descriptionEnglish
125                 ↪ + "\\","
126                 ↪ "\"
127                 ↪ descriptionFrench
128                 ↪ \":\\" +
129                 ↪ ambit.
130                 ↪ descriptionFrench
131                 ↪ + "\\","
132                 ↪ "\"
133                 ↪ descriptionBasque
134                 ↪ \":\\" +
135                 ↪ ambit.
136                 ↪ descriptionBasque
137                 ↪ + "\\","
138                 ↪ "\"
139                 ↪ descriptionCatalan
140                 ↪ \":\\" +
141                 ↪ ambit.

```

```

    ↪ descriptionCatalan
    ↪ + "\," +
116      "\"descriptionDutch
    ↪ \":\\"" +
    ↪ ambit.
    ↪ descriptionDutch
    ↪ + "\," +
    ↪ "\"
117      descriptionGalician
    ↪ \":\\"" +
    ↪ ambit.
    ↪ descriptionGalician
    ↪ + "\," +
    ↪ "\"
118      descriptionGerman
    ↪ \":\\"" +
    ↪ ambit.
    ↪ descriptionGerman
    ↪ + "\," +
    ↪ "\"
119      descriptionItalian
    ↪ \":\\"" +
    ↪ ambit.
    ↪ descriptionItalian
    ↪ + "\," +
    ↪ "\"
120      descriptionPortuguese
    ↪ \":\\"" +
    ↪ ambit.
    ↪ descriptionPortuguese
    ↪ + "\}" ;
121      allData.Add(JsonDocument.
    ↪ Parse(amb));
122  }
123
124  var subAmbits = _context.
    ↪ SubAmbits.ToList();
125  foreach (var subAmbit in
    ↪ subAmbits) {
126      String subAmb = "{\"
    ↪ idSubAmbit\":\\"" +
    ↪ subAmbit.idSubAmbit +
    ↪ "\," +

```

127

```
"\" idAmbit \" : \""
  ↪ " "
  ↪ subAmbit.
  ↪ idAmbit +
  ↪ "\", " +
```

128

```
"\""
  ↪ descriptionSpanish
  ↪ "\" : \" " +
  ↪ subAmbit.
  ↪ descriptionSpanish
  ↪ " ", " +
  ↪ +
```

129

```
"\""
  ↪ descriptionEnglish
  ↪ "\" : \" " +
  ↪ subAmbit.
  ↪ descriptionEnglish
  ↪ " ", " +
  ↪ +
```

130

```
"\""
  ↪ descriptionFrench
  ↪ "\" : \" " +
  ↪ subAmbit.
  ↪ descriptionFrench
  ↪ " ", " +
  ↪ +
```

131

```
"\""
  ↪ descriptionBasque
  ↪ "\" : \" " +
  ↪ subAmbit.
  ↪ descriptionBasque
  ↪ " ", " +
  ↪ +
```

132

```
"\""
  ↪ descriptionCatalan
  ↪ "\" : \" " +
  ↪ subAmbit.
  ↪ descriptionCatalan
  ↪ " ", " +
  ↪ +
```

133

```
"\""
  ↪ descriptionDutch
  ↪ "\" : \" " +
```

```

    ↪ subAmbit.
    ↪ descriptionDutch
    ↪ + "\\", "
    ↪ +
    ↪ "\"
    ↪ descriptionGalician
    ↪ \" : \\" " +
    ↪ subAmbit.
    ↪ descriptionGalician
    ↪ + "\\", "
    ↪ +
    ↪ "\"
    ↪ descriptionGerman
    ↪ \" : \\" " +
    ↪ subAmbit.
    ↪ descriptionGerman
    ↪ + "\\", "
    ↪ +
    ↪ "\"
    ↪ descriptionItalian
    ↪ \" : \\" " +
    ↪ subAmbit.
    ↪ descriptionItalian
    ↪ + "\\", "
    ↪ +
    ↪ "\"
    ↪ descriptionPortuguese
    ↪ \" : \\" " +
    ↪ subAmbit.
    ↪ descriptionPortuguese
    ↪ + "\")";
138     allData.Add(JsonDocument.
    ↪ Parse(subAmb));
139 }
140
141     var subSubAmbits = _context.
    ↪ SubSubAmbits.ToList();
142     foreach (var subSubAmbit in
    ↪ subSubAmbits) {
143         String subSubAmb = "{\\"
    ↪ idSubSubAmb\" : \\" " +
    ↪ subSubAmbit.

```

```

    ↪ idSubSubAmbit + "\\", "
    ↪ +
144
    ↪ "\" idSubAmbit
    ↪ ":" \" " +
    ↪ subSubAmbit
    ↪ .
    ↪ idSubAmbit
    ↪ + "\\", "
    ↪ +
    ↪ "\" idAmbit "\" : \""
    ↪ " +
    ↪ subSubAmbit
    ↪ . idAmbit
    ↪ + "\\", " +
    ↪ "
145
    ↪ descriptionSpanish
    ↪ ":" \" " +
    ↪ subSubAmbit
    ↪ .
    ↪ descriptionSpanish
    ↪ + "\\", "
    ↪ +
    ↪ "
146
    ↪ descriptionEnglish
    ↪ ":" \" " +
    ↪ subSubAmbit
    ↪ .
    ↪ descriptionEnglish
    ↪ + "\\", "
    ↪ +
    ↪ "
147
    ↪ descriptionFrench
    ↪ ":" \" " +
    ↪ subSubAmbit
    ↪ .
    ↪ descriptionFrench
    ↪ + "\\", "
    ↪ +
    ↪ "
148
    ↪ descriptionBasque
    ↪ ":" \" " +
    ↪ subSubAmbit
    ↪ .

```

```

    ↳ descriptionBasque
    ↳ + "\\", "
    ↳ +
    " \
    ↳ descriptionCatalan
    ↳ \" :\\" " +
    ↳ subSubAmbit
    ↳ .
    ↳ descriptionCatalan
    ↳ + "\\", "
    ↳ +
    " \
    ↳ descriptionDutch
    ↳ \" :\\" " +
    ↳ subSubAmbit
    ↳ .
    ↳ descriptionDutch
    ↳ + "\\", "
    ↳ +
    " \
    ↳ descriptionGalician
    ↳ \" :\\" " +
    ↳ subSubAmbit
    ↳ .
    ↳ descriptionGalician
    ↳ + "\\", "
    ↳ +
    " \
    ↳ descriptionGerman
    ↳ \" :\\" " +
    ↳ subSubAmbit
    ↳ .
    ↳ descriptionGerman
    ↳ + "\\", "
    ↳ +
    " \
    ↳ descriptionItalian
    ↳ \" :\\" " +
    ↳ subSubAmbit
    ↳ .
    ↳ descriptionItalian
    ↳ + "\\", "
    ↳ +

```

150

151

152

153

154

```

155          " \\
156          ↪ descriptionPortuguese
157          ↪ \":\\\" + 
158          ↪ subSubAmbit
159          ↪ .
160          ↪ descriptionPortuguese
161          ↪ + "\\\"};"
162      allData.Add(JsonDocument.
163          ↪ Parse(subSubAmb));
164  }
165  String tams = "{\\\"numIndicators
166          ↪ \":\\\" + indicators.Count
167          ↪ + "\\\",";
168  if (evaluationType == "COMPLETE
169          ↪ ") {
170      tams += "\\\"numEvidences
171          ↪ \":\\\" + tamEvidences
172          ↪ + "\\\",";
173  }
174  tams+="\\\"numAmbits\\\":\\\" +
175          ↪ ambits.Count + "\\\","
176          ↪ "\\\"numSubAmbits\\\":\\\" +
177          ↪ subAmbits.Count
178          ↪ + "\\\","
179          ↪ "\\\"numSubSubAmbits\\\":\\\" +
180          ↪ " + subSubAmbits.
181          ↪ Count + "\\\"};"
182  allData.Add(JsonDocument.Parse(
183          ↪ tams));
184  return Ok(allData);
185 }
186 catch (Exception ex)
187 {
188     return BadRequest(ex.Message);
189 }
190 }
191
192     /// <summary>
193     /// Method that obtains all the
194     ↪ indicators of an ambit
195     /// </summary>
196     /// <param name="idAmbit">Ambit
197     ↪ identifier</param>

```

```

178     /// <returns>Indicators list</returns>
179     [HttpGet("ambit")]
180     [Authorize]
181     public IActionResult GetAllByType([
182         ↪ FromQuery] int idAmbit, [
183         ↪ FromHeader] string Authorization)
184     {
185         try
186         {
187             var indicators = _context.
188                 ↪ Indicators.Where(i => i.
189                 ↪ idAmbit == idAmbit && i.
190                 ↪ isActive == 1).ToList();
191             return Ok(indicators);
192         }
193         catch (Exception ex)
194         {
195             return BadRequest(ex.Message);
196         }
197     }

198     /// <summary>
199     /// Method that obtains an indicator
200     /// from the database
201     /// </summary>
202     /// <param name="idIndicator">Indicator
203     /// identifier</param>
204     /// <param name="indicatorType">
205     /// Indicator type</param>
206     /// <param name="idSubSubAmbit">Second
207     /// level division of the ambit</
208     /// param>
209     /// <param name="idSubAmbit">First
210     /// level division of the ambit</
211     /// param>
212     /// <param name="idAmbit">Ambit
213     /// identifier</param>
214     /// <param name="indicatorVersion">
215     /// Indicator version</param>
216     /// <param name="evaluationType">
217     /// Evaluation type</param>
218     /// <returns>Indicator if success, null
219     /// if not</returns>

```

```
205
206     [HttpGet("get")]
207     [Authorize]
208     public ActionResult<Indicator> Get([
209         ↪ FromQuery] int idIndicator, [
210         ↪ FromQuery] string indicatorType,
211         ↪ [FromQuery] int idSubSubAmbit, [
212         ↪ FromQuery] int idSubAmbit, [
213         ↪ FromQuery] int indicatorVersion,
214         ↪ [FromQuery] string evaluationType
215         ↪ , [FromHeader] string
216         ↪ Authorization)
217     {
218         ↪ try
219         {
220             var indicator = _context.
221                 ↪ Indicators.FirstOrDefault
222                 ↪ (i => i.idIndicator ==
223                 ↪ idIndicator && i.
224                 ↪ indicatorType ==
225                 ↪ indicatorType && i.
226                 ↪ idSubSubAmbit ==
227                 ↪ idSubSubAmbit && i.
228                 ↪ idSubAmbit == idSubAmbit
229                 ↪ && i.idAmbit == idAmbit
230                 ↪ && i.indicatorVersion ==
231                 ↪ indicatorVersion && i.
232                 ↪ evaluationType ==
233                 ↪ evaluationType);
234
235             if (indicator == null)
236                 return NotFound();
237
238             return indicator;
239         }
240         catch (Exception ex)
241         {
242             return BadRequest(ex.Message);
243         }
244     }
245 }
```

```

226
227     ///<summary>
228     /// Method that creates an indicator
229     ///</summary>
230     ///<param name="indicator">Indicator</
231     ↪ param>
232     ///<returns>Indicator if success, null
233     ↪ if not</returns>
234     [HttpPost]
235     [Authorize(Policy = "Administrator")]
236     public IActionResult Create([FromBody]
237     ↪ Indicator indicator, [FromHeader]
238     ↪ string Authorization)
239     {
240         try
241         {
242             _context.Indicators.Add(
243                 ↪ indicator);
244             _context.SaveChanges();
245             return CreatedAtAction(nameof(
246                 ↪ Get), new { id =
247                 ↪ indicator.idIndicator,
248                 ↪ type = indicator.
249                 ↪ indicatorType,
250                 ↪ idSubSubAmbit = indicator
251                 ↪ .idSubSubAmbit,
252                 ↪ idSubAmbit = indicator.
253                 ↪ idSubAmbit, idAmbit =
254                 ↪ indicator.idAmbit,
255                 ↪ version = indicator.
256                 ↪ indicatorVersion,
257                 ↪ evaluationType = indicator
258                 ↪ .evaluationType },
259                 ↪ indicator);
260         }
261         catch (Exception ex)
262         {
263             return BadRequest(ex.Message);
264         }
265     }
266
267     ///<summary>
268     /// Method that updates an indicator
269

```

```

250         ///</summary>
251         ///<param name="idIndicator">Indicator
252             ↪ identifier</param>
253         ///<param name="indicatorType">
254             ↪ Indicator type</param>
255         ///<param name="idSubSubAmbit">Second
256             ↪ level division of the ambit</
257             ↪ param>
258         ///<param name="idSubAmbit">First
259             ↪ level division of the ambit</
260             ↪ param>
261         ///<param name="idAmbit">Ambit
262             ↪ identifier</param>
263         ///<param name="indicatorVersion">
264             ↪ Indicator version</param>
265         ///<param name="evaluationType">
266             ↪ Evaluation type</param>
267         ///<param name="indicator">Indicator</
268             ↪ param>
269         ///<returns>Indicator if success, null
270             ↪ if not</returns>
271         [HttpPost]
272         [Authorize(Policy = "Administrator")]
273         public IActionResult Update([FromQuery]
274             ↪ int idIndicator, [FromQuery]
275             ↪ string indicatorType, [FromQuery]
276             ↪ int idSubSubAmbit, [FromQuery]
277             ↪ int idSubAmbit, [FromQuery] int
278             ↪ idAmbit, [FromQuery] int
279             ↪ indicatorVersion, [FromQuery]
280             ↪ string evaluationType, [FromBody]
281             ↪ Indicator indicator, [FromHeader
282             ↪ ] string Authorization)
283     {
284         try
285     {
286         if (idIndicator != indicator.
287             ↪ idIndicator ||
288             ↪ indicatorType !=
289             ↪ indicator.indicatorType
290             ||
291             ↪ indicatorVersion !=
292             ↪ indicator.
293             ↪ indicatorVersion - 1 ||

```

```

    ↵ idSubSubAmbit != indicator.idSubSubAmbit
    ↵ || idSubAmbit != indicator.idSubAmbit ||
    ↵ idAmbit != indicator.idAmbit || evaluationType
    ↵ != indicator.evaluationType)
    ↵ return BadRequest();
267
268 var existingIndicator =
    ↵ _context.Indicators.
    ↵ FirstOrDefault(i => i.
    ↵ idIndicator == idIndicator && i.
    ↵ indicatorType == indicatorType && i.
    ↵ idSubSubAmbit == idSubSubAmbit && i.
    ↵ idSubAmbit == idSubAmbit && i.idAmbit == idAmbit
    ↵ && i.indicatorVersion == indicatorVersion && i.
    ↵ evaluationType == evaluationType);
269 if (existingIndicator is null)
270     ↵ return NotFound();
271
272     ↵ _context.Indicators.Add(
    ↵ indicator);
273     ↵ existingIndicator.isActive = 0;
274     ↵ _context.SaveChanges();
275
276     ↵ return Ok(indicator);
277 }
278 catch (Exception ex)
279 {
280     ↵ return BadRequest(ex.Message);
281 }
282 }
283
284 /// <summary>
285 /// Method that deletes an indicator
286 /// </summary>

```

```

287         ///<param name="idIndicator">Indicator
288         ↪ identifier</param>
289         ///<param name="indicatorType">
290         ↪ Indicator type</param>
291         ///<param name="idSubSubAmbit">Second
292         ↪ level division of the ambit</
293         ↪ param>
294         ///<param name="idSubAmbit">First
295         ↪ level division of the ambit</
296         ↪ param>
297         ///<param name="idAmbit">Ambit
298         ↪ identifier</param>
299         ///<param name="indicatorVersion">
300         ↪ Indicator version</param>
301         ///<param name="evaluationType">
302         ↪ Evaluation type</param>
303         ///<returns>Indicator if success, null
304         ↪ if not</returns>
305         [HttpDelete]
306         [Authorize(Policy = "Administrator")]
307         public IActionResult Delete([FromQuery]
308             ↪ int idIndicator, [FromQuery]
309             ↪ string indicatorType, [FromQuery]
310             ↪ int idSubSubAmbit, [FromQuery]
311             ↪ int idSubAmbit, [FromQuery] int
312             ↪ idAmbit, [FromQuery] int
313             ↪ indicatorVersion, [FromQuery]
314             ↪ string evaluationType, [
315             ↪ FromHeader] string Authorization)
316         {
317             try
318             {
319                 var indicator = _context.
320                     ↪ Indicators.FirstOrDefault
321                     ↪ (i => i.idIndicator ==
322                     ↪ idIndicator && i.
323                     ↪ indicatorType == i.
324                     ↪ idSubSubAmbit ==
325                     ↪ idSubSubAmbit && i.
326                     ↪ idSubAmbit == idSubAmbit
327                     && i.idAmbit == idAmbit
328                     && i.indicatorVersion ==
329             }
330         }

```

```

    ↪ indicatorVersion && i.
    ↪ evaluationType ==
    ↪ evaluationType);

302         if (indicator is null)
303             return NotFound();
304
305         _context.Indicators.Remove(
306             ↪ indicator);
307         _context.SaveChanges();
308         return NoContent();
309     }
310     catch (Exception ex)
311     {
312         return BadRequest(ex.Message);
313     }
314 }
315 }
316 }
```

Como se puede comprobar en este controlador, todos los métodos vienen acompañados por `HttpGet`, `HttpPost`, `HttpPut` o `HttpDelete`, acompañados del endpoint correspondiente establecido. En el caso del `HttpPut` y del `HttpPost`, siempre viene acompañado su parámetro por un `FromBody`, el cual indica que el objeto procede de un formato JSON. También cabe resaltar que antes de colocar el `public class`, se coloca con anterioridad `ApiController` y posteriormente `Route("Indicators")`, que establecen el endpoint base de la API.

- En el lado del cliente también tenemos tres tipos de clase diferentes, pero llamadas de diferente forma:
 - **Callers:** Son los métodos que se encargan de realizar de forma asíncrona las peticiones del servidor y a su vez procesar la respuesta del cliente, todo ello de forma asíncrona mediante programación multihilo, ya sea mediante `Future` o mediante el uso de `Callback`. Dichos Callers sigue el patrón de diseño Singleton, ya que no es necesario contar con múltiples instancias de la clase, pudiendo crear una sola para cada uno de estos. Cuando se tienen múltiples instancias repartidas en el tiempo de ejecución, los tiempos de respuesta son muy lentos, por lo que utilizar este

patrón de diseño es muy importante para poder acelerar dicho proceso. Ejemplo:

```

1          /**
2   * Class controller for indicators operations
3   *
4   * @author Pablo Ahita del Barrio
5   * @version 1
6   * */
7 public class IndicatorsController {
8
9     /**Indicators api to connect to the class*/
10    private static IndicatorsApi api;
11
12    /**Controller instance*/
13    private static IndicatorsController
14        ↪ instance;
15
16    /**Number of attempts*/
17    private static int numAttempts=0;
18
19    /**Class controller*/
20    private IndicatorsController(){
21        api=ConnectionClient.getInstance().
22            ↪ getRetrofit().create(
23            ↪ IndicatorsApi.class);
24    }
25
26    /**
27     * Method that obtains the singleton
28     * ↪ instance of the controller
29     *
30     * @return Controller instance
31     * */
32    public static IndicatorsController
33        ↪ getInstance(){
34        if(instance==null){
35            synchronized (IndicatorsController.
36                ↪ class){
37                if(instance==null){
38                    instance=new
39                        ↪ IndicatorsController
40                        ↪ ();
41                }
42            }
43        }
44    }

```

```

34             }
35         }
36         return instance;
37     }
38
39     /**Refresh API*/
40     public static void refreshApi(){
41         instance=new IndicatorsController();
42     }
43
44     /**
45      * Method that obtains all the indicators
46      * ↪ of an ambit
47      *
48      * @param idAmbit - Ambit identifier
49      * @return Indicators list
50      */
51     public static List<Indicator>
52         ↪ GetAllByIdAmbit(int idAmbit){
53         ExecutorService executor = Executors.
54             ↪ newSingleThreadExecutor();
55         Callable<List<Indicator>> callable =
56             ↪ new Callable<List<Indicator>>() {
57             @Override
58             public List<Indicator> call()
59                 ↪ throws Exception {
60                 Call<List<Indicator>> call =
61                     ↪ api.GetAllByIdAmbit(
62                     ↪ idAmbit, Session.
63                     ↪ getInstance().getToken()
64                     ↪ ;
65                 Response<List<Indicator>>
66                     ↪ response = call.execute()
67                     ↪ ;
68                 if (response.isSuccessful()) {
69                     return response.body();
70                 } else {
71                     throw new IOException(""
72                         ↪ Error: " + response.
73                         ↪ code() + " " +
74                         ↪ response.message());
75                 }
76             }

```

```

63         }
64     };
65     try {
66         Future<List<Indicator>> future =
67             ↳ executor.submit(callable);
68         List<Indicator> list = future.get()
69             ↳ ;
70         executor.shutdown();
71         return list;
72     } catch (InterruptedException |
73         ↳ ExecutionException e) {
74         throw new RuntimeException(e);
75     }
76 }
77 /**
78 * Method that updates an indicator
79 *
80 * @param idIndicator - Indicator
81 *   ↳ identifier
82 * @param indicatorType - Indicator type
83 * @param idSubSubAmbit - Second level
84 *   ↳ division of the ambit
85 * @param idSubAmbit - First level
86 *   ↳ division of the ambit
87 * @param idAmbit - Ambit identifier
88 * @param indicatorVersion - Indicator
89 *   ↳ version
90 * @param indicator - Indicator
91 * @return Updated indicator if success,
92 *   ↳ null if not
93 */
94 public static Indicator Update(int
95     ↳ idIndicator, String indicatorType,
96     ↳ int idSubSubAmbit, int idSubAmbit,
97     ↳ int idAmbit, int indicatorVersion,
98     ↳ String evaluationType, Indicator
99     ↳ indicator){
100     ExecutorService executor = Executors.
101         ↳ newSingleThreadExecutor();
102     Callable<Indicator> callable = new
103         ↳ Callable<Indicator>() {

```

```

91     @Override
92     public Indicator call() throws
93         ↪ Exception {
94         Call<Indicator> call = api.
95             ↪ Update(idIndicator,
96             ↪ indicatorType,
97             ↪ idSubSubAmbit, idSubAmbit,
98             ↪ idAmbit, indicatorVersion,
99             ↪ evaluationType, indicator,
100            ↪ Session.getInstance().
101            ↪ getToken());
102
103        Response<Indicator> response =
104            ↪ call.execute();
105        if (response.isSuccessful()) {
106            ↪ return response.body();
107        } else {
108            ↪ throw new IOException(
109                ↪ "Error: " + response.
110                ↪ code() + " " +
111                ↪ response.message());
112        }
113    };
114
115    try {
116        Future<Indicator> future = executor
117            ↪ .submit(callable);
118        Indicator result = future.get();
119        executor.shutdown();
120        return result;
121    } catch (InterruptedException |
122        ↪ ExecutionException e) {
123        ↪ throw new RuntimeException(e);
124    }
125
126    /**
127     * Method that creates an indicator
128     *
129     * @param indicator - Indicator
130     * @return Indicator if success, null if
131         ↪ not
132     * */
133

```

