



Lookttery

App de resultados de loterías con micro-frontales

Rubén García de Longoria Pulido

Grado de Ingeniería Informática

Desarrollo multiplataforma de aplicaciones móviles

Carles Sànchez Rosa

Jordi Almirall López

Carles Garrigues Olivella

Martes, 11 junio de 2024



Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada [3.0 España de Creative Commons](https://creativecommons.org/licenses/by-nc-nd/3.0/es/)

Licencias alternativas (elegir alguna de las siguientes y sustituir la de la página anterior)

A) Creative Commons:



Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada [3.0 España de Creative Commons](https://creativecommons.org/licenses/by-nc-nd/3.0/es/)



Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-CompartirIgual [3.0 España de Creative Commons](https://creativecommons.org/licenses/by-nc-sa/3.0/es/)



Esta obra está sujeta a una licencia de Reconocimiento-NoComercial [3.0 España de Creative Commons](https://creativecommons.org/licenses/by-nc/3.0/es/)



Esta obra está sujeta a una licencia de Reconocimiento-SinObraDerivada [3.0 España de Creative Commons](https://creativecommons.org/licenses/by-nd/3.0/es/)



Esta obra está sujeta a una licencia de Reconocimiento-CompartirIgual [3.0 España de Creative Commons](https://creativecommons.org/licenses/by-sa/3.0/es/)



Esta obra está sujeta a una licencia de Reconocimiento [3.0 España de Creative Commons](https://creativecommons.org/licenses/by/3.0/es/)

B) GNU Free Documentation License (GNU FDL)

Copyright © 2024 RUBEN-GARCIA_DE_LONGORIA-PULIDO.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

C) Copyright

© (el autor/a)

Reservados todos los derechos. Está prohibido la reproducción total o parcial de esta obra por cualquier medio o procedimiento, comprendidos la impresión, la reprografía, el microfilme, el tratamiento informático o cualquier otro sistema, así como la distribución de ejemplares mediante alquiler y préstamo, sin la autorización escrita del autor o de los límites que autorice la Ley de Propiedad Intelectual.

FICHA DEL TRABAJO FINAL

Título del trabajo:	<i>Lookttery – App de resultados de loterías con arquitectura de micro-frontales</i>
Nombre del autor:	<i>Rubén García de Longoria Pulido</i>
Nombre del consultor/a:	<i>Carles Sànchez y Jordi Almirall López</i>
Nombre del PRA:	<i>Carles Garrigues Olivella</i>
Fecha de entrega (mm/aaaa):	06/2024
Titulación:	<i>Grado de Ingeniería Informática</i>
Área del Trabajo Final:	<i>Desarrollo multiplataforma de aplicaciones móviles</i>
Idioma del trabajo:	<i>Castellano</i>
Palabras clave	<i>Micro-frontales, API Rest, Lotería, Angular, Capacitor, NestJS, Notificaciones Push, Native Federation</i>

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

Prototipo de arquitectura de micro-frontales basado en una App de resultados de loterías, con capacidad para ser actualizada sin pasar por las tiendas oficiales. Ofrece ventajas únicas respecto a tecnologías nativas o multiplataforma orientadas a arquitecturas monolíticas.

Abstract (in English, 250 words or less):

Prototype of a micro-frontend architecture based on a lottery results app, with the capability to be updated without going through official stores. It offers unique advantages over native or cross-platform technologies oriented towards monolithic architectures.