```

118     public static Indicator Create(Indicator
119         ↪ indicator){
120         ExecutorService executor = Executors.
121             ↪ newSingleThreadExecutor();
122         Callable<Indicator> callable = new
123             ↪ Callable<Indicator>() {
124                 @Override
125                 public Indicator call() throws
126                     ↪ Exception {
127                     Call<Indicator> call = api.
128                         ↪ Create(indicator, Session.
129                             ↪ getInstance().getToken())
130                         ↪ ;
131                     Response<Indicator> response =
132                         ↪ call.execute();
133                     if (response.isSuccessful()) {
134                         return response.body();
135                     } else {
136                         throw new IOException(""
137                             ↪ Error: " + response.
138                             ↪ code() + " " +
139                             ↪ response.message());
140                     }
141                 }
142             };
143             try {
144                 Future<Indicator> future = executor
145                     ↪ .submit(callable);
146                 Indicator result = future.get();
147                 executor.shutdown();
148                 return result;
149             } catch (InterruptedException |
150                 ↪ ExecutionException e) {
151                 throw new RuntimeException(e);
152             }
153         }
154     /**
155      * Method that obtains an indicator from
156      * ↪ the database
157      *
158      * @param idIndicator - Indicator
159      * ↪ identifier

```

```

146     * @param indicatorType - Indicator type
147     * @param idSubSubAmbit - Second level
148         ↪ division of the ambit
149     * @param idSubAmbit - First level
150         ↪ division of the ambit
151     * @param idAmbit - Ambit identifier
152     * @param indicatorVersion - Indicator
153         ↪ version
154     * @return Indicator if success, null if
155         ↪ not
156     */
157     public static Indicator Get(int idIndicator
158         ↪ , String indicatorType, int
159         ↪ idSubSubAmbit, int idSubAmbit, int
160         ↪ idAmbit, int indicatorVersion, String
161         ↪ evaluationType){
162         ExecutorService executor = Executors.
163             ↪ newSingleThreadExecutor();
164         Callable<Indicator> callable = new
165             ↪ Callable<Indicator>() {
166                 @Override
167                 public Indicator call() throws
168                     ↪ Exception {
169                     Call<Indicator> call = api.Get(
170                         ↪ idIndicator, indicatorType
171                         ↪ , idSubSubAmbit, idSubAmbit
172                         ↪ , idAmbit, indicatorVersion
173                         ↪ , evaluationType, Session.
174                         ↪ getInstance().getToken()
175                         ↪ ;
176                     Response<Indicator> response =
177                         ↪ call.execute();
178                     if (response.isSuccessful()) {
179                         ↪ return response.body();
180                     } else {
181                         ↪ throw new IOException(""
182                             ↪ Error: " + response.
183                             ↪ code() + " " +
184                             ↪ response.message());
185                     }
186                 }
187             };
188         try {

```

```

168         Future<Indicator> future = executor
169             ↪ .submit(callable);
170         Indicator result = future.get();
171         executor.shutdown();
172         return result;
173     } catch (InterruptedException |
174             ↪ ExecutionException e) {
175         throw new RuntimeException(e);
176     }
177 }
178 /**
179 * Method that obtains all the indicators
180 *
181 * @return Indicators list
182 */
183 public static List<JsonObject> GetAll(
184     ↪ String evaluationType){
185     ExecutorService executor = Executors.
186         ↪ newSingleThreadExecutor();
187     Callable<List<JsonObject>> callable =
188         ↪ new Callable<List<JsonObject>>()
189         ↪ {
190         @Override
191         public List<JsonObject> call()
192             ↪ throws Exception {
193             Call<List<JsonObject>> call =
194                 ↪ api.GetAll(evaluationType
195                 ↪ ,Session.getInstance().
196                 ↪ getToken());
197             Response<List<JsonObject>>
198                 ↪ response = call.execute()
199                 ↪ ;
200             if (response.isSuccessful()) {
201                 return response.body();
202             } else {
203                 throw new IOException(""
204                     ↪ Error: " + response.
205                     ↪ code() + " " +
206                     ↪ response.message());
207             }
208         }
209     };
210 }
```

```

196     try {
197         Future<List<JsonObject>> future =
198             ↪ executor.submit(callable);
199         List<JsonObject> list = future.get
200             ↪ ();
201         executor.shutdown();
202         numAttempts=0;
203         return list;
204     } catch (InterruptedException |
205         ↪ ExecutionException e) {
206         if(e.getCause() instanceof
207             ↪ SocketTimeoutException){
208             numAttempts++;
209             if(numAttempts<3) {
210                 return GetAll(
211                     ↪ evaluationType);
212             }
213             else{
214                 numAttempts=0;
215                 return null;
216             }
217         }
218     }
219     /**
220      * Method that deletes an indicator
221      *
222      * @param idIndicator - Indicator
223      *          ↪ identifier
224      * @param indicatorType - Indicator type
225      * @param idSubSubAmbit - Second level
226      *          ↪ division of the ambit
227      * @param idSubAmbit - First level
228      *          ↪ division of the ambit
229      * @param idAmbit - Ambit identifier
230      * @param indicatorVersion - Indicator
231      *          ↪ version
232      * @return Indicator if success, null if
233      *          ↪ not

```

```
229     * */
230     public static Indicator Delete(int
231         ↪ idIndicator, String indicatorType,
232         ↪ int idSubSubAmbit, int idSubAmbit,
233         ↪ int idAmbit, int indicatorVersion,
234         ↪ String evaluationType){
235         ExecutorService executor = Executors.
236             ↪ newSingleThreadExecutor();
237         Callable<Indicator> callable = new
238             ↪ Callable<Indicator>() {
239                 @Override
240                 public Indicator call() throws
241                     ↪ Exception {
242                     Call<Indicator> call = api.Get(
243                         ↪ idIndicator, indicatorType
244                         ↪ , idSubSubAmbit, idSubAmbit
245                         ↪ , idAmbit, indicatorVersion
246                         ↪ , evaluationType, Session.
247                         ↪ getInstance().getToken()
248                         ↪ ;
249                     Response<Indicator> response =
250                         ↪ call.execute();
251                     if (response.isSuccessful()) {
252                         return response.body();
253                     } else {
254                         throw new IOException(
255                             ↪ "Error: " + response.
256                             ↪ code() + " " +
257                             ↪ response.message());
258                     }
259                 }
260             };
261         try {
262             Future<Indicator> future = executor
263                 ↪ .submit(callable);
264             Indicator result = future.get();
265             executor.shutdown();
266             return result;
267         } catch (InterruptedException |
268             ↪ ExecutionException e) {
269             throw new RuntimeException(e);
270         }
271     }
```

253 }

Como se puede comprobar, todos los métodos utilizan en primer lugar **Future** para poder comunicarse con la API, que contiene el endpoint referenciado en el servidor, cuya URL base se encuentra en la clase **ConnectionClient**, la cual utiliza también el patrón Singleton por el mismo motivo que los callers. **Future**, mediante el callable del ejecutor, se encarga de llamar al método de la API para luego procesar la respuesta. El método siempre devuelve **future.get()**, el cual devuelve la respuesta de forma bloqueante. Como punto adicional, el tiempo de respuesta es más rápido en endpoints que devuelven listas si se devuelven como **JsonObject** debido a que se trabaja con JSON puro, el cual no requiere ser serializado o deserializado en el proceso de comunicación cliente-servidor.

- **Api:** Las API en Java están definidas como interfaces. Las llamadas a los endpoint se realizan mediante *Retrofit*, como se ha mencionado en la memoria. Ejemplo:

```

1      /**
2       * API for indicators operations
3       *
4       * @author Pablo Ahita del Barrio
5       * @version 1
6       */
7       public interface IndicatorsApi {
8
9           /**
10            * Updates an indicator
11            *
12            * @param idIndicator -
13            *         ↗ Indicator identifier
14            * @param indicatorType -
15            *         ↗ Indicator type
16            * @param idSubSubAmbit - Second
17            *         ↗ level division of the
18            *         ↗ ambit
19            * @param idSubAmbit - First
20            *         ↗ level division of the
21            *         ↗ ambit
22            * @param idAmbit - Ambit
23            *         ↗ identifier
24        }
25    }

```

```

17          * @param indicatorVersion -
18          *   ↪ Indicator version
19          * @param indicator - Indicator
20          * */
21          @PUT("Indicators")
22          Call<Indicator> Update(@Query(
23              ↪ idIndicator) int
24              ↪ idIndicator, @Query(
25                  ↪ indicatorType) String
26                  ↪ indicatorType, @Query(
27                      ↪ idSubSubAmbit) int
28                      ↪ idSubSubAmbit, @Query(
29                          ↪ idSubAmbit) int
30                          ↪ idSubAmbit, @Query(
31                              ↪ idAmbit) int idAmbit,
32                              ↪ @Query("indicatorVersion")
33                              ↪ int indicatorVersion,
34                              ↪ @Query("evaluationType")
35                              ↪ String evaluationType,
36                              ↪ @Body Indicator indicator,
37                              ↪ @Header("Authorization")
38                              ↪ String Authorization);
39
40          /**
41          * Creates an indicator
42          *
43          * @param indicator - Indicator
44          * */
45          @POST("Indicators")
46          Call<Indicator> Create(@Body
47              ↪ Indicator indicator,
48              ↪ @Header("Authorization")
49              ↪ String Authorization);
50
51          /**
52          * Gets an indicator
53          *
54          * @param idIndicator -
55          *   ↪ Indicator identifier
56          * @param indicatorType -
57          *   ↪ Indicator type

```

```

36             * @param idSubSubAmbit - Second
37             *          ↪ level division of the
38             *          ↪ ambit
39             * @param idSubAmbit - First
40             *          ↪ level division of the
41             *          ↪ ambit
42             * @param idAmbit - Ambit
43             *          ↪ identifier
44             * @param indicatorVersion -
45             *          ↪ Indicator version
46             * */
47
48             @GET("Indicators/get")
49             Call<Indicator> Get(@Query(
50                 ↪ "idIndicator") int
51                 ↪ idIndicator, @Query(
52                     ↪ "indicatorType") String
53                     ↪ indicatorType, @Query(
54                         ↪ "idSubSubAmbit") int
55                         ↪ idSubSubAmbit, @Query(
56                             ↪ "idSubAmbit") int
57                             ↪ idSubAmbit, @Query(
58                                 ↪ "idAmbit") int idAmbit,
59                                 ↪ @Query("indicatorVersion")
60                                 ↪ int indicatorVersion,
61                                 ↪ @Query("evaluationType")
62                                 ↪ String evaluationType,
63                                 ↪ @Header("Authorization")
64                                 ↪ String Authorization);
65
66             /**
67             * Gets all indicators*/
68             @GET("Indicators/all")
69             Call<List<JsonObject>> GetAll(
70                 ↪ @Query("evaluationType")
71                 ↪ String evaluationType,
72                 ↪ @Header("Authorization")
73                 ↪ String Authorization);
74
75             /**
76             * Gets all ambits by ambit
77             *          ↪ identifier
78             *
79             * @param idAmbit - Ambit
80             *          ↪ identifier

```

```

52          * */
53      @GET("Indicators/ambit")
54      Call<List<Indicator>>
55          ↳ GetAllByIdAmbit(@Query(
56              ↳ idAmbit") int idAmbit,
57              ↳ @Header("Authorization")
58              ↳ String Authorization);
59
60      /**
61      * Deletes an indicator
62      *
63      * @param idIndicator -
64      *     ↳ Indicator identifier
65      * @param indicatorType -
66      *     ↳ Indicator type
67      * @param idSubSubAmbit - Second
68      *     ↳ level division of the
69      *     ↳ ambit
70      * @param idSubAmbit - First
71      *     ↳ level division of the
72      *     ↳ ambit
73      * @param idAmbit - Ambit
74      *     ↳ identifier
75      * @param indicatorVersion -
76      *     ↳ Indicator version
77      * */
78
79      @DELETE("Indicators")
80      Call<Indicator> Delete(@Query(
81          ↳ idIndicator") int
82          ↳ idIndicator, @Query(
83              ↳ indicatorType") String
84              ↳ indicatorType, @Query(
85              ↳ idSubSubAmbit") int
86              ↳ idSubSubAmbit, @Query(
87              ↳ idSubAmbit") int
88              ↳ idSubAmbit, @Query(
89              ↳ idAmbit") int idAmbit,
90              ↳ @Query("indicatorVersion")
91              ↳ int indicatorVersion,
92              ↳ @Query("evaluationType")
93              ↳ String evaluationType,
94              ↳ @Header("Authorization")
95              ↳ String Authorization);

```

```
68 }
```

Como se puede apreciar, todos los métodos vienen con la notación GET, POST, PUT o DELETE acompañada de su endpoint correspondiente. Si se desean añadir datos a los endpoint se tiene que utilizar Path, mientras que para los POST se tiene que utilizar BODY para la serialización en JSON

- **Modelos:** Se trata de las propias clases de Java. Ejemplo:

```

1      /**
2      * Model class for the indicators
3      *
4      * @author Pablo Ahita del Barrio
5      * @version 1
6      */
7
8  public class Indicator implements Serializable
9      {
10
11      /**Indicator's identifier*/
12      @SerializedName("idIndicator")
13      public int idIndicator;
14
15      /**Indicator's type*/
16      @SerializedName("indicatorType")
17      public String indicatorType;
18
19      /**Indicator's description in English*/
20      @SerializedName("descriptionEnglish")
21      public String descriptionEnglish;
22
23      /**Indicator's description in Spanish*/
24      @SerializedName("descriptionSpanish")
25      public String descriptionSpanish;
26
27      /**Indicator's description in French*/
28      @SerializedName("descriptionFrench")
29      public String descriptionFrench;
30
31      /**Indicator's description in Basque*/
32      @SerializedName("descriptionBasque")
33      public String descriptionBasque;
34
35      /**Indicator's description in Catalan*/

```

```
35     @SerializedName("descriptionCatalan")
36     public String descriptionCatalan;
37
38     /**Indicator's description in Dutch*/
39     @SerializedName("descriptionDutch")
40     public String descriptionDutch;
41
42     /**Indicator's description in Galician*/
43     @SerializedName("descriptionGalician")
44     public String descriptionGalician;
45
46     /**Indicator's description in German*/
47     @SerializedName("descriptionGerman")
48     public String descriptionGerman;
49
50     /**Indicator's description in Italian*/
51     @SerializedName("descriptionItalian")
52     public String descriptionItalian;
53
54     /**Indicator's description in Portuguese*/
55     @SerializedName("descriptionPortuguese")
56     public String descriptionPortuguese;
57
58     /**Evidence list of the indicator*/
59     public List<Evidence> evidences;
60
61     /**Second level division of the ambit. It
62      ↪ will be -1 if there is no division*/
63     @SerializedName("idSubSubAmbit")
64     public int idSubSubAmbit;
65
66     /**First level division of the ambit. It
67      ↪ will be -1 if there is no division*/
68     @SerializedName("idSubAmbit")
69     public int idSubAmbit;
70
71     /**Ambit identifier*/
72     @SerializedName("idAmbit")
73     public int idAmbit;
74
75     /**Indicator priority*/
76     @SerializedName("indicatorPriority")
77     public String indicatorPriority;
```

```
76
77     /**Indicator version*/
78     @SerializedName("indicatorVersion")
79     public int indicatorVersion;
80
81     /**Boolean that determines if the indicator
82      ↪ is or not is activated. 1 if is, 0
83      ↪ if not*/
84     @SerializedName("isActive")
85     public int isActive;
86
87     /**Evaluation type*/
88     @SerializedName("evaluationType")
89     public String evaluationType;
90
91     /**
92      * Class constructor
93      *
94      * @param idIndicator - Indicator
95      ↪ identifier
96      * @param indicatorType - Indicator type
97      * @param idSubSubAmbit - Second level
98      ↪ division of the ambit. It will be -1
99      ↪ if there is no division
100     * @param idSubAmbit - First level division
101     ↪ of the ambit. It will be -1 if
102     ↪ there is no division
103     * @param idAmbit - Ambit identifier
104     * @param descriptionEnglish - Indicator's
105     ↪ description in English
106     * @param descriptionSpanish - Indicator's
107     ↪ description in Spanish
108     * @param descriptionFrench - Indicator's
109     ↪ description in French
110     * @param descriptionBasque - Indicator's
111     ↪ description in Basque
112     * @param descriptionCatalan - Indicator's
113     ↪ description in Catalan
114     * @param descriptionDutch - Indicator's
115     ↪ description in Dutch
```

```

105      * @param descriptionGalician - Indicator's
106          ↪ description in Galician
107      * @param descriptionGerman - Indicator's
108          ↪ description in German
109      * @param descriptionItalian - Indicator's
110          ↪ description in Italian
111      * @param descriptionPortuguese - Indicator
112          ↪ 's description in Portuguese
113      * @param indicatorPriority - Indicator
114          ↪ priority
115      * @param indicatorVersion - Indicator
116          ↪ version
117      * @param isActive - Boolean that
118          ↪ determines if the indicator is or
119          ↪ not activated. 1 if is, 0 if not
120      * @param evaluationType - Evaluation type
121      */
122
123  public Indicator(int idIndicator, String
124                  ↪ indicatorType, int idSubSubAmbit, int
125                  ↪ idSubAmbit, int idAmbit, String
126                  ↪ descriptionEnglish, String
127                  ↪ descriptionSpanish, String
128                  ↪ descriptionFrench, String
129                  ↪ descriptionBasque, String
130                  ↪ descriptionCatalan, String
131                  ↪ descriptionDutch, String
132                  ↪ descriptionGalician, String
133                  ↪ descriptionGerman, String
134                  ↪ descriptionItalian, String
135                  ↪ descriptionPortuguese, String
136                  ↪ indicatorPriority, int
137                  ↪ indicatorVersion, int isActive,
138                  ↪ String evaluationType) {
139      setIdIndicator(idIndicator);
140      setIndicatorType(indicatorType);
141      setIdSubSubAmbit(idSubSubAmbit);
142      setIdSubAmbit(idSubAmbit);
143      setIndicatorType(indicatorType);
144      setDescriptionEnglish(
145          ↪ descriptionEnglish);
146      setDescriptionSpanish(
147          ↪ descriptionSpanish);

```

```

123         setDescriptionFrench(descriptionFrench)
124             ↪ ;
125         setDescriptionBasque(descriptionBasque)
126             ↪ ;
127         setDescriptionCatalan(
128             ↪ descriptionCatalan);
129         setDescriptionDutch(descriptionDutch);
130         setDescriptionGalician(
131             ↪ descriptionGalician);
132         setDescriptionGerman(descriptionGerman)
133             ↪ ;
134         setDescriptionItalian(
135             ↪ descriptionItalian);
136         setDescriptionPortuguese(
137             ↪ descriptionPortuguese);
138         setIndicatorPriority(indicatorPriority)
139             ↪ ;
140         setIndicatorVersion(indicatorVersion);
141         setIsActive(isActive);
142         setEvaluationType(evaluationType);
143     }
144
145 /**
146 * Method that obtains the indicator
147 * ↪ identifier
148 *
149 * @return Indicator identifier
150 */
151 public int getIdIndicator() {
152     return idIndicator;
153 }
154
155 /**
156 * Method that sets the new indicator
157 * ↪ identifier
158 *
159 * @param idIndicator - Indicator
160 * ↪ identifier
161 */
162 public void setIdIndicator(int idIndicator)
163     ↪ {
164     this.idIndicator = idIndicator;
165 }
```

```
154
155     /**
156      * Method that obtains the indicator type
157      *
158      * @return Indicator type
159      * */
160
161     public String getIndicatorType() {
162         return indicatorType;
163     }
164
165     /**
166      * Method that sets the new indicator type
167      *
168      * @param indicatorType - Indicator type
169      * */
170     public void setIndicatorType(String
171         ↪ indicatorType) {
172         this.indicatorType = indicatorType;
173     }
174
175     /**
176      * Method that sets the evidences of an
177      * ↪ indicator
178      *
179      * @param evidences - Evidences list
180      * */
181     public void setEvidences(List<Evidence>
182         ↪ evidences) {
183         this.evidences=evidences;
184     }
185
186     /**
187      * Method that obtains the evidences of an
188      * ↪ indicator
189      *
190      * @return Evidences of an indicator
191      * */
192     public List<Evidence> getEvidences() {
193         return evidences;
194     }
195
196     /**
```

```
193     * Method that obtains the indicator
194     *         ↪ description in English
195     *
196     * @return Indicator's description in
197     *         ↪ English
198     * */
199     public String getDescriptionEnglish() {
200         return descriptionEnglish;
201     }
202
203     /**
204     * Method that sets the new description in
205     *         ↪ English
206     *
207     * @param descriptionEnglish - Indicator's
208     *         ↪ description in English
209     * */
210     public void setDescriptionEnglish(String
211         ↪ descriptionEnglish) {
212         this.descriptionEnglish =
213             ↪ descriptionEnglish;
214     }
215
216     /**
217     * Method that obtains the indicator
218     *         ↪ description in Spanish
219     *
220     * @return Indicator's description in
221     *         ↪ Spanish
222     * */
223     public String getDescriptionSpanish() {
224         return descriptionSpanish;
225     }
226
227     /**
228     * Method that sets the new description in
229     *         ↪ Spanish
230     *
231     * @param descriptionSpanish - Indicator's
232     *         ↪ description in Spanish
233     * */
234     public void setDescriptionSpanish(String
235         ↪ descriptionSpanish) {
```

```
225         this.descriptionSpanish =
226             ↪ descriptionSpanish;
227     }
228
229     /**
230      * Method that obtains the indicator
231      ↪ description in French
232      *
233      * @return Indicator's description in
234      ↪ French
235      */
236
237     /**
238      * Method that sets the new description in
239      ↪ French
240      *
241      * @param descriptionFrench - Indicator's
242      ↪ description in French
243      */
244
245     /**
246      * Method that obtains the indicator
247      ↪ description in Basque
248      *
249      * @return Indicator's description in
250      ↪ Basque
251      */
252
253     /**
254      * Method that sets the new description in
255      ↪ Basque
256      *
257      */
```

```
258     * @param descriptionBasque - Indicator's
259     *      ↪ description in Basque
260     * */
261     public void setDescriptionBasque(String
262         ↪ descriptionBasque) {
263         this.descriptionBasque =
264             ↪ descriptionBasque;
265     }
266
267     /**
268      * Method that obtains the indicator
269      *      ↪ description in Catalan
270      *
271      * @return Indicator's description in
272          ↪ Catalan
273      * */
274     public String getDescriptionCatalan() {
275         return descriptionCatalan;
276     }
277
278     /**
279      * Method that sets the new description in
280      *      ↪ Catalan
281      *
282      * @param descriptionCatalan - Indicator's
283      *      ↪ description in Catalan
284      * */
285     public void setDescriptionCatalan(String
286         ↪ descriptionCatalan) {
287         this.descriptionCatalan =
288             ↪ descriptionCatalan;
289     }
290
291     /**
292      * Method that obtains the indicator
293      *      ↪ description in Dutch
294      *
295      * @return Indicator's description in Dutch
296      * */
297     public String getDescriptionDutch() {
298         return descriptionDutch;
299     }
300
```

```
291     /**
292      * Method that sets the new description in
293      * ↪ Dutch
294      *
295      * @param descriptionDutch - Indicator's
296      * ↪ description in Dutch
297      * */
298     public void setDescriptionDutch(String
299         ↪ descriptionDutch) {
300         this.descriptionDutch =
301             ↪ descriptionDutch;
302     }
303
304     /**
305      * Method that obtains the indicator
306      * ↪ description in Galician
307      *
308      * @return Indicator's description in
309      * ↪ Galician
310      * */
311     public String getDescriptionGalician() {
312         return descriptionGalician;
313     }
314
315     /**
316      * Method that sets the new description in
317      * ↪ Galician
318      *
319      * @param descriptionGalician - Indicator's
320      * ↪ description in Galician
321      * */
322     public void setDescriptionGalician(String
323         ↪ descriptionGalician) {
324         this.descriptionGalician =
325             ↪ descriptionGalician;
326     }
327
328     /**
329      * Method that obtains the indicator
330      * ↪ description in German
331      *
332      * @return Indicator's description in
333      * ↪ German
```

```
322     * */
323     public String getDescriptionGerman() {
324         return descriptionGerman;
325     }
326
327     /**
328      * Method that sets the new description in
329      * ↗ German
330      *
331      * @param descriptionGerman - Indicator's
332      * ↗ description in German
333      * */
334     public void setDescriptionGerman(String
335         ↗ descriptionGerman) {
336         this.descriptionGerman =
337             ↗ descriptionGerman;
338     }
339
340     /**
341      * Method that obtains the indicator
342      * ↗ description in Italian
343      *
344      * @return Indicator's description in
345      * ↗ Italian
346      * */
347     public String getDescriptionItalian() {
348         return descriptionItalian;
349     }
350
351     /**
352      * Method that sets the new description in
353      * ↗ Italian
354      *
355      * @param descriptionItalian - Indicator's
356      * ↗ description in Italian
357      * */
358     public void setDescriptionItalian(String
359         ↗ descriptionItalian) {
360         this.descriptionItalian =
361             ↗ descriptionItalian;
362     }
363
364     /**
365      * */
366 
```

```
355     * Method that obtains the indicator
356     *      ↪ description in Portuguese
357     *
358     * @return Indicator's description in
359     *      ↪ Portuguese
360     * */
361     public String getDescriptionPortuguese() {
362         return descriptionPortuguese;
363     }
364
365     /**
366     * Method that sets the new description in
367     *      ↪ Portuguese
368     *
369     * @param descriptionPortuguese - Indicator
370     *      ↪ 's description in Portuguese
371     * */
372     public void setDescriptionPortuguese(String
373     *      ↪ descriptionPortuguese) {
374         this.descriptionPortuguese =
375             ↪ descriptionPortuguese;
376     }
377
378     /**
379     * Method that obtains the indicator
380     *      ↪ version
381     *
382     * @return Indicator version
383     * */
384     public int getIndicatorVersion() {
385         return indicatorVersion;
386     }
387
388     /**
389     * Method that sets the new indicator
390     *      ↪ version
391     *
392     * @param indicatorVersion - Indicator
393     *      ↪ version
394     * */
395     public void setIndicatorVersion(int
396     *      ↪ indicatorVersion) {
```

```
388         this.indicatorVersion =
389             ↪ indicatorVersion;
390     }
391
392     /**
393      * Method that obtains the ambit identifier
394      *
395      * @return Ambit identifier
396      */
397     public int getIdAmbit() {
398         return idAmbit;
399     }
400
401     /**
402      * Method that sets the new ambit
403      * identifier
404      *
405      * @param idAmbit - Ambit identifier
406      */
407     public void setIdAmbit(int idAmbit) {
408         this.idAmbit = idAmbit;
409     }
410
411     /**
412      * Method that obtains the second level
413      * division of the ambit
414      *
415      * @return Second level division of the
416      * ambit
417      */
418     public int getIdSubSubAmbit() {
419         return idSubSubAmbit;
420     }
421
422     /**
423      * Method that sets the new second level
424      * division of the ambit
425      *
426      * @param idSubSubAmbit - Second level
427      * division of the ambit
428      */
429     public void setIdSubSubAmbit(int
430             ↪ idSubSubAmbit) {
```

```
424         this.idSubSubAmbit = idSubSubAmbit;
425     }
426
427     /**
428      * Method that obtains the first level
429      * ↪ division of the ambit
430      *
431      * @return First level division of the
432      * ↪ ambit
433      * */
434     public int getIdSubAmbit() {
435         return idSubAmbit;
436     }
437
438     /**
439      * Method that sets the new first level
440      * ↪ division of the ambit
441      *
442      * @param idSubAmbit - First level division
443      * ↪ of the ambit
444      * */
445     public void setIdSubAmbit(int idSubAmbit) {
446         this.idSubAmbit = idSubAmbit;
447     }
448
449     /**
450      * Method that obtains the new indicator
451      * ↪ priority
452      *
453      * @return Indicator priority
454      * */
455     public String getIndicatorPriority() {
456         return indicatorPriority;
457     }
458
459     /**
460      * Method that sets the new indicator
461      * ↪ identifier
462      *
463      * @param indicatorPriority - Indicator
464      * ↪ priority
465      * */
466 }
```

```
459     public void setIndicatorPriority(String
460         ↪ indicatorPriority) {
461         this.indicatorPriority =
462             ↪ indicatorPriority;
463     }
464
465     /**
466      * Method that obtains if the indicator is
467      * ↪ active
468      *
469      * @return 1 if is activated, 0 if not
470      * */
471     public int getIsActive() {
472         return isActive;
473     }
474
475     /**
476      * Method that sets if the indicator is
477      * ↪ active
478      *
479      * @param isActive - 1 if is activated, 0
480      * ↪ if not
481      * */
482     public void setIsActive(int isActive) {
483         this.isActive = isActive;
484     }
485
486     /**
487      * Method that obtains the evaluation type
488      *
489      * @return Evaluation type
490      * */
491
492     /**
493      * Method that sets the new evaluation type
494      *
495      * @param evaluationType - Evaluation type
496      * */
```