Índice

1. INTRODUCCIÓN	1
1.1 CONTEXTO Y JUSTIFICACIÓN DEL TRABAJO.....	1
1.2 OBJETIVOS DEL TRABAJO.....	1
1.3 ENFOQUE Y MÉTODO SEGUIDO.....	2
1.4 PLANIFICACIÓN DEL TRABAJO.....	2
1.5 BREVE SUMARIO DE PRODUCTOS OBTENIDOS.....	4
1.6 BREVE DESCRIPCIÓN DE LOS OTROS CAPÍTULOS DE LA MEMORIA.....	4
2. DISEÑO CENTRADO EN EL USUARIO	5
2.1 ANÁLISIS COMPETITIVO.....	5
2.2 PERFILES DE USUARIO.....	5
2.3 EVALUACIÓN.....	7
2.4 FLUJOS DE INTERACCIÓN.....	9
2.5 PROTOTIPO.....	10
3. DISEÑO TÉCNICO	15
3.1 DEFINICIÓN DE LOS CASOS DE USO.....	15
3.2 DIAGRAMA UML DE LA BASE DE DATOS Y ENTIDADES.....	19
3.3 DIAGRAMA EXPLICATIVO DE LA ARQUITECTURA.....	19
4. TECNOLOGÍAS UTILIZADAS	21
4.1 – CAPACITOR.....	21
4.2 – ANGULAR 17.....	22
4.3 – NATIVE FEDERATION.....	23
4.4 – IONIC 7.....	25
4.5 – NESTJS 3.....	26
4.6 – MONGODB.....	27
4.7 – GITHUB PAGES.....	28
4.8 – FIREBASE.....	29
4.9 – RAILWAY.....	30
5 – LOOKTTERY APP	31
5.1 – INTRODUCCIÓN.....	31
5.2 – MVP.....	31
5.3 – API REST.....	38
5.4 – PRUEBAS DE USUARIO Y DESPLIEGUE EN PRODUCCIÓN.....	40
5.5 – TEST UNITARIOS.....	41
6 – DESARROLLO Y EJECUCIÓN DEL PROYECTO EN LOCAL	42
6.1 – PROYECTO LOOKTTERY-API.....	42
6.1.1 – Ejecución en local.....	42
6.1.2 – Herramientas de debug.....	44
6.1.3 – Colección de endpoints.....	45
6.1.4 – Entidades de negocio.....	46
6.1.5 – Registro y login silencioso en una App sin usuarios.....	47
6.1.6 – Tarea Cron para conseguir datos de los sorteos.....	49
6.1.7 – Despliegue en producción.....	50
6.2 – PROYECTO LOOKTTERY-APP.....	51
6.2.1 – Ejecución en local.....	51
6.2.2 – Herramientas de debug.....	54
6.2.3 – Estructura del proyecto.....	55
6.2.4 – Exposición de rutas desde el micro-frontal.....	58

6.2.5 - Solución de problemas con interceptores y assets.....	59
6.2.6 – Despliegue en producción.....	60
6.3 – PROYECTO CORE-SHELL.....	66
6.3.1 – Ejecución en local.....	66
6.3.2 – Herramientas de debug.....	68
6.3.3 – Herramientas de debug nativas.	70
6.3.4 – Estructura del proyecto	71
6.3.5 – Despliegue en producción.....	74
6.4 – PUBLICACIÓN EN LAS TIENDAS OFICIALES	75
3. CONCLUSIONES	78
4. GLOSARIO.....	79
5. BIBLIOGRAFÍA	80
6. ANEXOS.....	81

Lista de figuras

Ilustración 1 - Diagrama de Gantt	3
Ilustración 2 - Prototipo 1	10
Ilustración 3 - Prototipo 2	11
Ilustración 4 - Prototipo 3	11
Ilustración 5 - Prototipo 4	12
Ilustración 6 - Prototipo 5	12
Ilustración 7 - Prototipo 6	13
Ilustración 8 - Prototipo 7	13
Ilustración 9 - Prototipo 8	14
Ilustración 10 - Diagrama UML Base de datos y Entidades	19
Ilustración 11 - Diagrama de arquitectura	20
Ilustración 12 - MongoDB Compass	27
Ilustración 13 - MongoDB Atlas Métricas	28
Ilustración 14 - Lookttery.com web	29
Ilustración 15 - Onboarding	32
Ilustración 16 - Pantalla inicial y detalle del sorteo	33
Ilustración 17 - Función para escanear décimos	34
Ilustración 18 - Listado de tickets	35
Ilustración 19 - Notificación Push	36
Ilustración 20 - Pantalla de configuración	37
Ilustración 21 - Sincronizar dispositivos	38
Ilustración 22 - Panel principal de Railway	39
Ilustración 23 - Logs de la API Rest	39
Ilustración 24 - Postman VSCode	40
Ilustración 25 - Test unitarios micro-frontal	41
Ilustración 26 - Cobertura test micro-frontal	41
Ilustración 27 - Lookttery-api instalación de dependencias	42
Ilustración 28 - Lookttery-api configuración de secretos	43
Ilustración 29 - Lookttery api ejecución en local	44
Ilustración 30 - Lookttery api debug	45
Ilustración 31 - Lookttery api colección de llamadas	46
Ilustración 32 - Diagrama de entidades	47
Ilustración 33 - Ejemplo entidad User	47
Ilustración 34 - Lookttery api decorador @Auth	48
Ilustración 35 - Lookttery api ejemplo de llamada @Auth y @GetUser	49
Ilustración 36 - - Lookttery api tarea Cron	50
Ilustración 37 - Lookttery api puesta en producción	51
Ilustración 38 - Lookttery app instalación de dependencias	52
Ilustración 39 - Lookttery app ejecución en local	53
Ilustración 40 - - Lookttery App ejecución en local sin API funcionando	54
Ilustración 41 - Lookttery app debug safari	54
Ilustración 42 - Lookttery app debug Chrome	55
Ilustración 43 - Lookttery app estructura principal	56
Ilustración 44 - Lookttery app explicación oficial del método bootstrapApplication	56
Ilustración 45 - Lookttery app error version ECMAScript	57
Ilustración 46 - Lookttery app rutas expuestas	58
Ilustración 47 - Lookttery app api service	59
Ilustración 48 - Lookttery app uso de assets	60
Ilustración 49 - Lookttery app scripts	60
Ilustración 50 - Lookttery app configuración de producción	61
Ilustración 51 - Lookttery app generación de ficheros para distribución	62
Ilustración 52 - Lookttery app configuración github-page	63
Ilustración 53 - Lookttery app fichero remoteEntry.json en el navegador	64
Ilustración 54 - Lookttery app como web	65
Ilustración 55 - Lookttery app escaneando desde la web	65
Ilustración 56 - Lookttery app ajustes desde la web, posibilidad de sincronizar	66