```
497     public void setEvaluationType(String
498         ↪ evaluationType) {
499     this.evaluationType = evaluationType;
500 }
```

Como se puede apreciar, todos los campos de la clase vienen acompañados por `SerializedName` acompañado del nombre que recibe el cliente de los JSON. El modelo implementa serializable para poder pasar los objetos entre las propias actividades de la aplicación.

Apéndice D

Documentación técnica de programación

D.1. Introducción

D.2. Estructura de directorios

La estructura de directorios del proyecto consta de las siguientes carpetas (teniendo en cuenta que el directorio raíz es *tfg2223*):

- *app/build/outputs/apk/debug*: Lugar donde se ubica la apk en el directorio.
- *app/main/java/gui/adapters*: Directorio que alberga las vistas adaptativas de los spinner o desplegables adaptados a cada una de las clases que lo requieren.
- *app/main/java/gui/data*: Clases auxiliares que ayudan a manejar la actividad de inicio de sesión
- *app/main/java/gui/mainMenu/admin*: Directorio que alberga la lógica detrás del menú principal del administrador
- *app/main/java/gui/mainMenu/evaluated*: Directorio que alberga la lógica detrás del menú principal del usuario de organización evaluada.
- *app/main/java/gui/mainMenu/evaluator*: Directorio que alberga la lógica detrás del menú principal del usuario de la *Fundación Miradas*.

- *connection/src/main/java/cli*: Directorio donde se albergan los modelos de las diferentes entidades manejadas por la aplicación.
- *connection/src/main/java/otea/connection*: Directorio donde se albergan los callers y los api del cliente
- *connection/src/main/java/misc*: Directorio donde se ubican clases auxiliares para formatear datos o comprobar los mismos.
- *oteaserver*: Directorio donde se encuentra el código del servidor.
- *videos*: Directorio donde se encuentran los vídeos del usuario.

D.3. Manual del programador

D.4. Compilación, instalación y ejecución del proyecto

Android Studio

En el lado del cliente cabe resaltar que se ha utilizado *Android Studio*, cuya descripción aparece en el apartado de Técnicas y Herramientas de la memoria de este proyecto. En este caso, la instalación consta de los siguientes pasos:

1. En la página de descarga de la aplicación se selecciona el botón **Android Studio Electric Eel** para descargar el instalador de Android Studio:

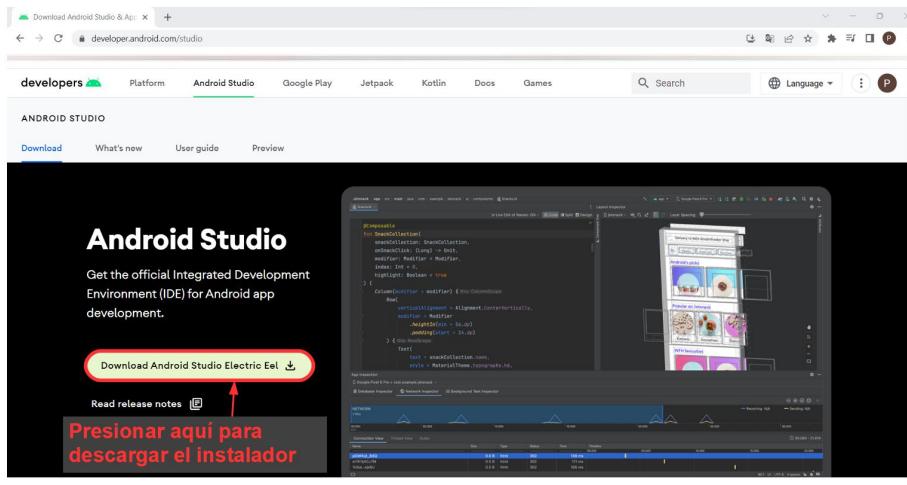


Figura D.1: Página de descarga del instalador de Android Studio

2. Posteriormente, se tienen que aceptar los términos y condiciones marcando la opción *I have read and agree with the above terms and conditions*, para presionar posteriormente el botón **Download Android Studio Electric Eel | 2022.1.1 for Windows**.



Figura D.2: Sección de términos y condiciones anterior a la descarga del instalador de Android Studio

3. Posteriormente se obtiene el instalador:

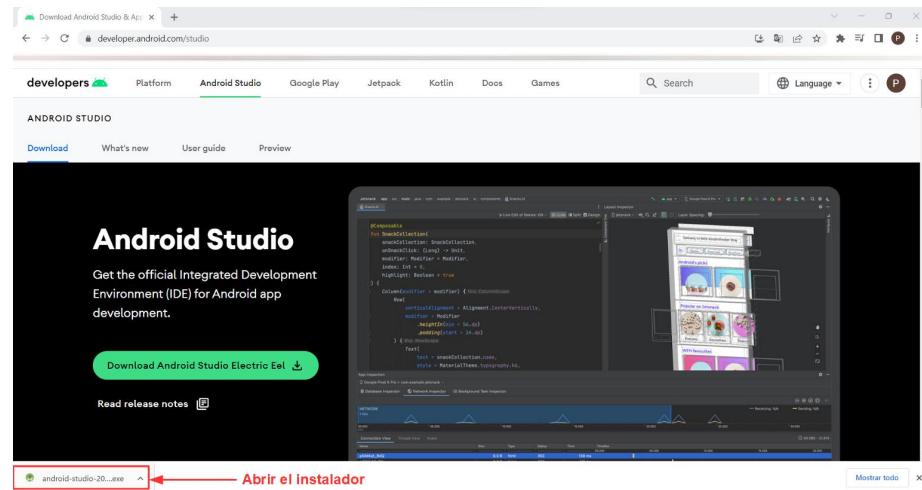


Figura D.3: El fichero de instalación ya ha sido descargado

4. Ya abierto el asistente de instalación, se tienen que seguir los pasos que nos piden:
 - a) En primer lugar se muestra el mensaje de bienvenida del asistente de instalación:

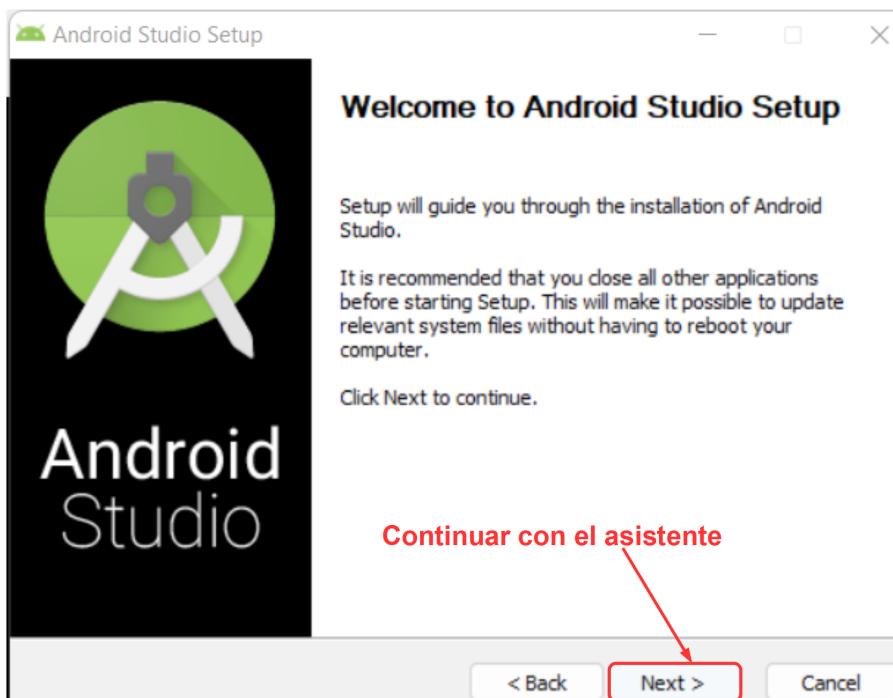


Figura D.4: Mensaje de bienvenida del instalador de Android Studio

- b) Posteriormente se seleccionan los componentes a instalar. Se seleccionan tanto *Android Studio*, que es la propia IDE de Google para el desarrollo de aplicaciones para Android, como *Android Virtual Device*, que se utiliza de forma integrada en Android Studio para la simulación del funcionamiento de las aplicaciones para diferentes aplicaciones:

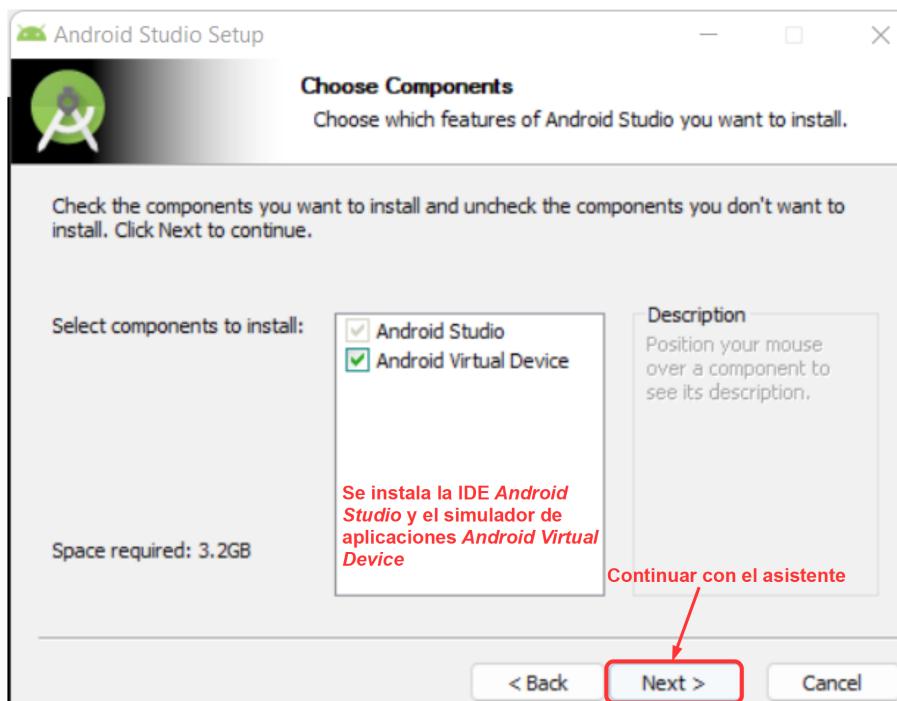


Figura D.5: Selección de componentes a instalar en el instalador de Android Studio

- c) Más adelante se establece un directorio para la instalación del programa. Se puede elegir libremente dicho directorio, pero en este caso se va a escoger el directorio por defecto:

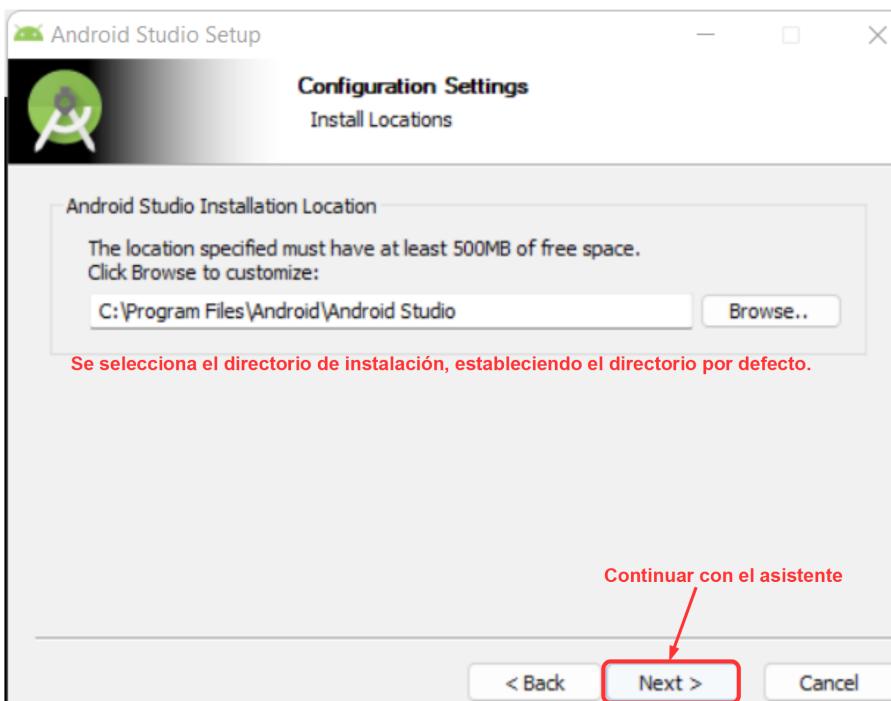


Figura D.6: Selección del directorio de instalación

- d) Además del directorio de instalación se puede establecer un directorio donde guardar el acceso directo al programa en el menú de inicio. No es un paso obligatorio, pero se establecerán las opciones por defecto:

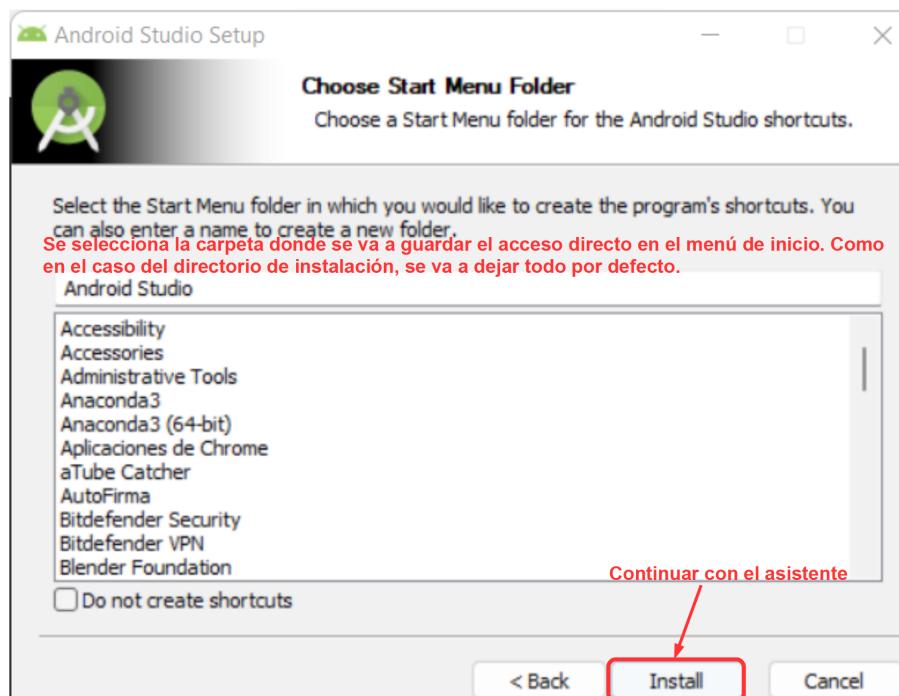


Figura D.7: Selección del directorio de creación de accesos directos en el menú de inicio

- e) Cuando se han seguido los pasos anteriores, se procede con la instalación:

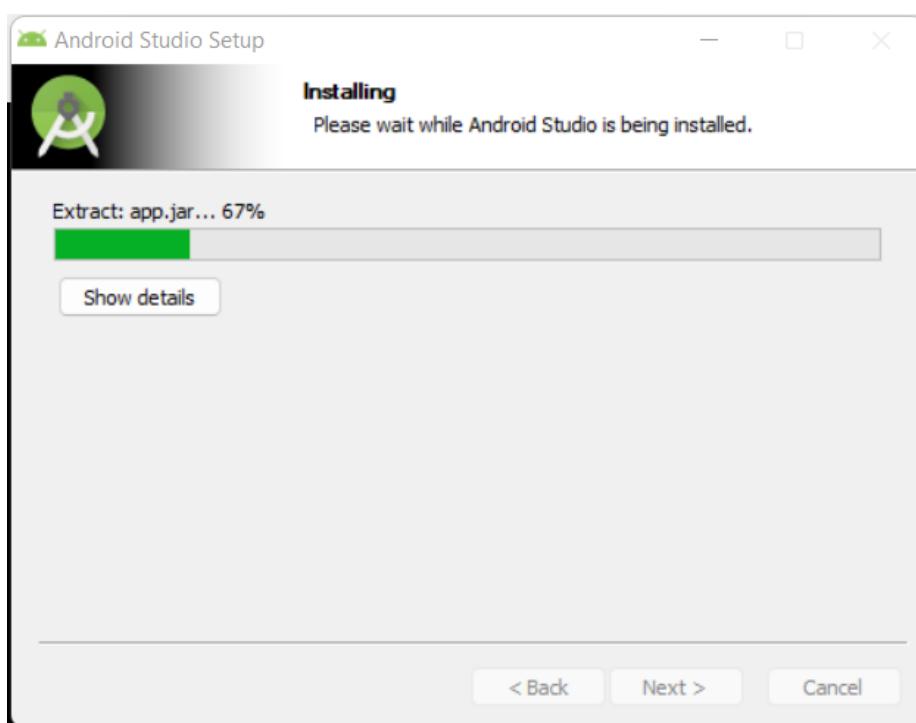


Figura D.8: Inicio de instalación de Android Studio

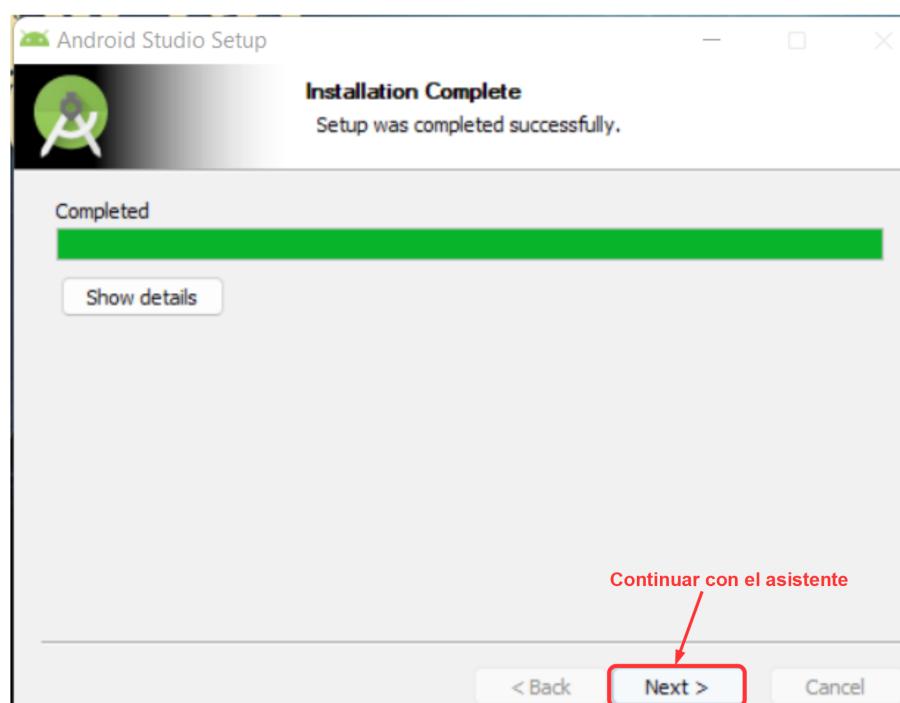


Figura D.9: Finalización de la instalación de Android Studio

- f) Por último se marca la opción de iniciar Android Studio para proceder así con la configuración del mismo:

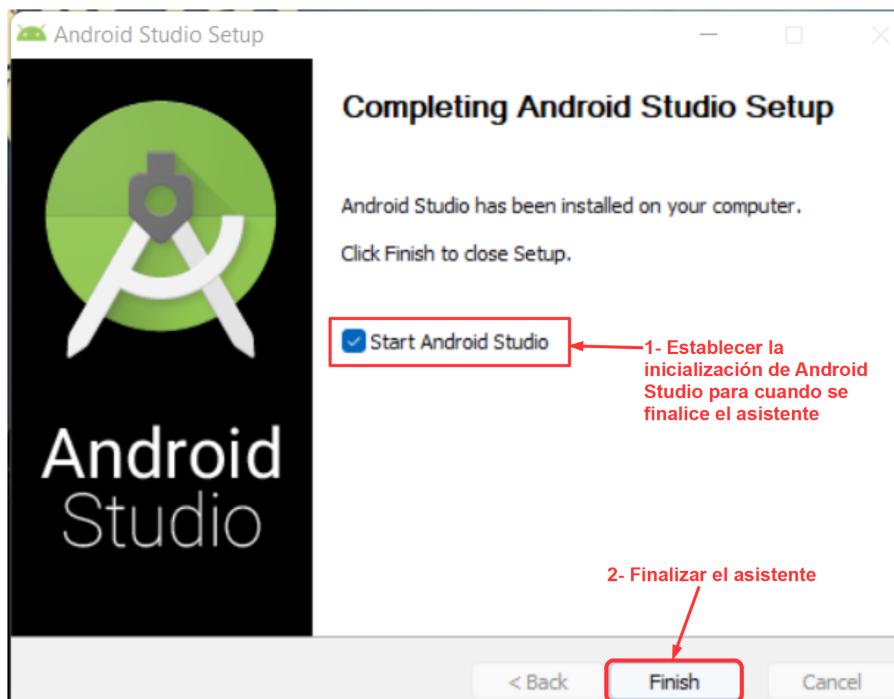


Figura D.10: Mensaje de finalización de la instalación

Tras haber finalizado con la instalación de *Android Studio*, se procede con su configuración inicial. En cuanto a la configuración inicial de *Android Studio* y de la aplicación, los pasos a seguir son los siguientes:

1. En primer lugar, *Android Studio* da la opción de importar la configuración de otro directorio, pero como es una instalación nueva, no se precisa de importar configuración alguna:
2. Posteriormente se muestra un mensaje para recolectar información sobre *Android Studio* y sus diferentes herramientas, el cual se va a rechazar:
- 3.

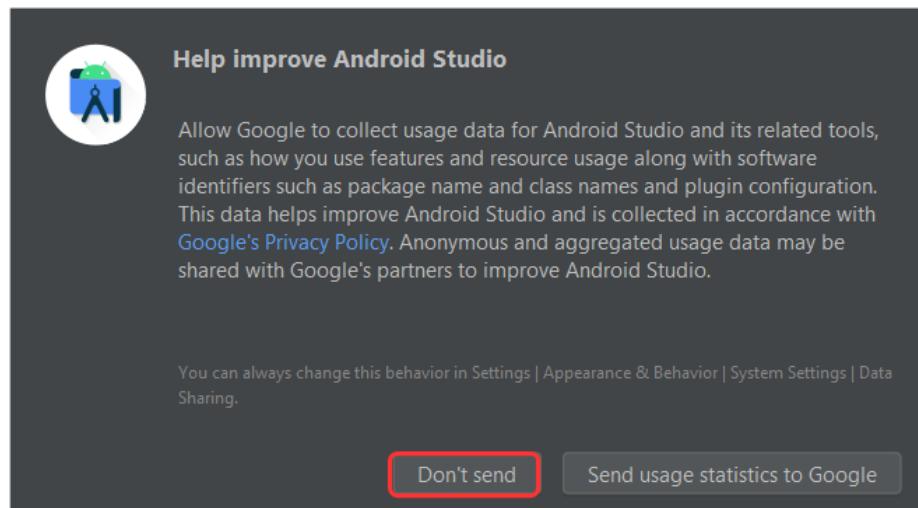


Figura D.11: Ventana emergente de importación de configuración

4. A continuación aparece el asistente de configuración de Android Studio, el cual aparece únicamente en la primera ejecución del programa. Para ir avanzando con el asistente se presiona el botón **Next**, el cual lleva al usuario al siguiente paso de la configuración:

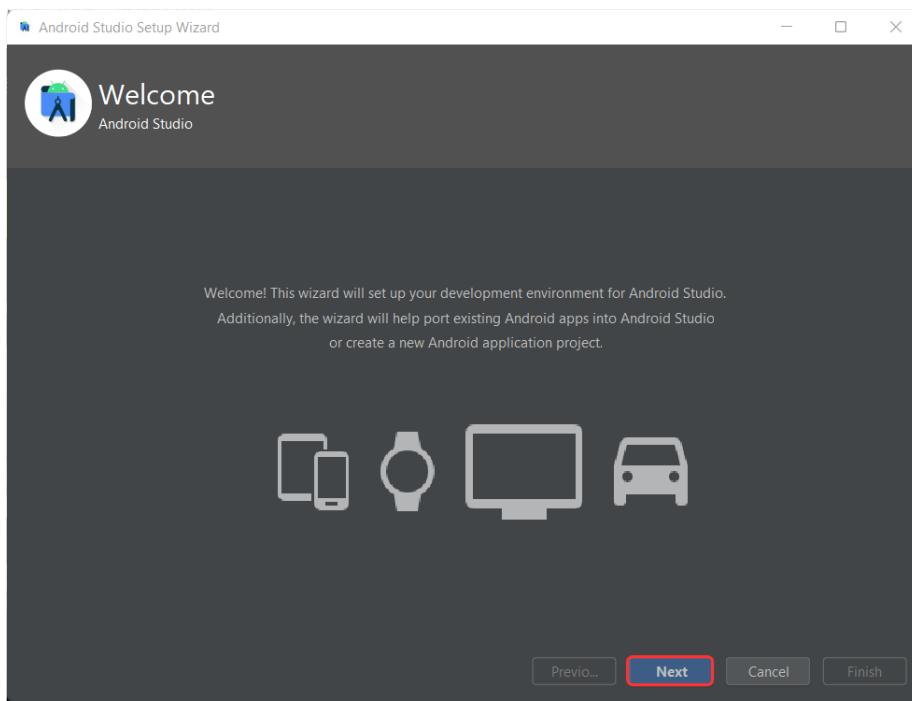


Figura D.12: Mensaje de bienvenida del asistente de configuración inicial de Android Studio

5. Como tipo de instalación, Android Studio proporciona dos diferentes opciones, **la opción estándar**, la cual instala los componentes predeterminados, y **la opción personalizada**, la cual permite al usuario seleccionar la configuración y los componentes a instalar:

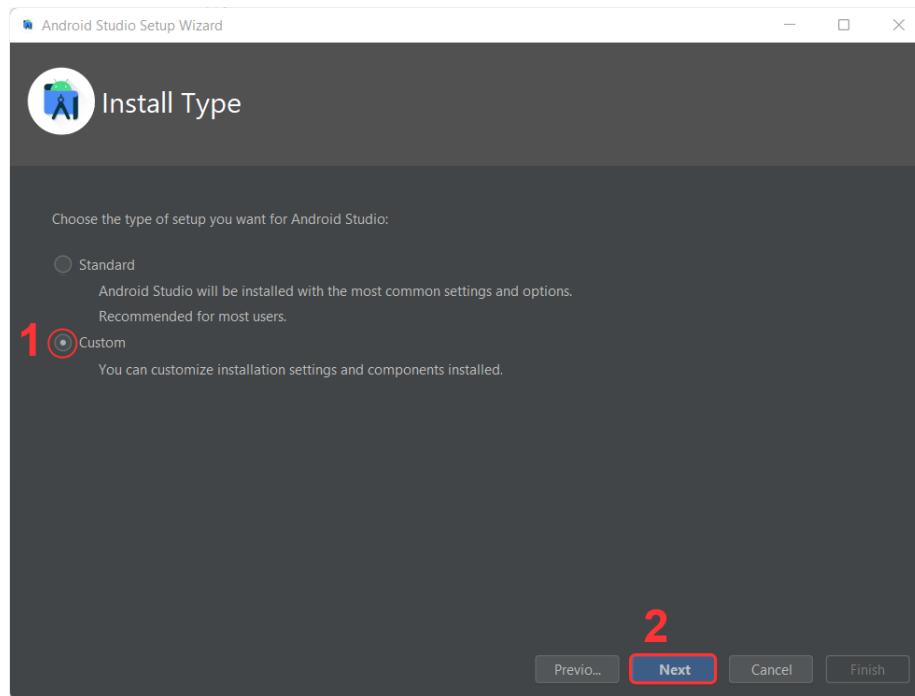


Figura D.13: Selección del tipo de instalación de los componentes de Android Studio

6. En cuanto a la selección del directorio del JDK, Android Studio mostrará el JDK instalado más reciente. También se puede personalizar si se dispone de más de un JDK en el equipo.

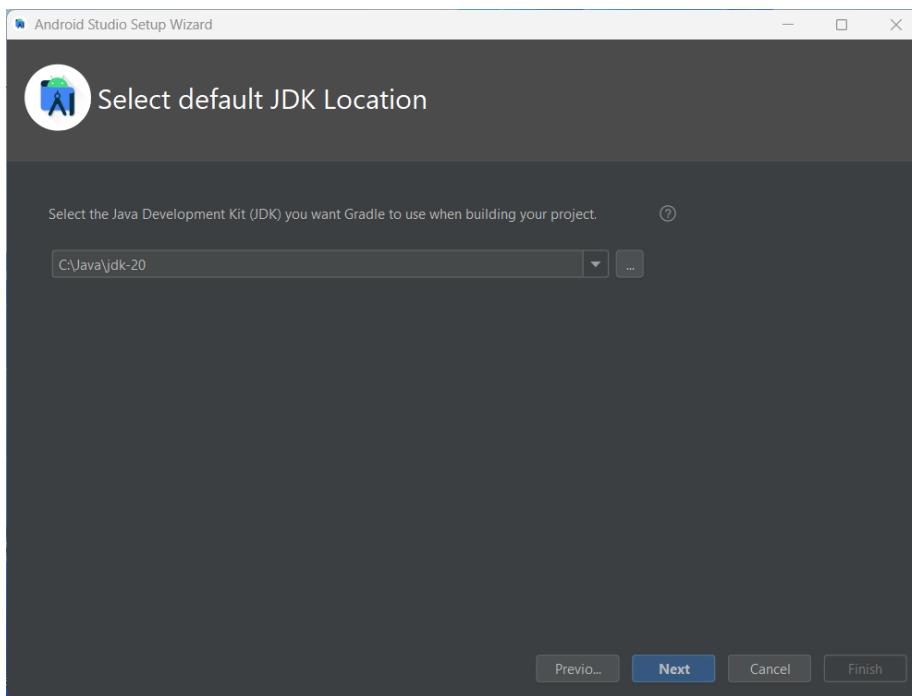


Figura D.14: Selección del directorio predeterminado del JDK

7. A continuación se selecciona el diseño de la interfaz, el cual es irrelevante para el correcto funcionamiento de la herramienta:

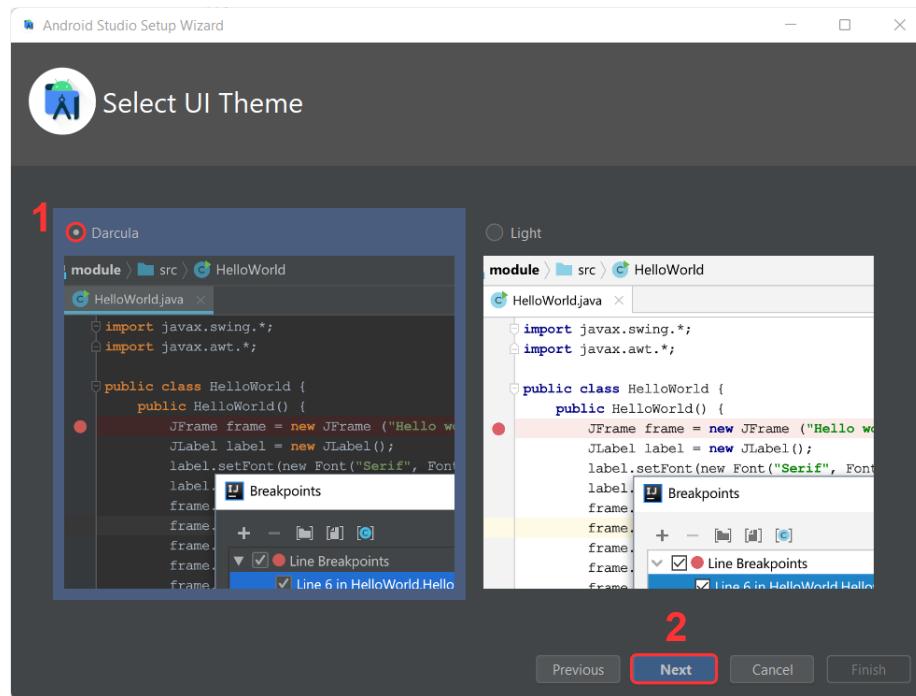


Figura D.15: Selección del diseño de la interfaz de Android Studio

8. Posteriormente se seleccionan los componentes SDK (*Software Development Kit*) a instalar, los cuales permiten desarrollar las aplicaciones para Android. En este caso se seleccionarán los componentes de la captura posterior:

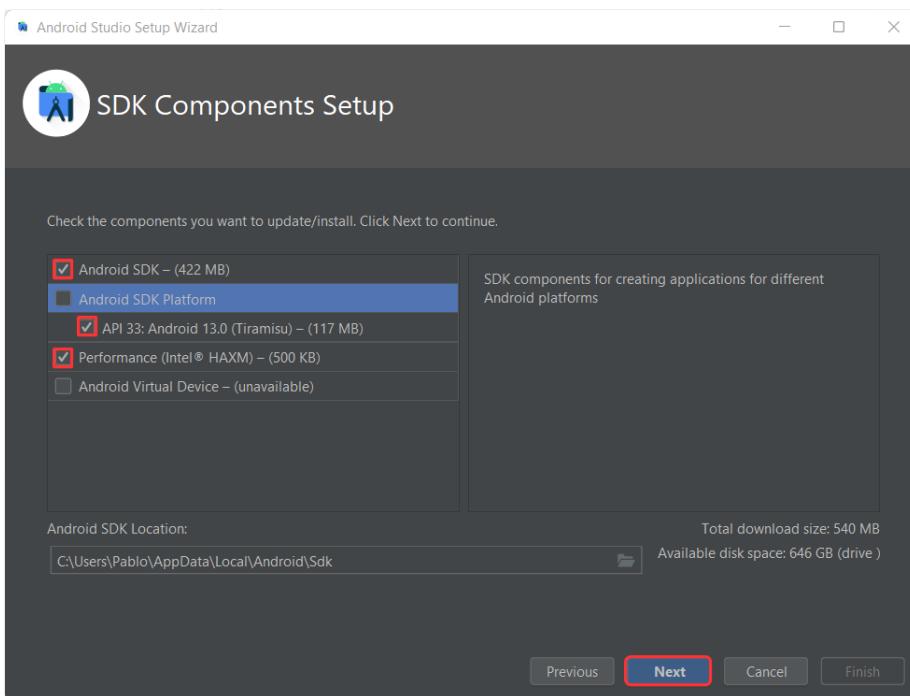


Figura D.16: Selección de componentes SDK a instalar en Android Studio

Como se puede comprobar, en el momento de haberse realizado la captura anterior, el componente Android Virtual Device no está disponible. Aun así, podemos proseguir con la instalación presionando **OK**:

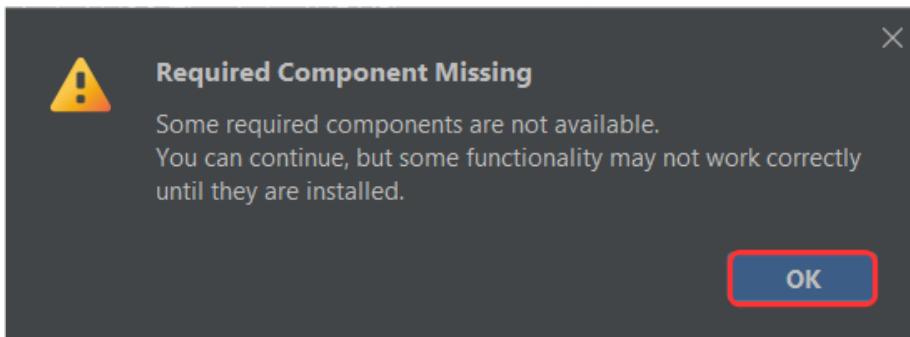


Figura D.17: Advertencia por falta de componentes a instalar

9. Posteriormente se configura la memoria RAM máxima que utilizará el emulador de aplicaciones. En este caso dejamos la opción recomendada de 2GB de RAM:

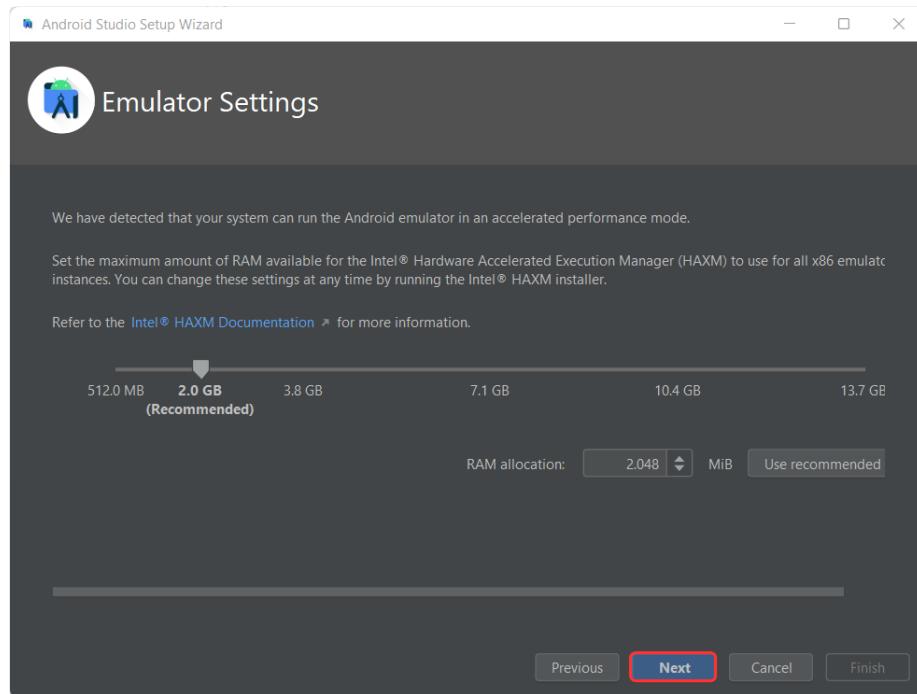


Figura D.18: Configuración de la memoria RAM utilizada por el emulador de Android

10. Más adelante se comprueban todos los componentes a instalar y a configurar durante la configuración inicial de Android Studio:

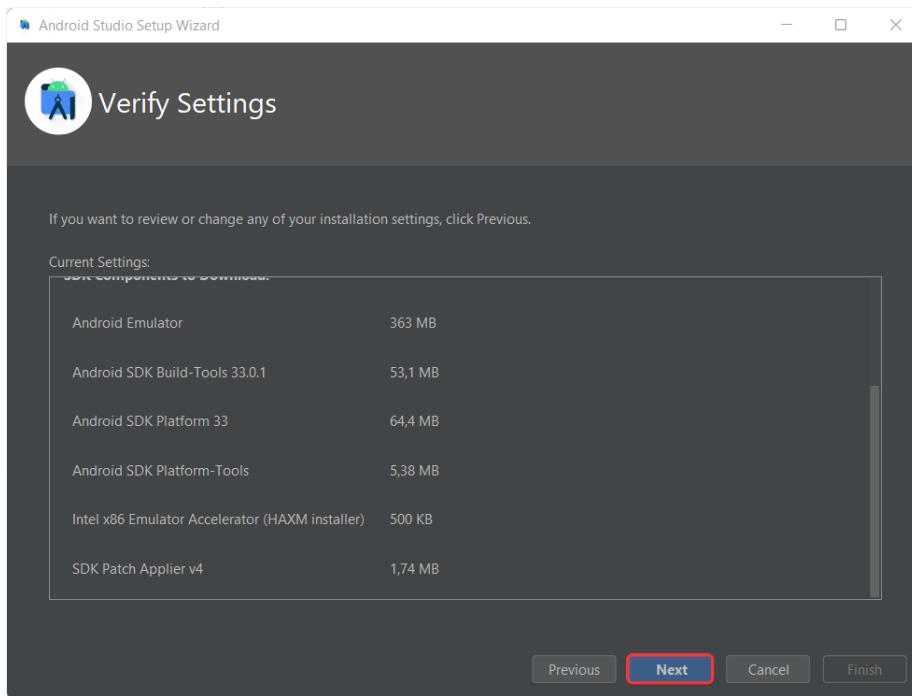


Figura D.19: Verificación de componentes a instalar en Android Studio

11. Antes de instalar los componentes, se tienen que aceptar la licencia del software tanto del *paquete SDK* como del *acelerador de emulación de Intel*:

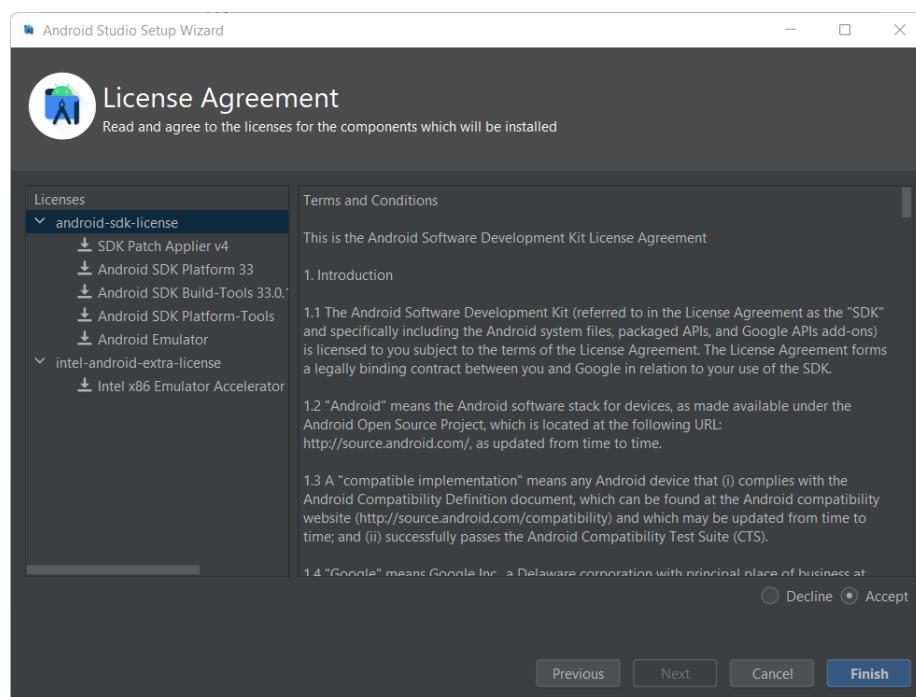


Figura D.20: Licencia de software del paquete SDK

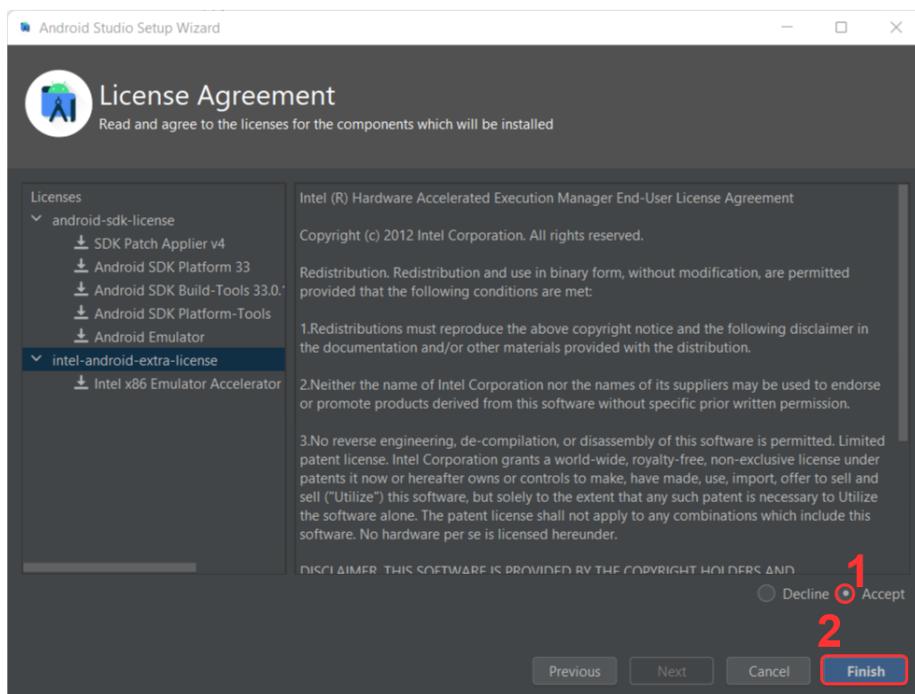


Figura D.21: Licencia de software del acelerador de emulación de Intel

12. Por último se procede con la instalación de los componentes mencionados con anterioridad y se finaliza con el asistente de configuración inicial presionando el botón **Finish**:

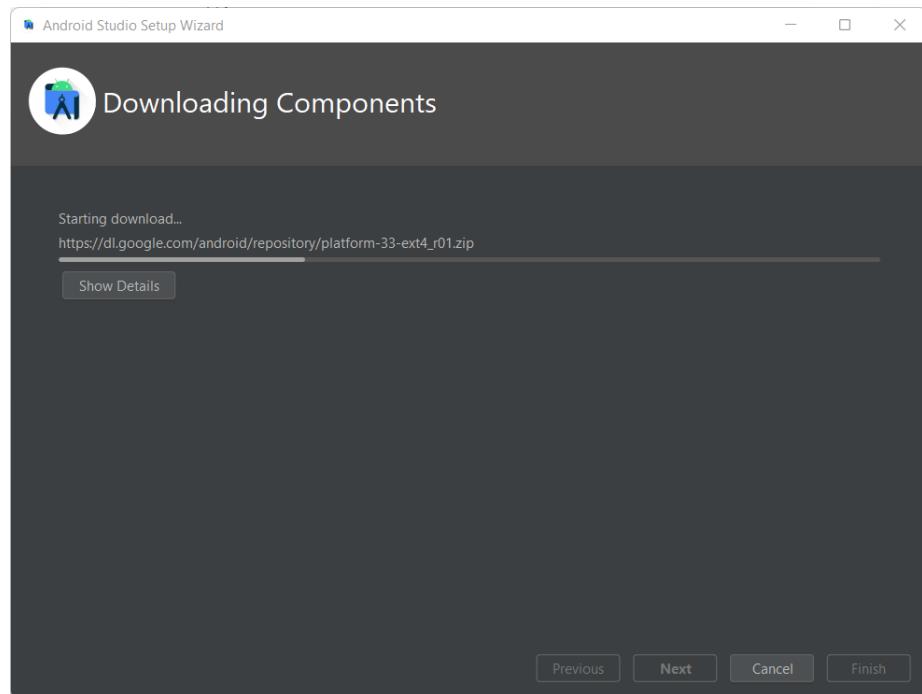


Figura D.22: Instalación de los componentes en la configuración inicial

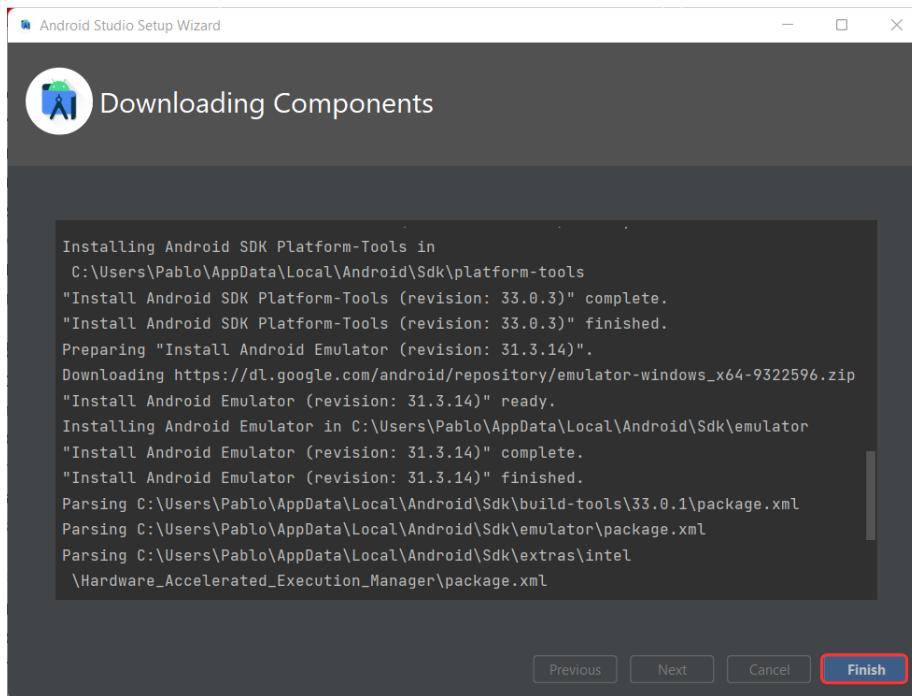


Figura D.23: Finalización del asistente de instalación de componentes

Tras haber finalizado con la instalación y con la configuración inicial de Android Studio, se tiene que iniciar un proyecto inicial donde se va a desarrollar la aplicación. Los pasos a seguir son los siguientes:

1. En primer lugar en la pantalla de bienvenida se presiona sobre el botón **New Project** para crear un proyecto de aplicación nuevo:

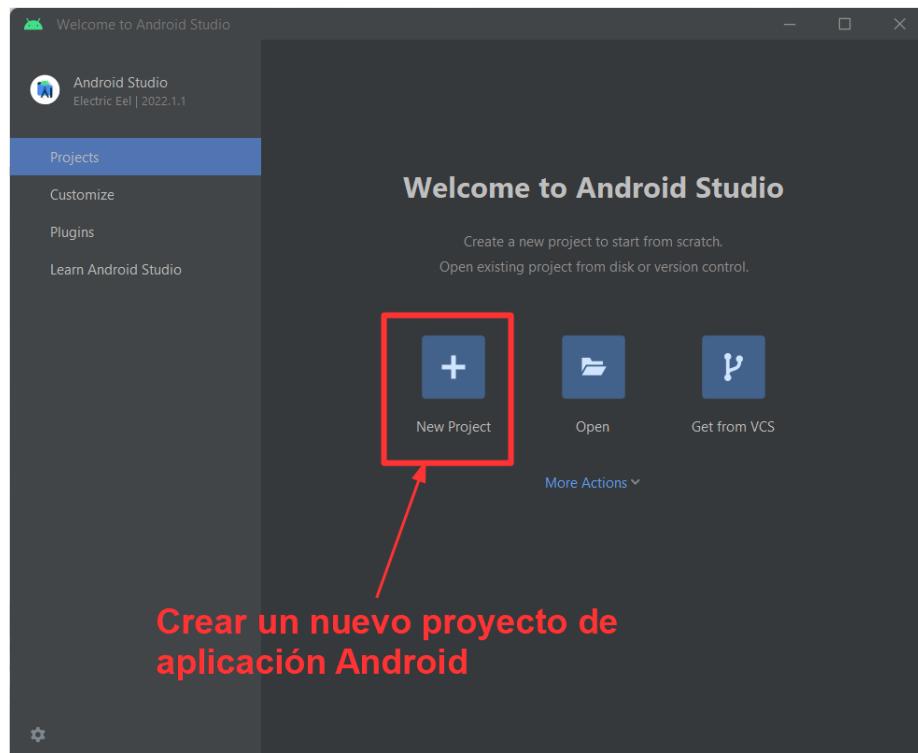


Figura D.24: Pantalla de selección de proyecto

2. Posteriormente se selecciona el diseño de la actividad principal de la aplicación. Para simplificar la estructura inicial del proyecto se selecciona la actividad vacía o *Empty Activity*. Al tener la actividad ya seleccionada se presiona el botón **Next** para continuar con la configuración de la actividad:

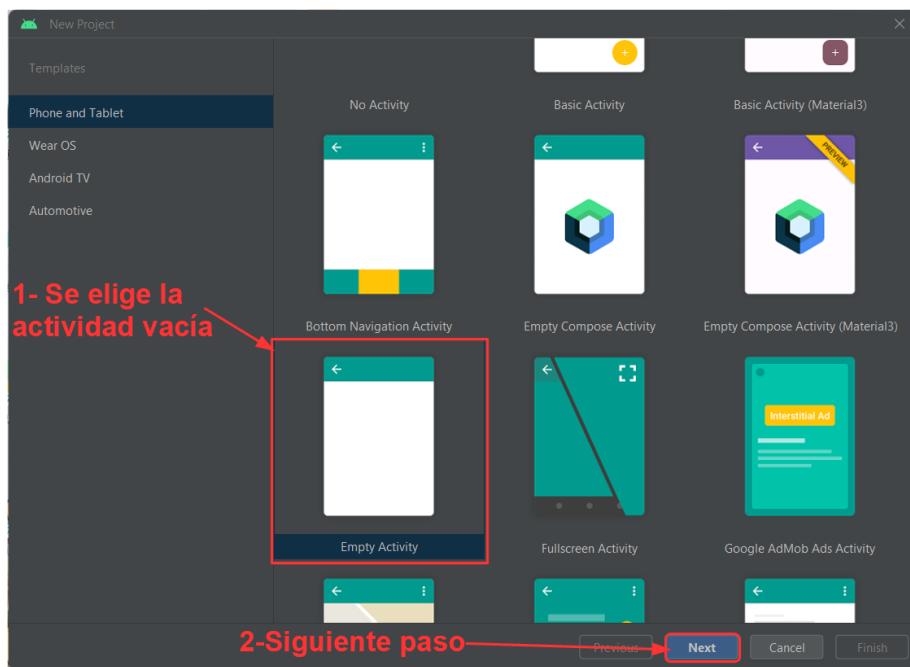


Figura D.25: Pantalla de selección de actividad principal