Ilustración 57 - Core shell instalación de dependencias	67
Ilustración 58 - Core shell sin conexión	68
Ilustración 59 - Core shell Chrome inspect desde emulador Android	69
Ilustración 60 - Core shell Chrome Dev Tools	70
Ilustración 61 - Core shell depurando plugin de notificaciones para ver token	71
Ilustración 62 - Core shell configuración del micro-frontal	71
Ilustración 63 - Core Shell configuración de Capacitor	72
Ilustración 64 - Core shell directorios Android y iOS	72
Ilustración 65 - Core shell fichero index.html	73
Ilustración 66 - Core shell definición de eventos y funcionalidades	73
Ilustración 67 - Core shell scripts prepare	74
Ilustración 68 - Core shell Android menú generate APK	74
Ilustración 69 - Core shell Generate Signed Bundle	75
Ilustración 70 - Core shell archive a build	75
Ilustración 71 - Lookttery en la App Store	76
Ilustración 72 - Lookttery portal de desarrollador de Apple	76
Ilustración 73 - Lookttery en la Play Store	77

1. Introducción

1.1 Contexto y justificación del Trabajo

La App **Lookttery** pretende marcar una diferencia respecto a otras Apps similares mediante funcionalidades propias. Además de aportar un valor en la lógica de negocio también pretende ser un prototipo reutilizable debido al uso de micro-frontales para la actualización en tiempo real de la lógica de negocio de la App sin necesidad de recompilar el código, generar de nuevo los ficheros nativos para finalmente tener que por las tiendas oficiales.

Actualmente existe muy poca documentación y ejemplos sencillos para el uso de micro-frontales en aplicaciones móviles. Con este proyecto se pretende utilizar una parte concreta los micro-frontales para conseguir mejorar drásticamente el desarrollo y mantenimiento habituales en este tipo de Software.

Actualmente, las Apps sobre loterías existentes, pueden agruparse en dos tipos según incluyan la posibilidad de realizar compras para participar en los sorteos o no. La App **Lookttery** pertenece al grupo de las que no permiten la venta de participaciones en los sorteos. Este tipo de Apps están ligadas a alguna administración de loterías física. TuLoterero, ScanLoteria y Loterías y apuestas del estado son las principales referencias en la Apple Store y la Play Store. Otras como por ejemplo Resuloto, son Apps consolidadas que solamente ofrece información de los distintos juegos y posibilidad de comprobar resultados.

En cualquier caso, solo ScanLoteria ofrece la posibilidad de un escáner centralizado que guarda los tickets, aunque está muy centrada en la venta de lotería y obliga al usuario a registrarse para utilizar casi todas las funciones. Por el contrario, Resuloto es una App global que muestra información de muchos sorteos de distintos países, sin necesidad de un registro tedioso, pero con una interfaz muy caótica y sin un escáner o posibilidad de comprobar juegos.

La App **Lookttery** pretende aunar lo mejor de estas Apps y, aunque en esta primera versión del MVP, solo aspira a trabajar con el sorteo de Lotería Nacional, el código se realizará de forma genérica y escalable para poder añadir juegos nuevos de forma rápida y sencilla, ya que aspira a tener información de sorteos de múltiples países y tener el único escáner global de reconocimiento de sorteos internacional.

1.2 Objetivos del Trabajo

Existen dos objetivos principales, el primero es construir App y la API REST necesaria, y el segundo es conseguir que el código de la App sea reutilizable a modo de prototipo para el desarrollo futuro de Apps basadas en micro-frontales.

Para lograrlo es necesario conseguir las siguientes metas específicas:

- Investigación previa de las tecnologías implicadas para trabajar con micro-frontales.
- Desarrollo de la API REST con la lógica de negocio básica.
- Desarrollo de una tarea automática capaz de recolectar la información externa necesaria en nuestra base de datos.
- Desarrollo de un proyecto base que será el proyecto que se compilará a nativo y capaz de acceder a las funcionalidades nativas de cada dispositivo.
- Desarrollo de un proyecto web básico, que será el micro-frontal cargado por el proyecto base y podrá utilizar las funcionalidades nativas del dispositivo a través del proyecto base.
- Desarrollar la lógica de negocio del MVP en el proyecto web.
- Desplegar el proyecto web en producción.
- Generar y publicar las Apps para iOS y Android.

La primera versión del MVP de la App incluirá las siguientes funcionalidades:

- Información de los sorteos realizados, posibilidad de acceder al detalle y comprobar resultados de forma manual.
- * Escáner de decimos, resguardos o tickets (en adelante tickets) que el usuario haya adquirido por su cuenta.
- * Almacenamiento de los tickets escaneados de tal forma que el usuario pueda consultarlos cuando sea necesario.
- Acceso sencillo a la App, no se requerirá registro manual de usuarios para evitar la pérdida de usuarios.
- * Posibilidad de clonar o sincronizar varios dispositivos mediante el escaneo de un código QR.
- Desarrollar una API REST propia y segura para realizar toda la lógica de necesaria.
- * Desarrollo de tareas automáticas que consulten fuentes externas de información para poblar la base de datos.

Las funcionalidades marcadas con un * al inicio son las que podrían marcar la diferencia además de ser consideradas críticas.

1.3 Enfoque y método seguido

No se parte de un producto previo, al tratar de utilizar unas tecnologías punteras como el uso de micro-frontales y, además, en aplicaciones móviles, no hay más alternativa que partir de cero.

Mi planteamiento se basa en mi experiencia previa, por ello sé que mejorar los tiempos a la hora de desarrollar y desplegar una App es algo ambicioso, pero a su vez muy beneficioso. Poder desarrollar un solo código fuente que pueda ser usado como Web y App, tener la flexibilidad de los micro-frontales pudiendo tener varios dispositivos apuntando a distintos entornos, sin la necesidad de tener conectada la App con un cable al ordenador de desarrollo, es algo que pienso aprovechar durante el desarrollo de la App y sé que me va a permitir adquirir un conocimiento muy valioso.

A su vez, pretendo construir una App de resultados de loterías que va a permitir a los usuarios que participan en estos sorteos disponer de una herramienta que les va a hacer más sencilla la comprobación de estos premios mediante el escaneo y guardado de tickets, incluso antes de la celebración del sorteo, para posteriormente consultar su listado de tickets y ver qué premio ha conseguido.

El método que trataré de seguir será lo más parecido a una metodología ágil, donde trabajaré sobre tareas priorizadas y con plazos razonables, con revisiones del código realizado posteriores y previas a las entregas.

Me ayudará de un tablero Kanban mediante Jira para organizarme.

1.4 Planificación del Trabajo

Mi intención es dedicar 4 horas en días laborables y 4 horas en días festivos. No obstante, puede que tenga que dedicar menos y recuperarlas después puntualmente.

Para la organización de tareas he utilizado [Jira](#), para la realización de esquemas, organigramas y gráficos similares usaré [Draw.io](#).

He representado cada épica de trabajo como una de las PEC a entregar y dentro de cada épica existen tareas principales que a su vez podrían contener subtareas dependiendo de los bugs encontrados o para organizar mejor algún detalle de cada una de ellas.