3. Ya en la pantalla de configuración de la actividad, se puede modificar el nombre de la misma, el nombre del paquete a la que pertenece, el directorio donde se va a guardar, el lenguaje de programación en la que va a ser programada y la versión mínima de Android en la que puede ser ejecutada:

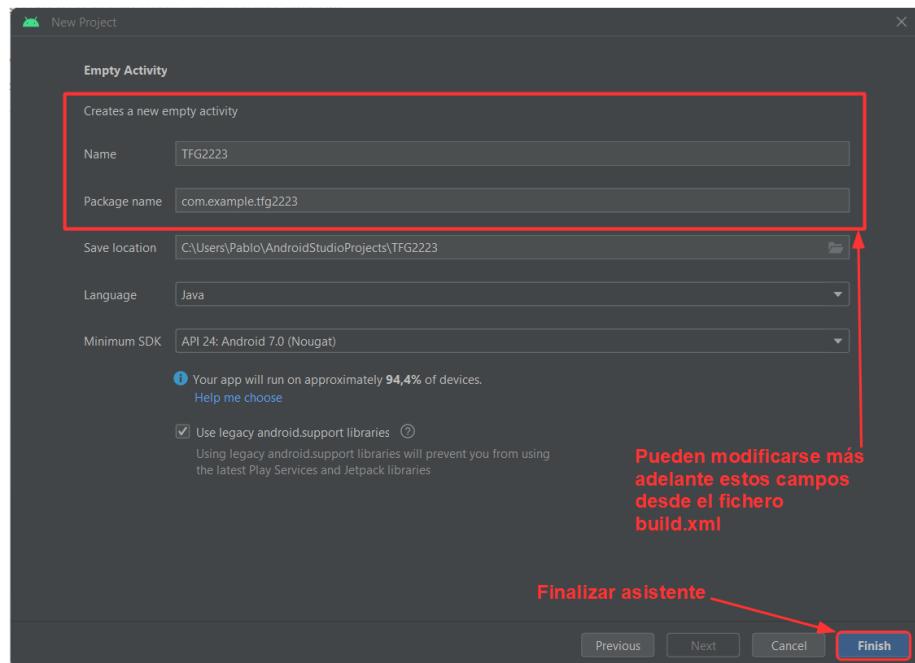


Figura D.26: Pantalla de configuración de la actividad

Cuando ya se ha configurado la actividad, se instalará el soporte JDK para la emulación de la misma:

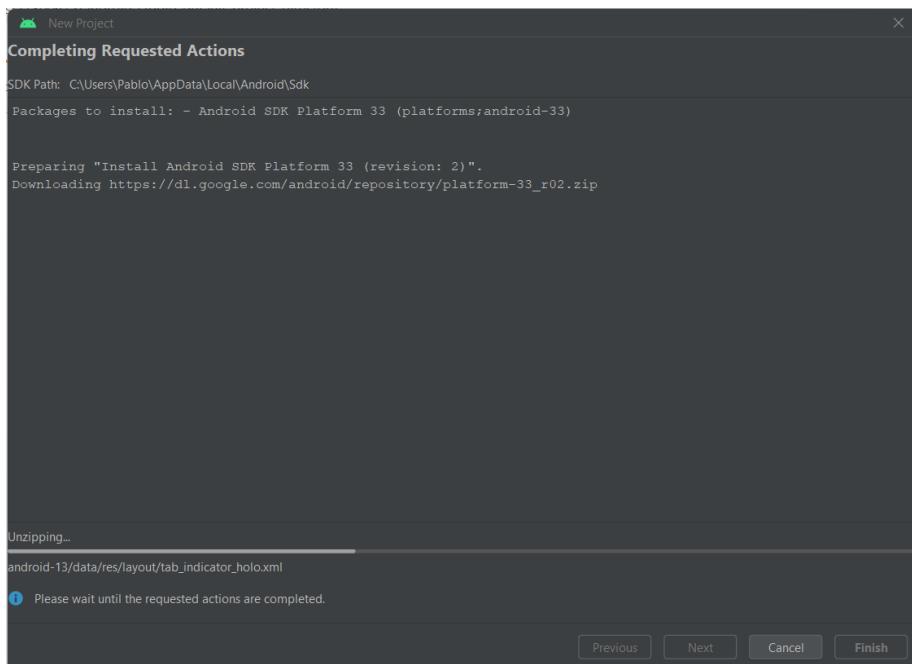


Figura D.27: Desarrollo del proceso de instalación del soporte JDK

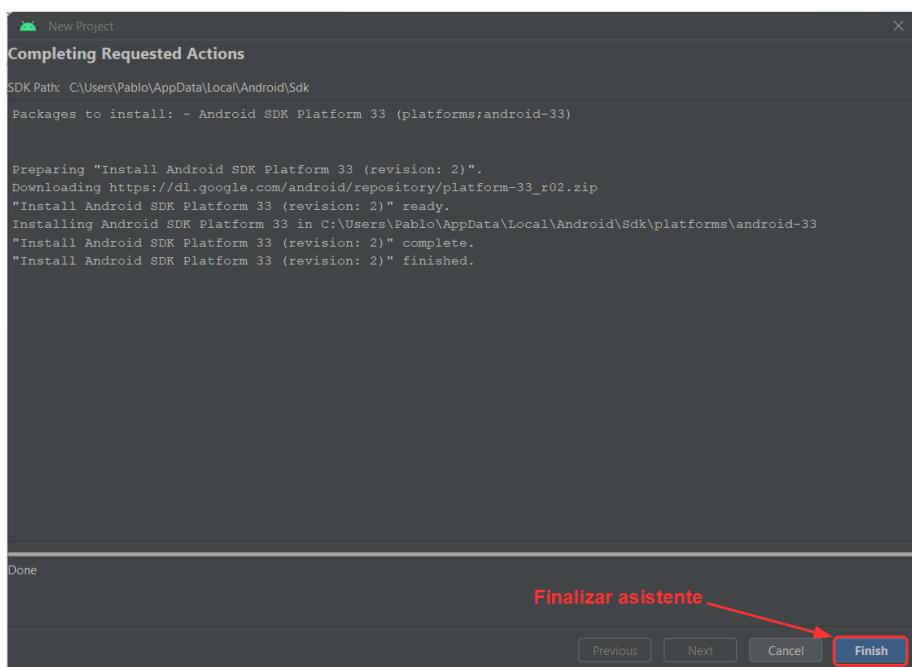


Figura D.28: Finalización del proceso de instalación del soporte JDK

Cuando ya se haya terminado con el asistente de creación del proyecto, ya nos mostrará el código Java de esta actividad vacía que acabamos de crear, además de que en la parte izquierda de la pantalla encontramos todas las rutas creadas dentro del directorio del proyecto.

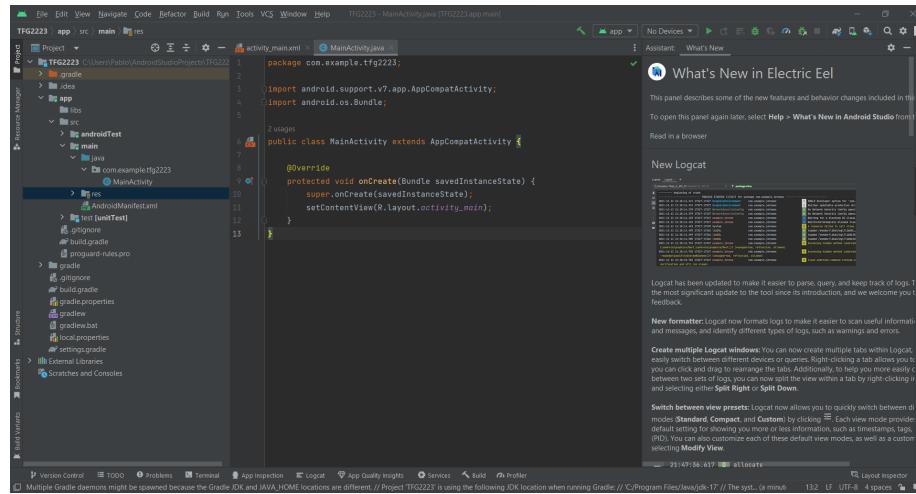


Figura D.29: Interfaz de Android Studio con el proyecto Android recién creado

4. Tras haberse creado el proyecto, se va a proceder a añadir una cuenta de GitHub para publicar los progresos que vayan realizándose durante el desarrollo de la aplicación. Para ello en primer lugar se va a **File → Settings:**

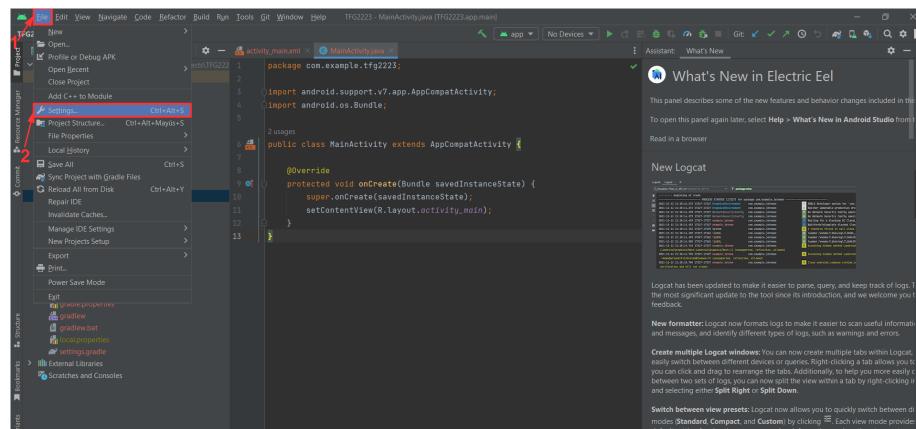


Figura D.30: Acceso a las opciones de Android Studio

5. Posteriormente se tiene que acceder a **Version Control → GitHub**

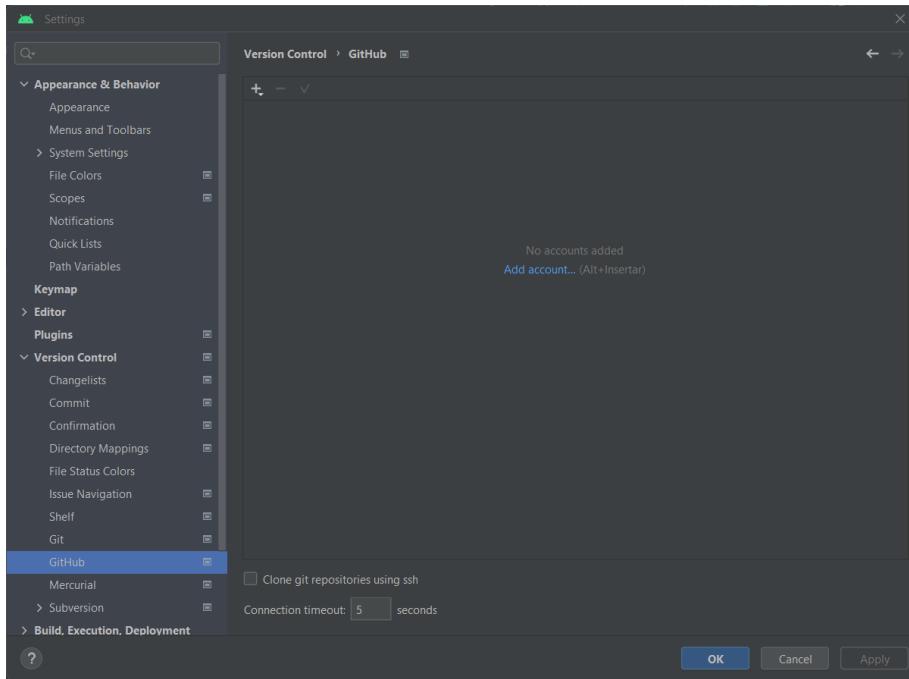


Figura D.31: Pantalla de control de cuentas de GitHub

6. Más adelante se accede a la configuración de GitHub, concretamente en **Settings → Developer Settings → Personal access tokens → Tokens (classic)**, para ir a la página de muestra de tokens de usuario **Generate new token → Generate new token (classic)**:

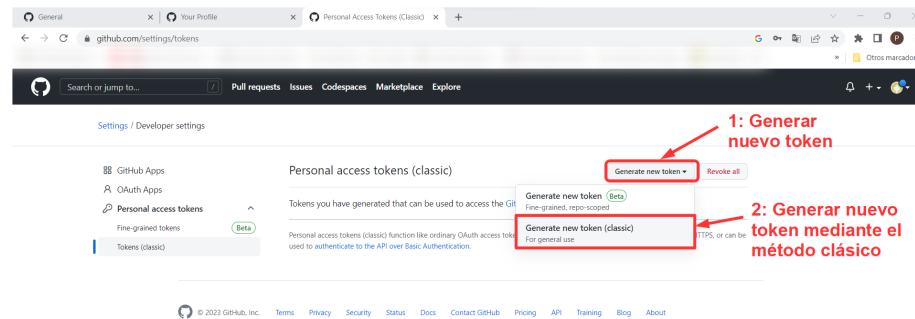


Figura D.32: Pantalla de tokens de usuario antes de ir a la pantalla de generación de tokens

7. En cuanto a la generación del token, se menciona en una nota para qué queremos utilizar ese token, la fecha de expiración del mismo (la cual se aconseja que sea de duración hasta junio, fecha de la defensa) y se seleccionan todos los permisos de uso del token:

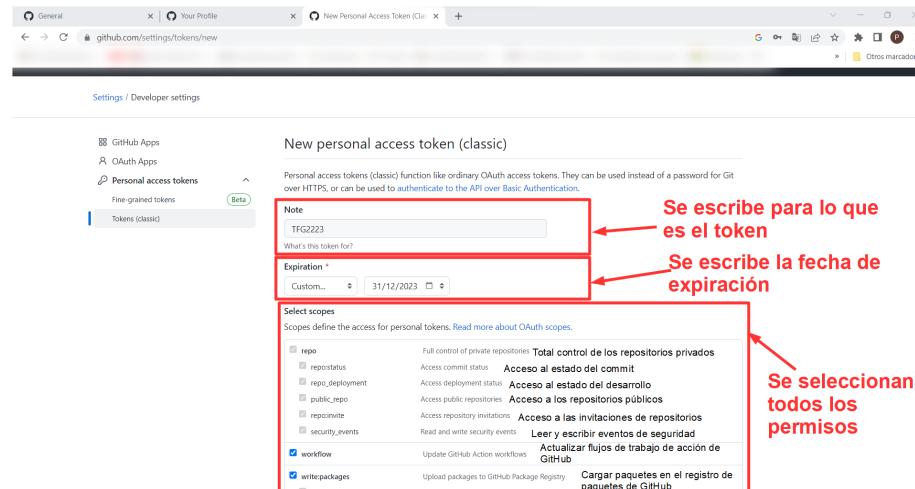


Figura D.33: Pantalla de generación de tokens con los permisos de usuario

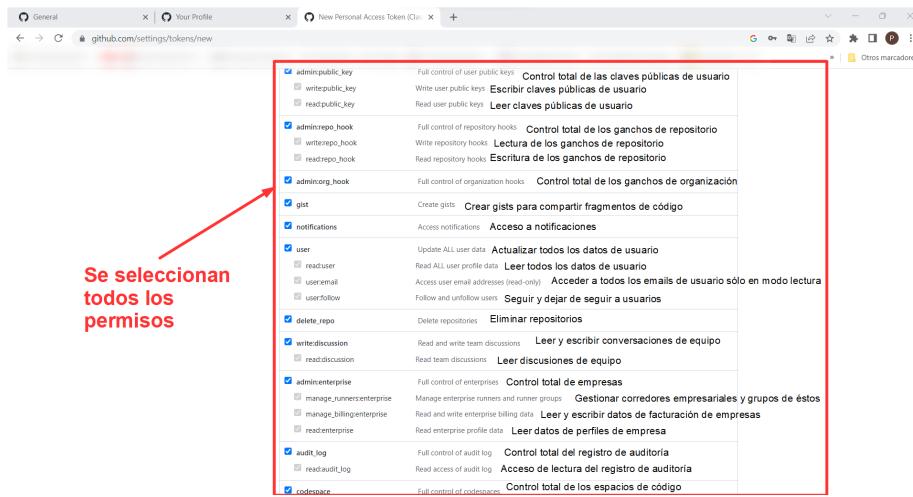


Figura D.34: Siguientes permisos de usuario

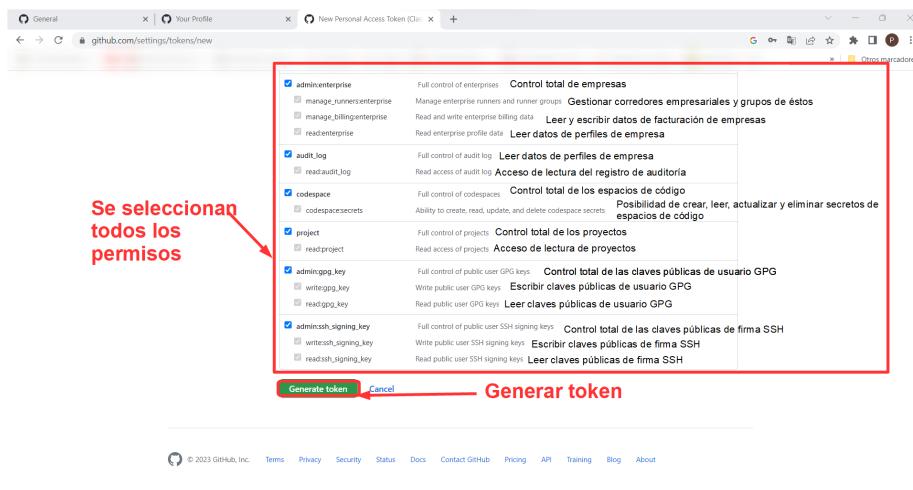


Figura D.35: Últimos permisos de usuario

8. Con el token recién generado, se copia para poder añadirlo a Android Studio:

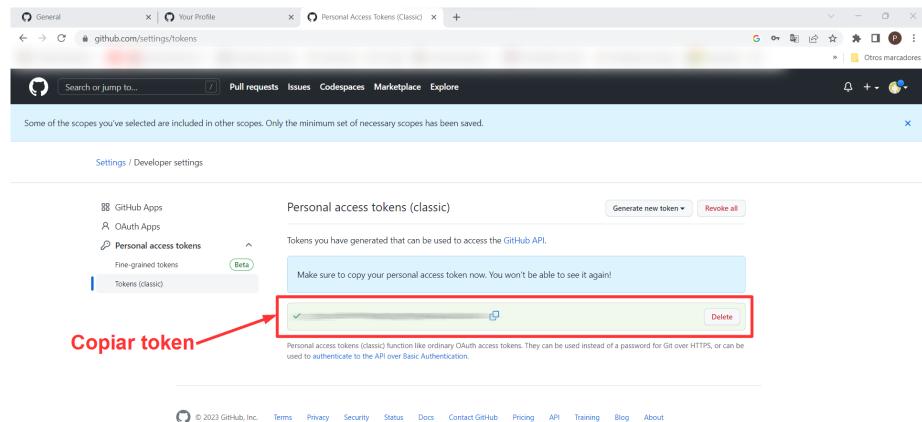


Figura D.36: Pantalla de tokens de usuario con el token de usuario generado

En la pantalla de control de cuentas de GitHub en Android Studio, se presiona en **Add account** y se añade el token

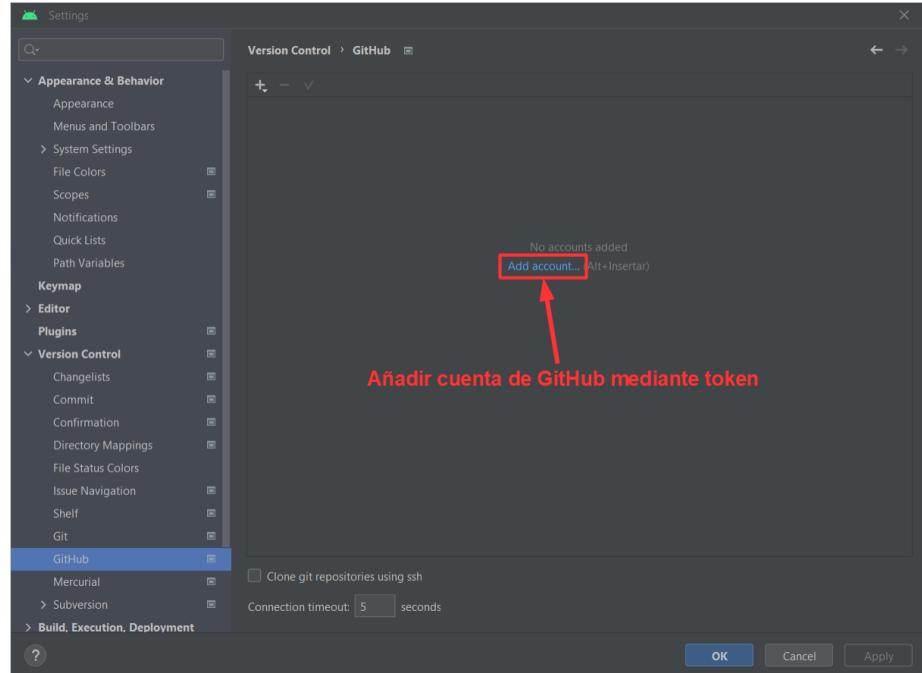


Figura D.37: Pantalla de control de cuentas de GitHub lista para añadir usuario

Para confirmar la agregación del token, se presiona sobre el botón **Add Account**:

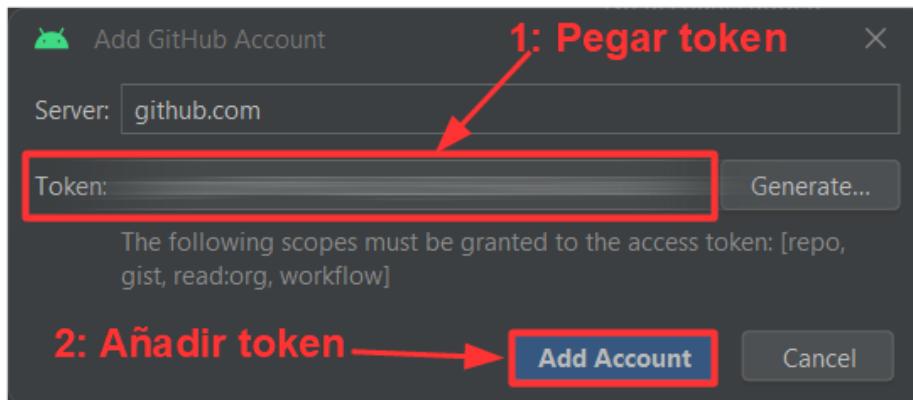


Figura D.38: Añadir token de usuario de GitHub en Android Studio

Con el token ya añadido, se presiona el botón **Ok**:

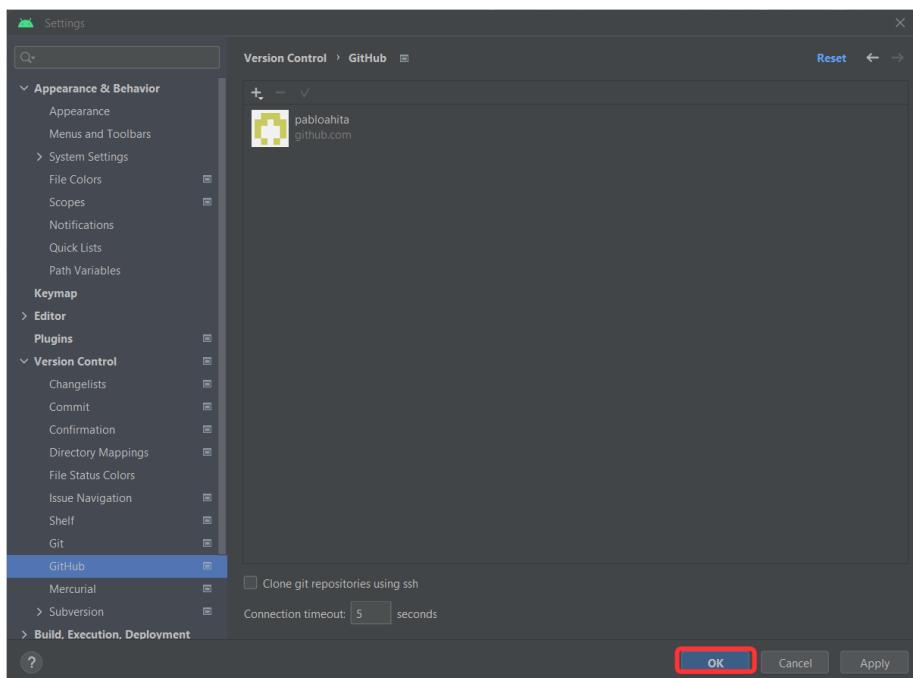


Figura D.39: Cuenta de GitHub recién añadida

9. Tras haber añadido satisfactoriamente la cuenta de GitHub a Android Studio, se va a proceder a añadir un dispositivo virtual. Para ello

se accede al menú desplegable con los dispositivos y posteriormente al administrador de dispositivos, es decir, **No Device** → **Device Manager**:

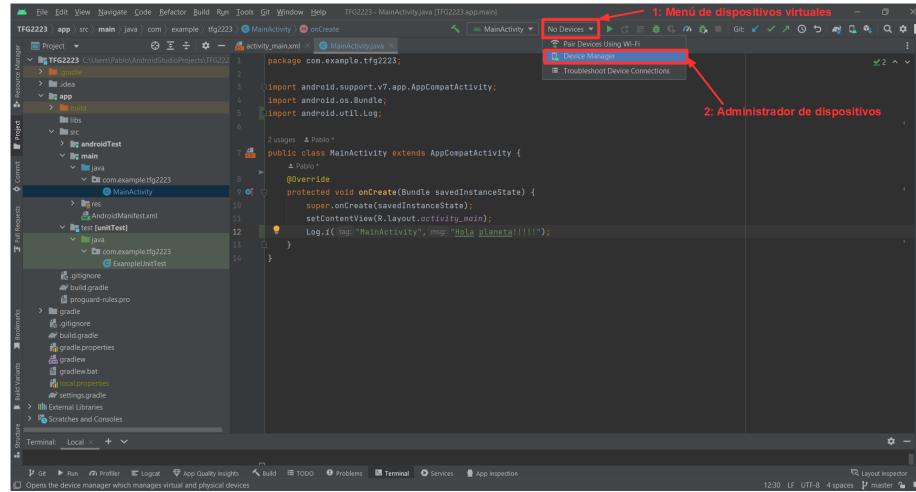


Figura D.40: Acceso al administrador de dispositivo mediante el menú desplegable de dispositivos

10. Posteriormente sale en la parte derecha de la pantalla el administrador de dispositivos (**Device Manager**), el cual muestra todos los dispositivos virtuales que se utilizan durante las simulaciones de funcionamiento. Para añadir un dispositivo virtual basta con presionar el botón **Create Device**:

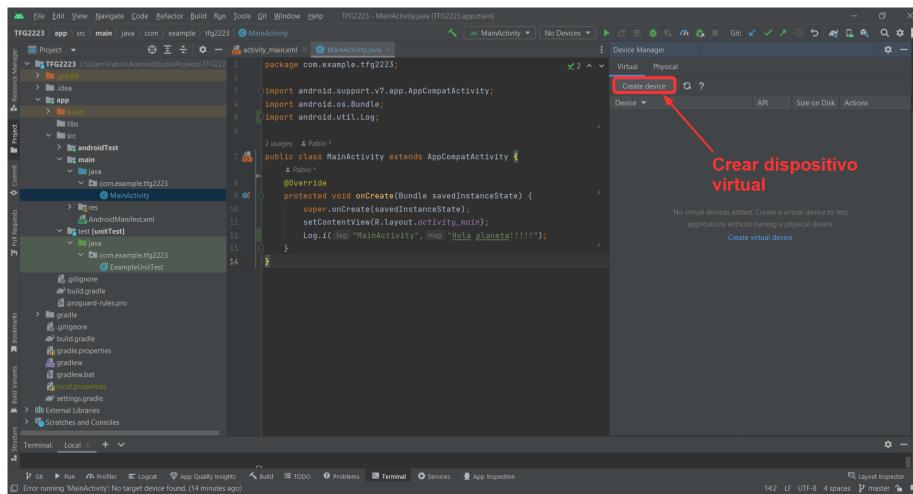


Figura D.41: Administrador de dispositivos

11. En cuanto a la configuración del nuevo dispositivo virtual, existen diferentes opciones para elegir, pudiendo simular aplicaciones para smartphones, tablets, dispositivos wearable, escritorio, smart TV y Android Auto. Como primer dispositivo se va a añadir el smartphone *Pixel 6 Pro* de Google. Posteriormente se presiona el botón **Next** para seguir con el siguiente paso:



Figura D.42: Selección del nuevo dispositivo virtual

12. Posteriormente se elige el sistema operativo de este dispositivo virtual. Para ello se va a seleccionar dicho sistema operativo para descargarlo para el dispositivo y posteriormente se presiona sobre **Next**:

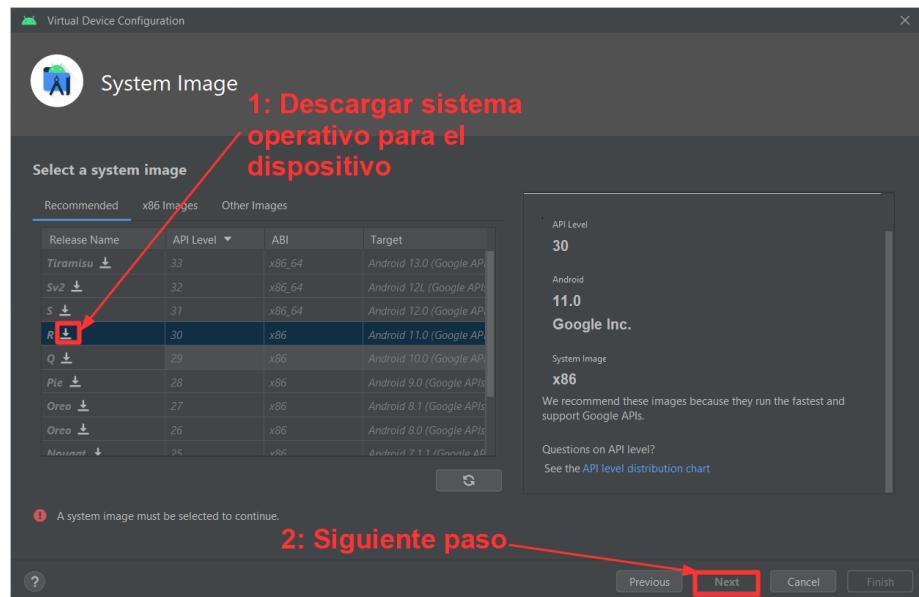


Figura D.43: Selección de sistema operativo para el nuevo dispositivo virtual

13. Tras haber seleccionado la configuración tanto de hardware como de software se procede con la instalación tanto del dispositivo como de su sistema operativo. Al finalizar se presiona el botón **Finish**:

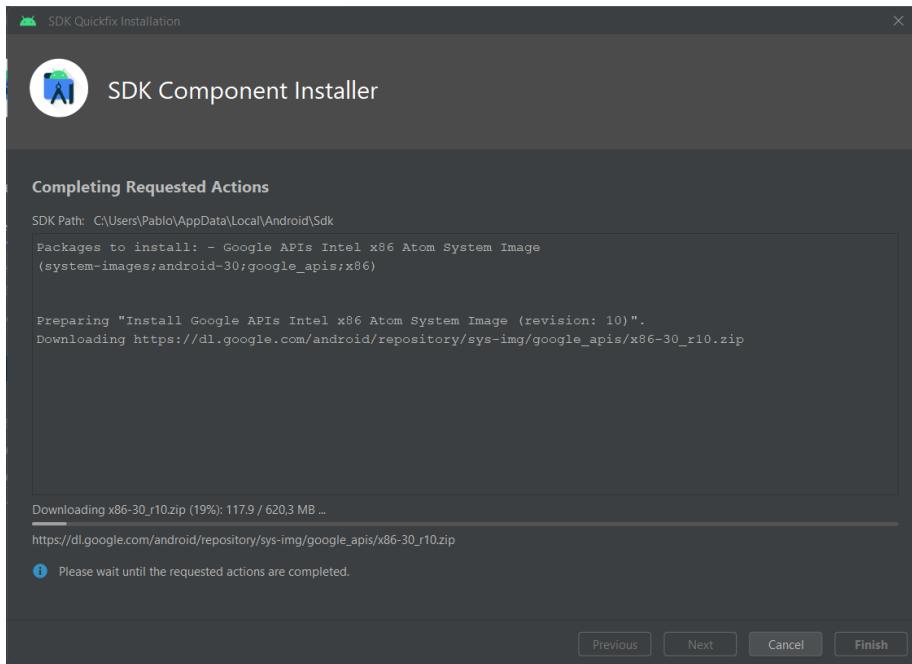


Figura D.44: Proceso de instalación del sistema operativo y de configuración del dispositivo virtual



Figura D.45: Finalización del proceso de instalación del sistema operativo y de configuración del dispositivo virtual

14. Para finalizar se puede comprobar la configuración que se ha establecido para el dispositivo nuevo, además de poder cambiar la orientación inicial del mismo. Cuando se quiera finalizar con la creación del dispositivo virtual, se presiona el botón **Finish**:



Figura D.46: Finalización del asistente de creación del dispositivo y comprobación de la configuración

Al tener la cuenta de GitHub agregada en Android Studio, se pueden realizar todas las operaciones de git desde la propia interfaz. Por lo tanto, los pasos a seguir para hacer un commit y publicarlo en GitHub son los siguientes:

1. En primer lugar hay que desplazarse a la barra de herramientas superior y ahí acceder a la ruta **Git** → **Commit**. Alternativamente se puede acceder a la configuración del nuevo commit mediante el comando *Ctrl+K*:

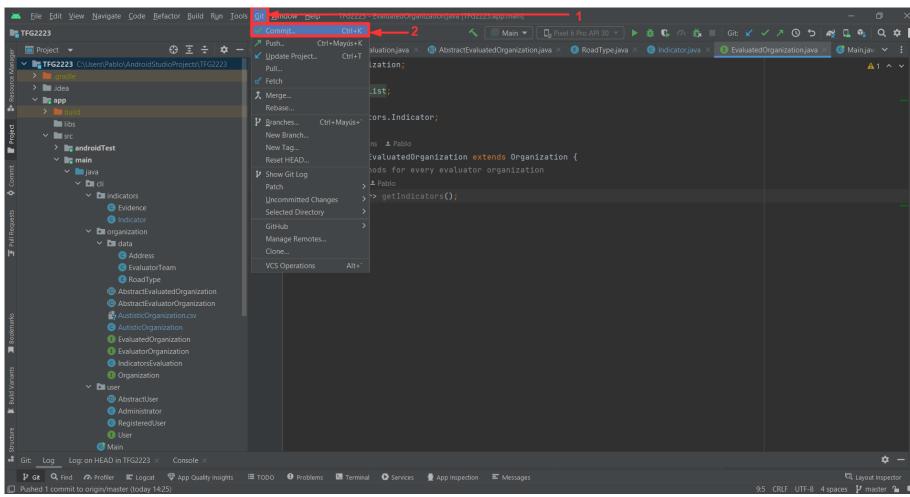


Figura D.47: Acceso a la configuración del nuevo commit

2. Posteriormente se seleccionan los ficheros que hayan tenido cambios, se escribe el mensaje de commit y se presiona en el botón ***Commit and Push***:

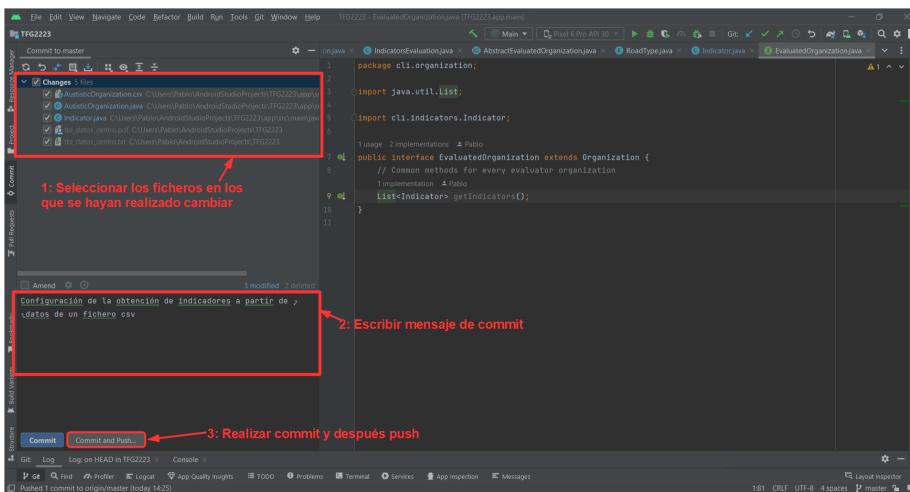


Figura D.48: Configuración del nuevo commit

Tras presionar ese botón analizará los posibles fallos y warnings que tenga el código de la aplicación. En la siguiente captura se han detectado cuatro warnings, dos en la clase **Indicator** y dos en la clase **AutisticOrganization**. Se puede optar por solucionar esos warnings y reiniciar la comprobación anteriormente mencionada, o por realizar las

operaciones de commit y push mediante el botón ***Commit Anyway and Push:***

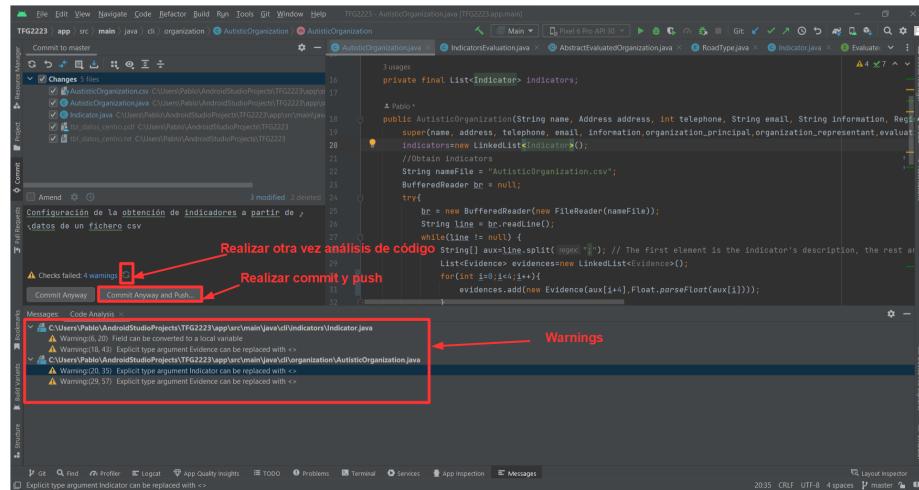


Figura D.49: Soluciones a warnings durante el proceso de commit

3. Tras haber solucionado todos los warnings, se presiona el botón ***Push*** para publicar los cambios realizados en GitHub:

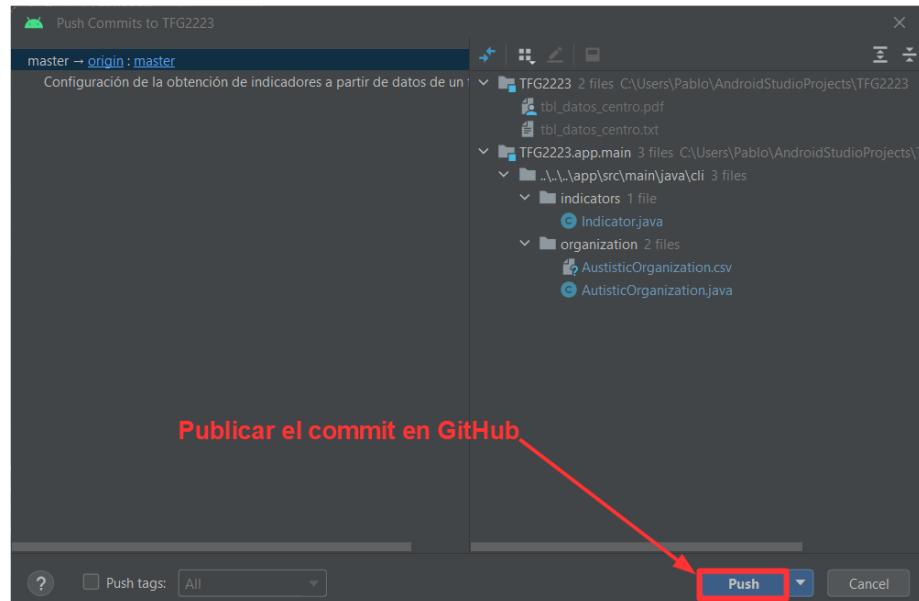


Figura D.50: Menú de publicación de cambios realizados

Al acabar con todo este proceso sale este mensaje:

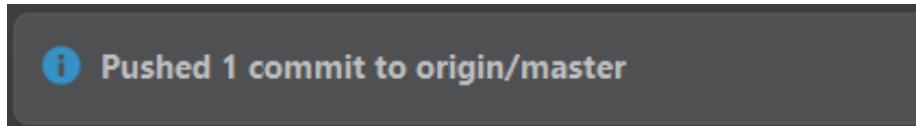


Figura D.51: Commit y push realizados satisfactoriamente

Azure Portal

Para crear un webservice con una base de datos en *Microsoft Azure*, se deben seguir los siguientes pasos:

1. En primer lugar se tiene que acceder al portal de *Azure* e iniciar sesión con la cuenta ya creada. Ya con la sesión iniciada, en la sección *Servicios de Azure*, se selecciona la opción ***App Services***, posteriormente en ***Crear*** y por último en ***Aplicación web + base de datos***, como se muestra en la siguiente captura:

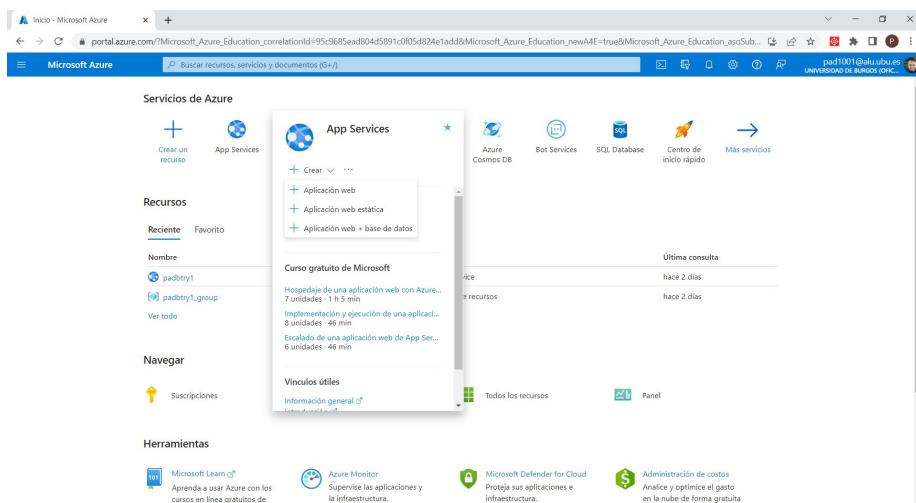


Figura D.52: Acceso a la creación de la web app desde el menú principal del portal de *Azure*

2. Posteriormente, se procede a la configuración de la aplicación base y de la base de datos. Como se puede comprobar en los ***Detalles del proyecto***, la ***suscripción elegida*** debe dejarse por defecto ya que

se trata de la suscripción asociada a la cuenta de *Azure*, el **grupo de recursos** debe ser nuevo y la **región** puede ser cualquiera de las disponibles. En cuanto a los **detalles de la aplicación web**, el **nombre** puede ser cualquiera que se encuentre disponible en ese momento y la **pila del entorno en tiempo de ejecución** debe ser *.NET 6* ya que el servidor está programado en C# utilizando *ASP.NET* (en caso de utilizarse Java, hay que seleccionar la versión en la que se compila el servicio web como *JAX-RS*).

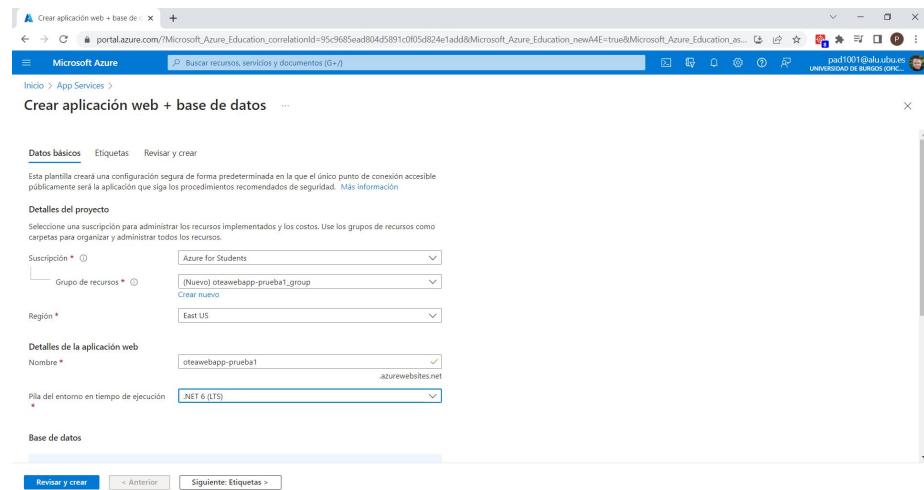


Figura D.53: Configuración de los detalles del proyecto y de la aplicación web del nuevo *App Service* con *Azure SQL*

En cuanto a la **base de datos**, el **motor** a elegir es *SQLAzure*, debido a su compatibilidad e integración con *Transact-SQL*, la adaptación de *Microsoft* del lenguaje SQL. El **nombre del servidor de la base de datos** y el **nombre de la base de datos** son personalizados bajo disponibilidad, recomendando utilizar los sufijos **server** y **database** respectivamente para poder relacionarlos con facilidad con la web app. Como no se va a agregar **Azure Cache for Redis** y el **plan de hospedaje** va a ser estándar, se presiona en *Revisar y crear* para comprobar antes de crear todo si la configuración es la deseada para lo que se le va a utilizar.

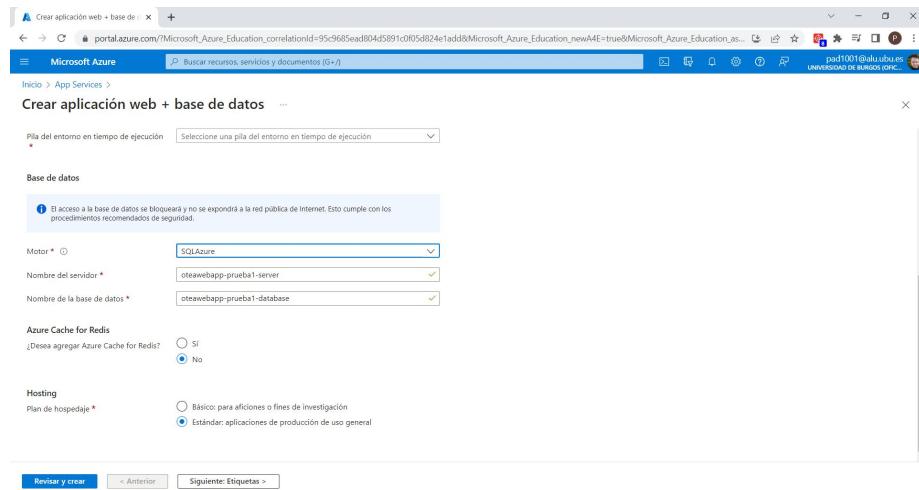


Figura D.54: Configuración de la base de datos del nuevo *App Service* con *Azure SQL*

3. Tras esperar a la validación de los cambios, se comprueba que todo esté configurado según las necesidades que se dispongan y posteriormente se presiona el botón *Crear*.

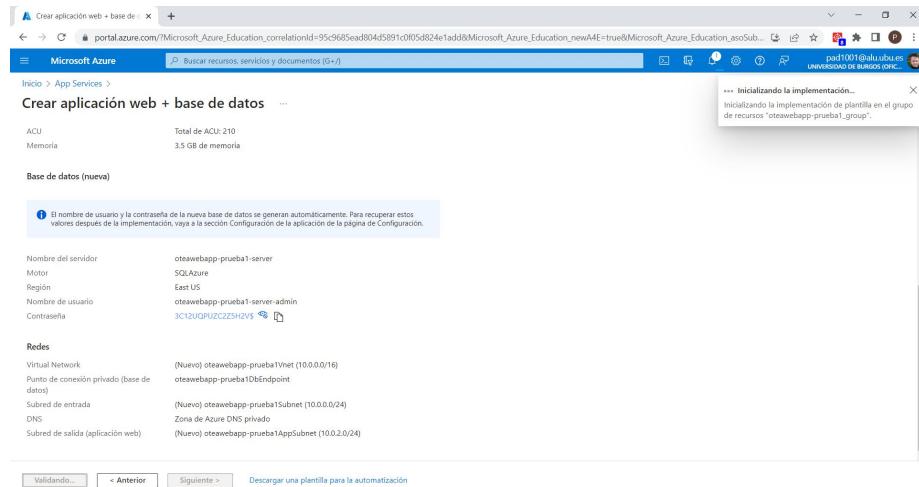


Figura D.55: La web app ya ha empezado a crearse

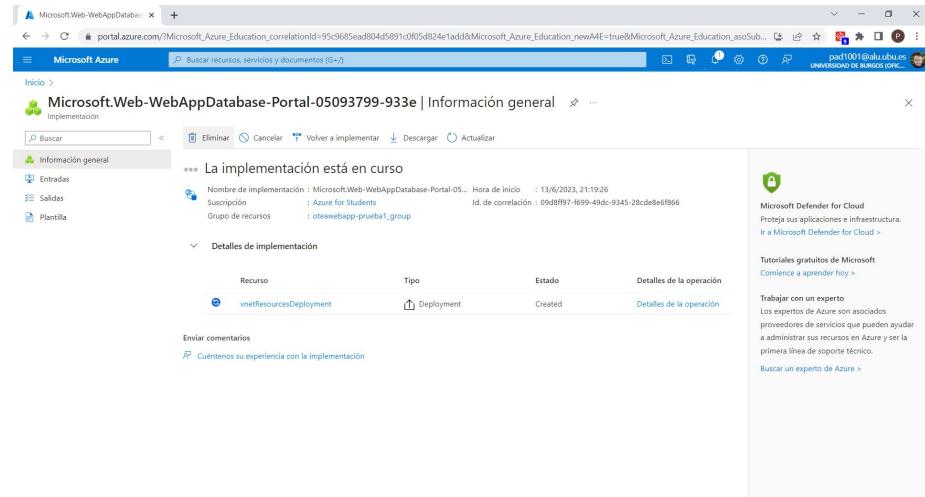
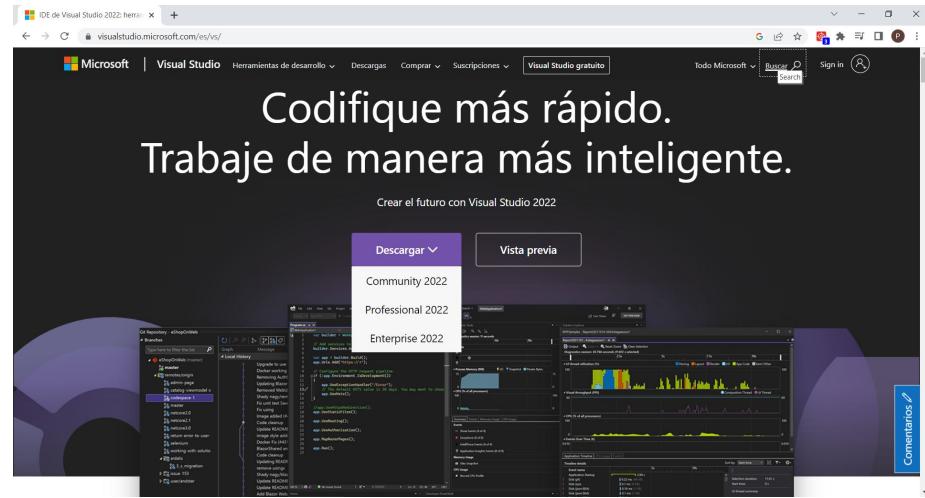


Figura D.56: Implementación en curso

Visual Studio 2022

Para poder instalar *Visual Studio 2022 Community*, hay que seguir los siguientes pasos:

1. En primer lugar se tiene que acceder a la página oficial de descarga de Microsoft y elegir la versión gratuita *Community 2022*. Cuando se ha elegido esa opción, se pasa a la descarga:

Figura D.57: Página oficial de descarga de *Visual Studio 2022*

2. Posteriormente se tiene que preparar el propio instalador, el cual bajará todas las características necesarias para el mismo

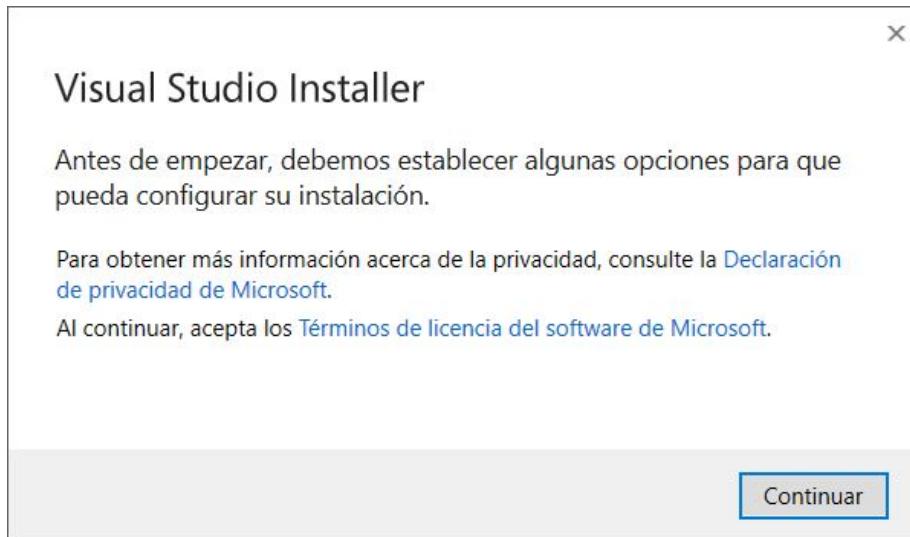


Figura D.58: Antes de empezar hay que preparar el instalador

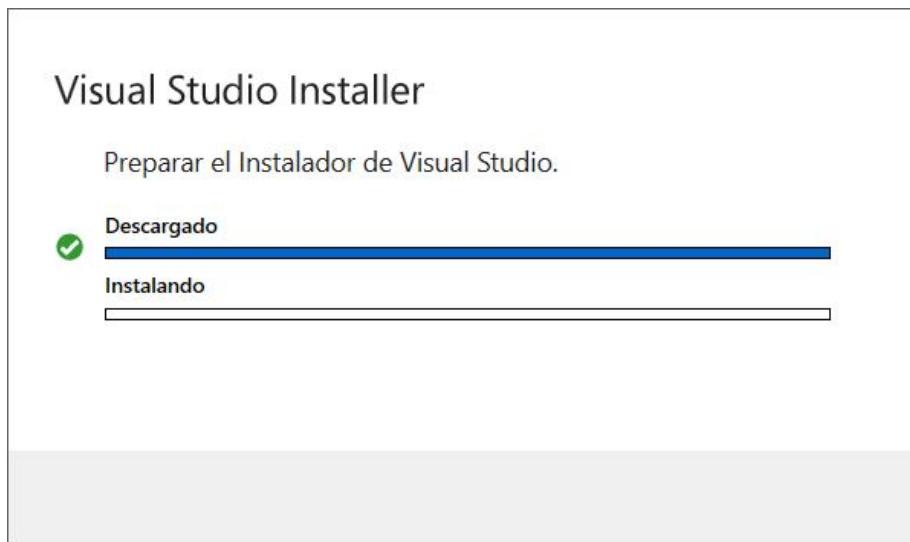


Figura D.59: Preparando el instalador...

3. Posteriormente se tienen que elegir las cargas de trabajo necesarias para nuestro caso. Aquí marcamos las opciones *Desarrollo de ASP.NET y web* y *Desarrollo de Azure*:

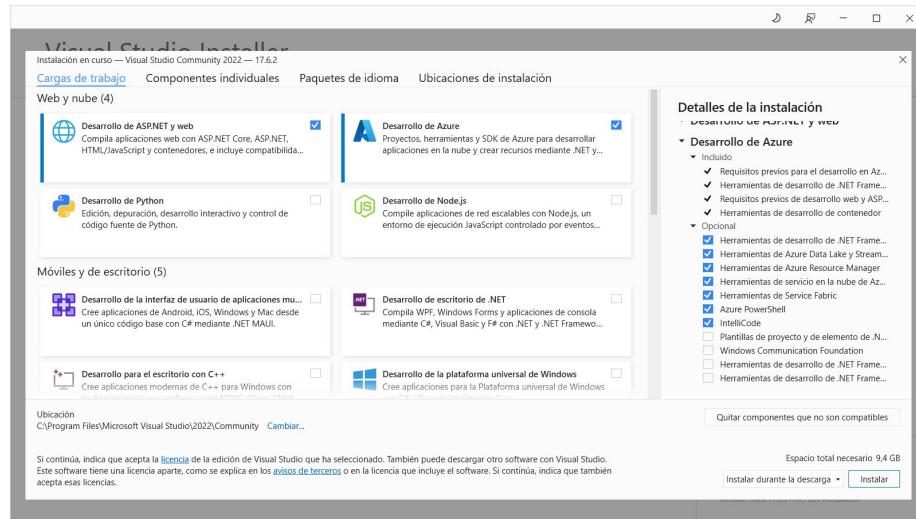


Figura D.60: Selección de las cargas de trabajo

4.

5. Posteriormente hay que esperar a que se instale todo:

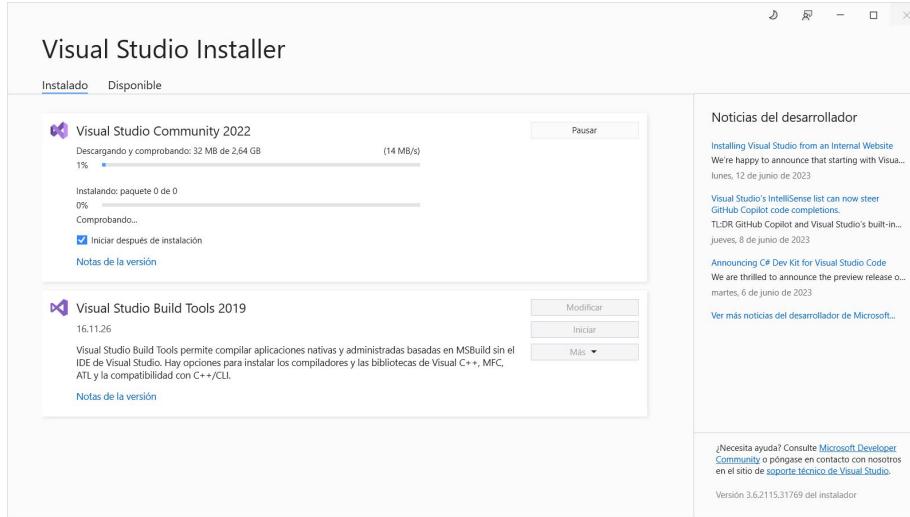


Figura D.61: Instalación en marcha

Cuando ya se ha terminado de configurar todo, se tiene que pasar a configurar el entorno correctamente:

1. En primer lugar se elegir una de las cuatro opciones existentes como tareas iniciales, en este caso se va a abrir un proyecto o una solución:

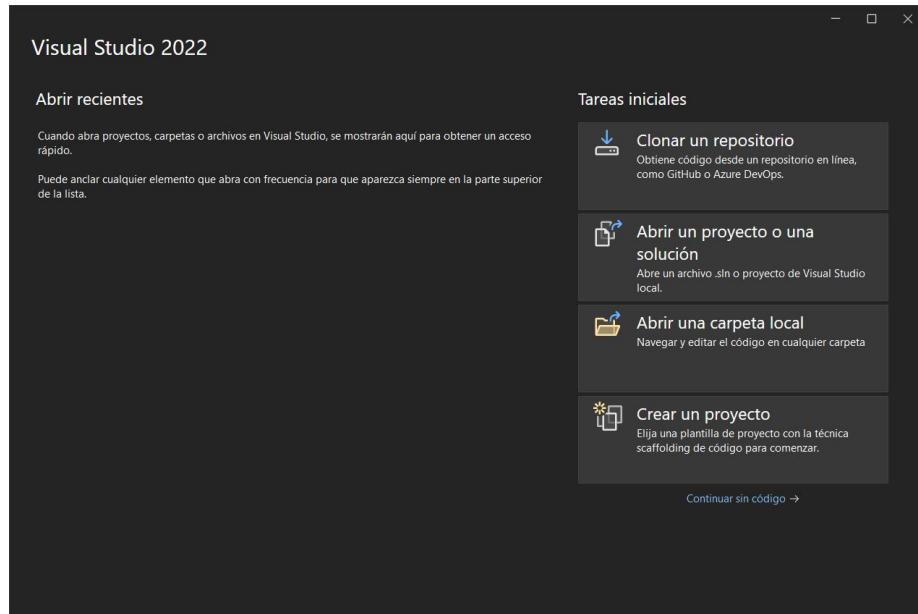


Figura D.62: Tareas iniciales

2. Posteriormente en el menú de selección de plantillas, tenemos que seleccionar *Aplicación web de ASP.NET Core*

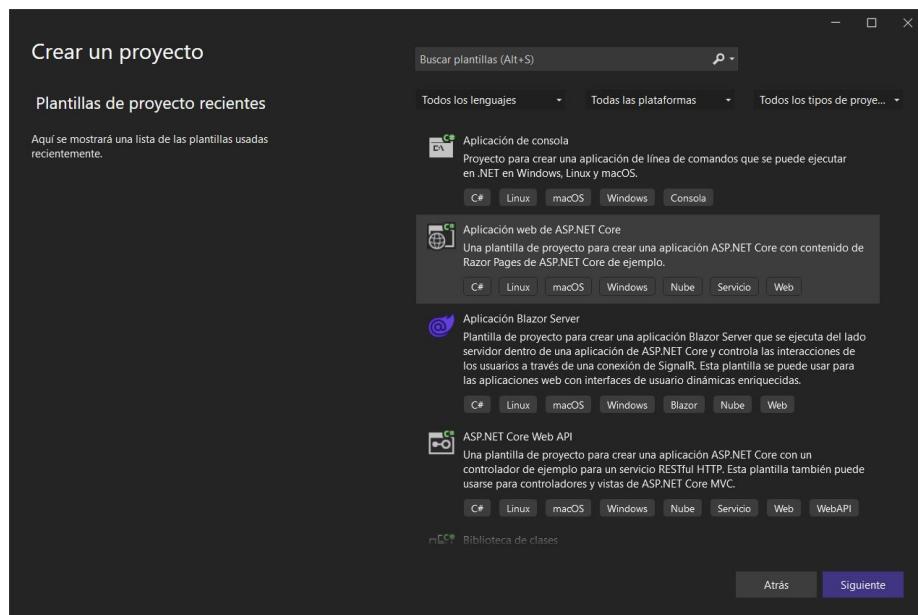


Figura D.63: Selección de plantilla

3. Más adelante se tiene que configurar el proyecto, eligiendo el nombre y el lugar en el que se va a guardar:

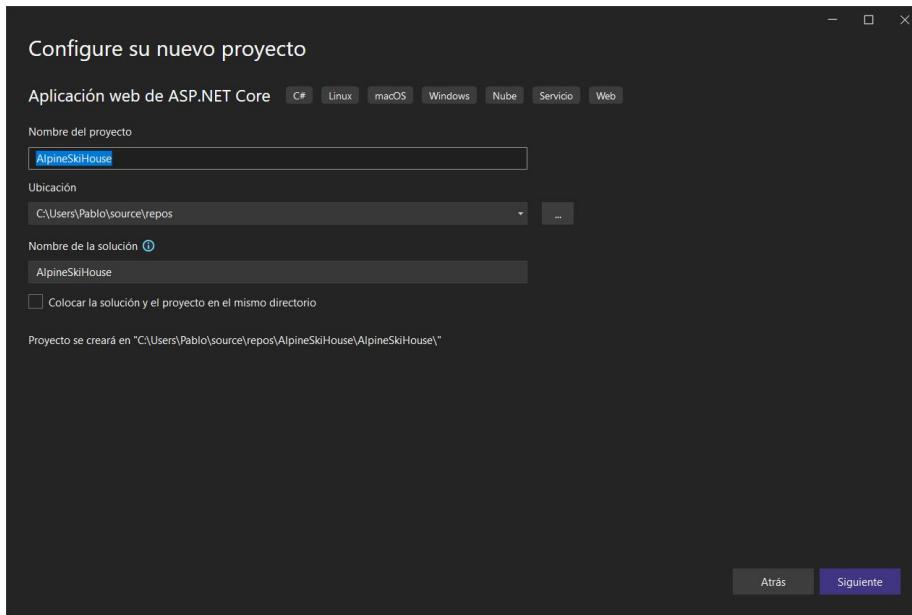


Figura D.64: Configuración de proyecto

4. Posteriormente se selecciona el framework .NET 6.0:

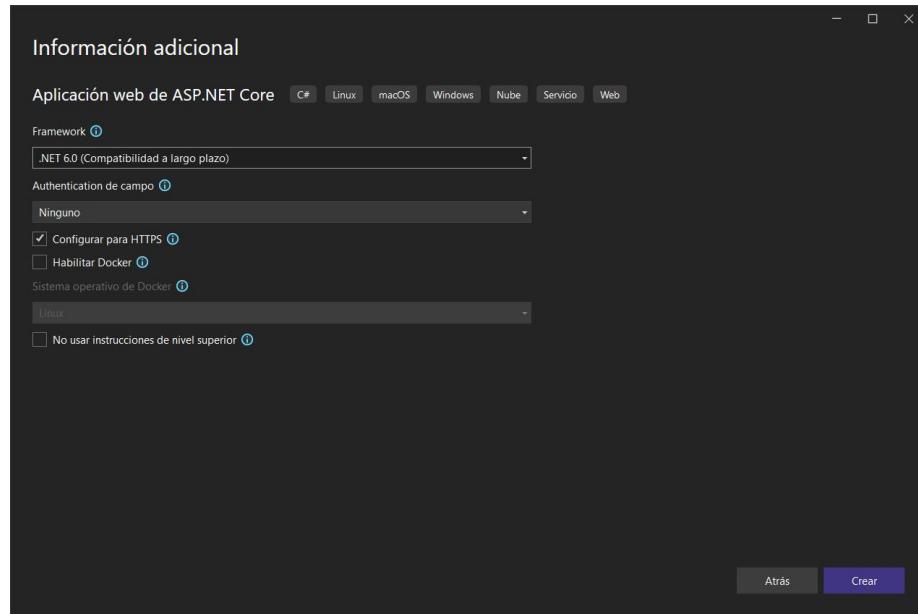


Figura D.65: Selección de framework

Por último, para actualizar los cambios en el servidor de Azure, se tienen que seguir los siguientes pasos:

1. En primer lugar hay que acceder al *Explorador de soluciones* para luego hacer click derecho sobre el nombre del proyecto. Más adelante le damos a *Publicar*

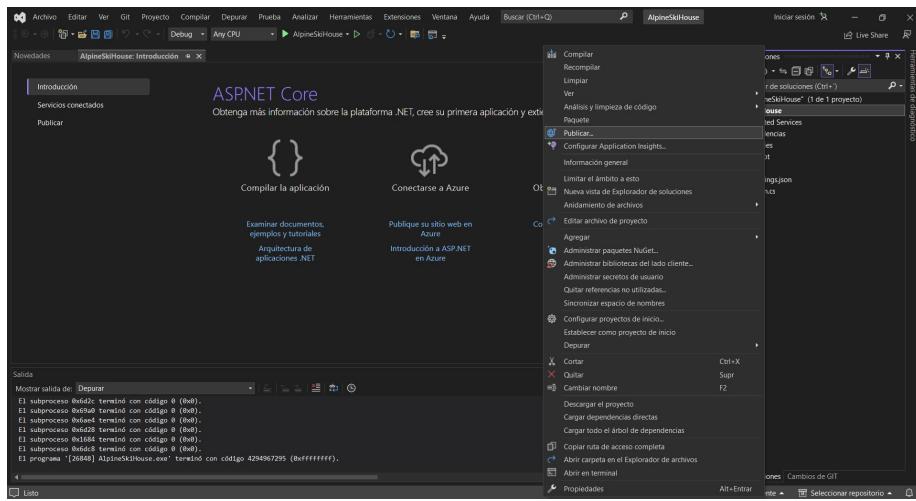


Figura D.66: Ir a publicar

- Como no se tiene ningún perfil de publicación, se va a proceder a hacer clic en *Agregar perfil de publicación*:

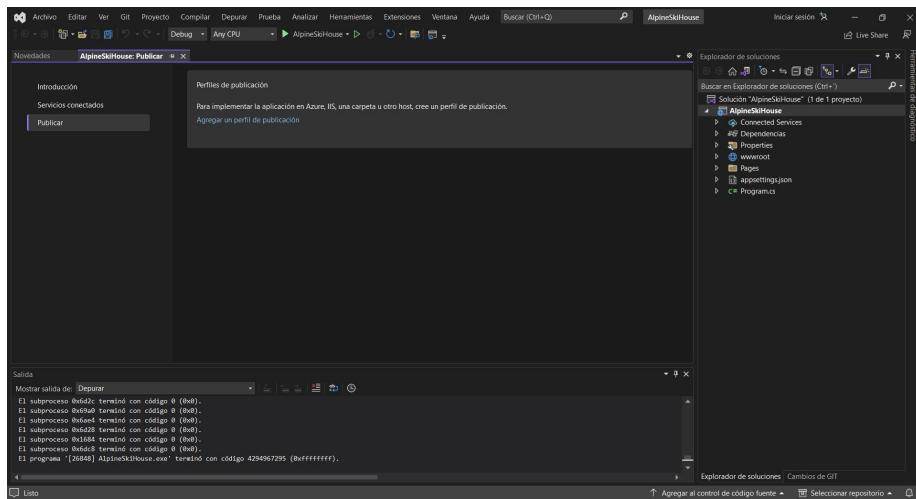


Figura D.67: Agregar perfil de publicación

- Posteriormente se selecciona Azure:

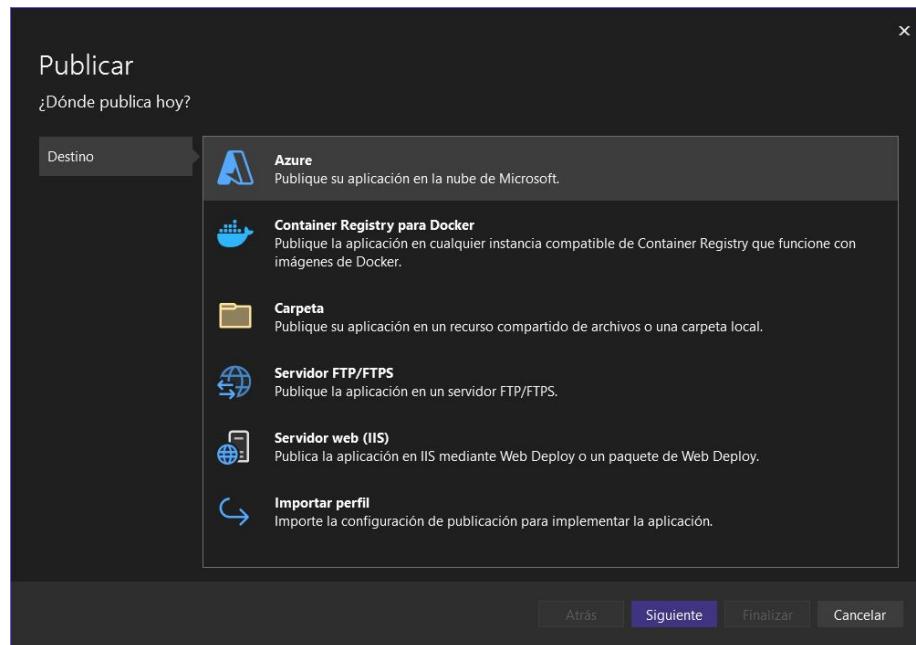


Figura D.68: Selección de publicación en Azure

4. Posteriormente se selecciona la web app de *Azure App Service*:
- 5.

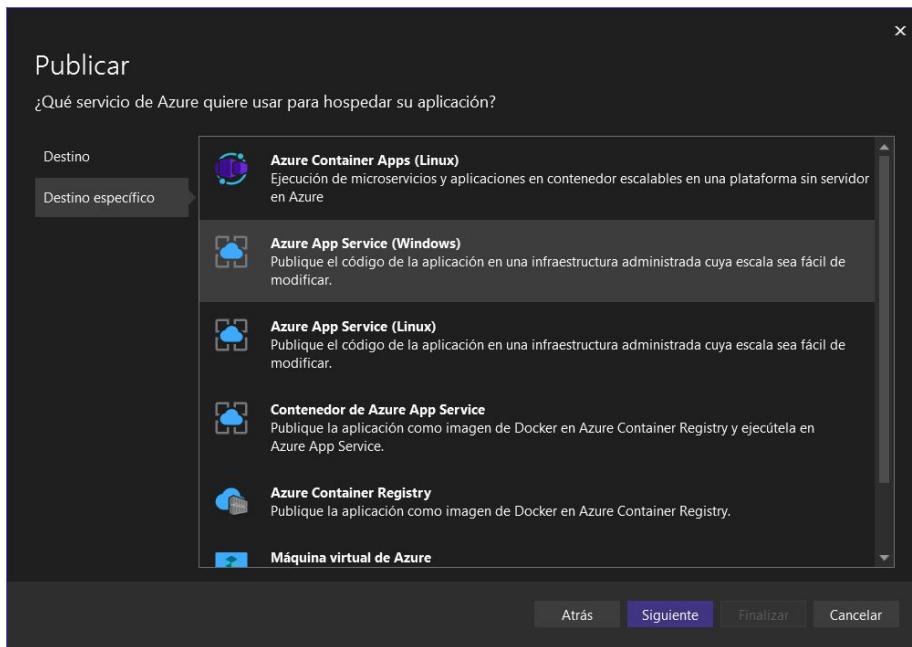


Figura D.69:

6. Tras haber iniciado sesión, se elige el servicio web donde se desea implementar la aplicación

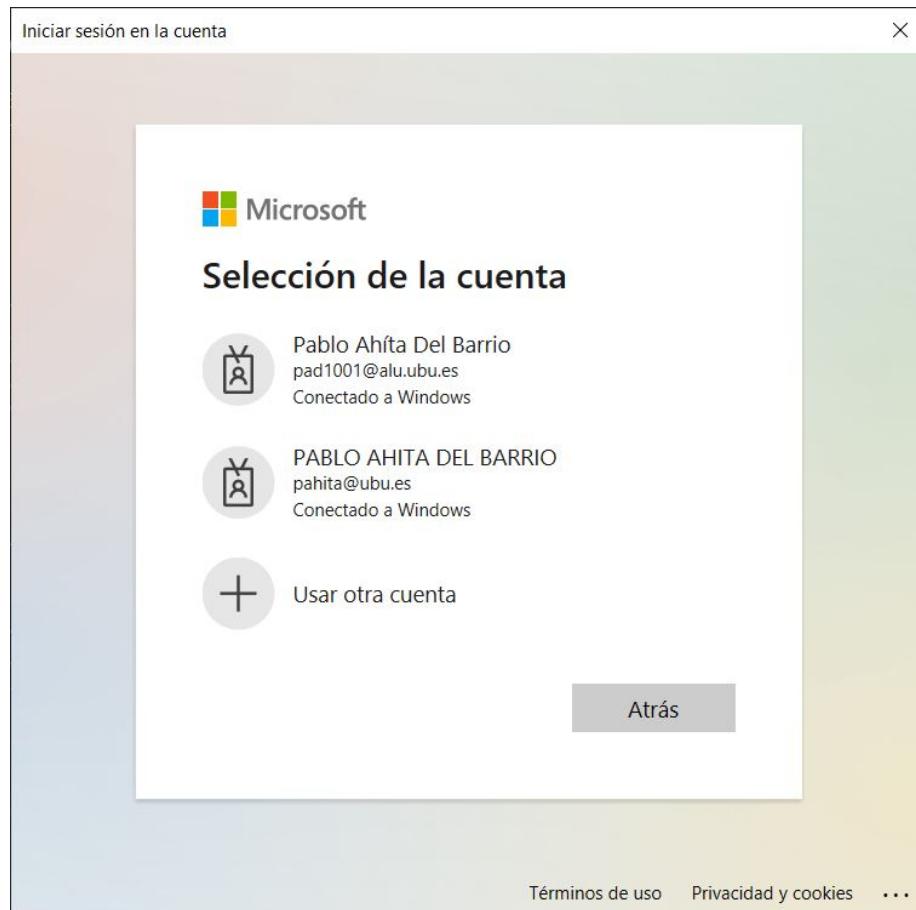


Figura D.70: Selección de web service

7. Cuando ya esté todo, ya se puede empezar a trabajar

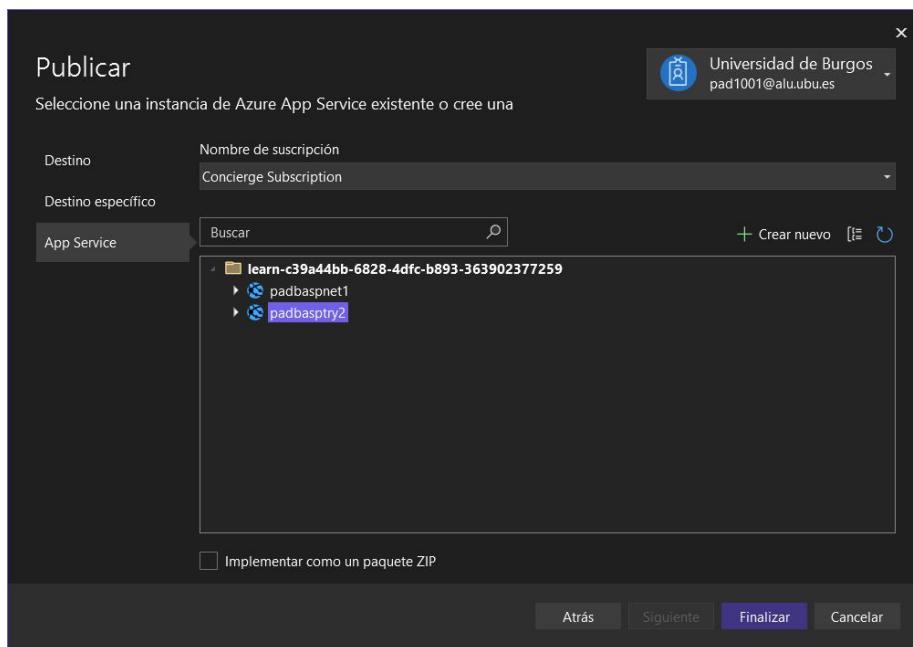


Figura D.71: Despliegue completado

Apéndice E

Documentación de usuario

E.1. Requisitos de usuarios

Para poder utilizar esta aplicación es necesario contar con cualquier móvil o tablet que sea mínimo del *API 33* de Android, ya que el dispositivo virtual en el que se ha probado tiene este sistema operativo. En cuanto a otros aspectos, no es una aplicación que requiera de otros recursos especiales puesto que es una aplicación bastante sencilla en cuanto a consumo de recursos.

E.2. Instalación

Para poder instalar la APK correctamente, hay que descargarla del repositorio de GitHub del proyecto final de grado, encontrándose en el directorio *app/build/outputs/apk/debug* del mismo, todo ello desde un dispositivo compatible. Antes de proceder a la instalación hay que activar los permisos de instalación necesarios para poder instalarlo, debido a que no se encuentra por ahora en *Google Play*. Cuando ya se han activado los permisos, se podrá instalar como ordena el dispositivo.

E.3. Manual del usuario

(Véanse vídeos de la carpeta de vídeos mencionada en el manual de programador)

Bibliografía

[1]

- [2] Google. Android studio, 2023. [Página de descarga del instalador de Android Studio].
- [3] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.
- [4] Wikipedia. Latex — wikipedia, la enciclopedia libre, 2015. [Internet; descargado 30-septiembre-2015].