Lookttery	start	end	404h
PEC1 - Plan de trabajo	01/03/24	12/03/24	45h
Brainstorming	01/03	01/03	4
Priorización de ideas	02/03	02/03	4
Definición del alcance del producto	03/03	04/03	8
Pruebas de concepto	05/03	06/03	8
Recopilación de información y recurs...	07/03	07/03	4
Organización y metodología de traba...	08/03	08/03	4
Planificación final	09/03	10/03	8
Generar App Hola Mundo	11/03	11/03	4
Entrega PEC1	12/03	12/03	1
PEC2 - Diseño y arquitectura	13/03/24	09/04/24	105h
Análisis según perspectiva de usuario	13/03	19/03	24
Diseño conceptual	20/03	28/03	36
Prototipado	29/03	08/04	44
Entrega PEC2	09/04	09/04	1
PEC3 - Implementación	10/04/24	21/05/24	165h
Diseño de la arquitectura principal	10/04	10/04	4
Desarrollo básico de la API en local	11/04	11/04	4
Desarrollo de los endpoints sobre la ...	12/04	12/04	4
Desarrollo de las tareas automáticas	13/04	13/04	4
Despliegue de la API en producción	14/04	14/04	4
Pruebas de la API en producción	15/04	15/04	4
Desarrollo básico del proyecto base	16/04	17/04	8
Desarrollo basico del proyecto web	18/04	19/04	8
Desarrollo final del proyecto base	20/04	23/04	16
Desarrollo final del proyecto web	24/04	01/05	32
Pruebas, QA y tiempo para solución ...	02/05	09/05	32
Publicar el proyecto web en producci...	10/05	10/05	4
Nativos a pruebas y testing	11/05	14/05	16
Pruebas finales, QA final y tiempo pa...	15/05	20/05	24
Entrega PEC3	21/05	21/05	1
PEC4 - Entrega final	22/05/24	11/06/24	89h
Preparación de la presentación final	22/05	23/05	16
Grabación y montaje de la presentaci...	24/05	28/05	20
Revisión de la memoria final	29/05	10/06	52
Entrega PEC4	11/06	11/06	1

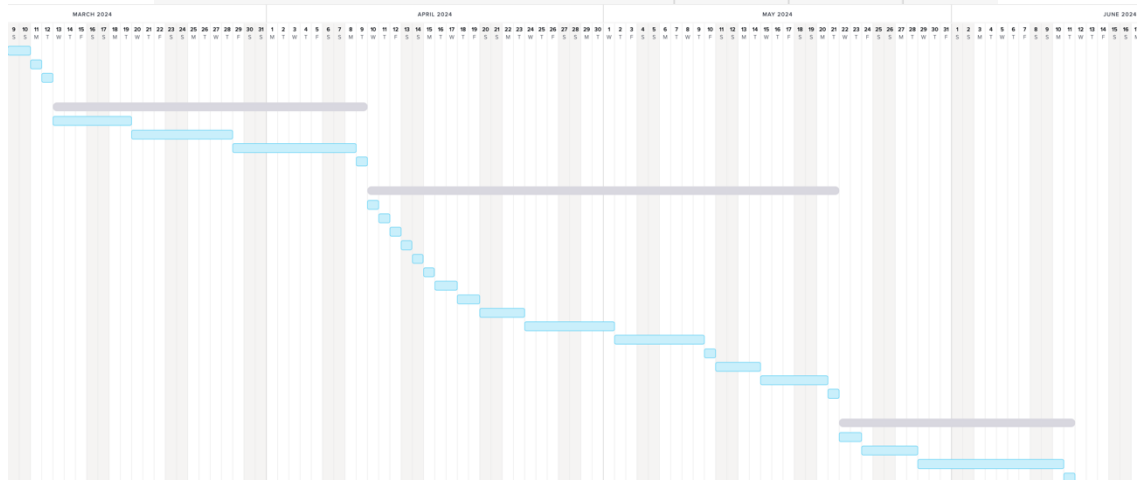


Ilustración 1 - Diagrama de Gantt

1.5 Breve resumen de productos obtenidos

En este caso los productos obtenidos serían todos los que componen el ecosistema necesario para que la App cumpla su objetivo, además de lo proyecto base y el proyecto web que funciona como micro-frontal, que pretenden ser un prototipo de partida para futuros proyectos.

- Los ficheros necesarios para publicar la versión MVP de la App en las tiendas de iOS y Android.
- Proyecto base basado en Angular 17 + Capacitor que contendrá la lógica necesaria para exponer las funcionalidades nativas mediante eventos.
- Proyecto web basado en Angular 17 que contendrá la lógica de negocio de la App y que será el micro-frontal cargado por el proyecto base.
- Proyecto API REST basado en NestJS que contendrá la lógica de servidor necesaria para el funcionamiento de la App. Además, será la capa encargada de manipular la base de datos y poblarla con información real mediante una tarea recurrente.
- Base de datos MongoDB con la estructura básica de la App. Esta base de datos se crea directamente desde el proyecto de API REST por lo que no será un producto entregable como tal.

1.6 Breve descripción de los otros capítulos de la memoria

A continuación, se detallará la información relevante del proyecto con distintos enfoques. Se trata el aspecto del diseño, centrado en el usuario y técnico. Explicación de cada tecnología principal que ha sido utilizada, motivos por los que se ha decidido usar frente a otras alternativas, inconvenientes encontrados y soluciones aplicadas durante el desarrollo. Presentación de la funcionalidad de la App, centrada en la lógica de negocio, motivaciones para realizar esta App, funcionalidades que se pretenden conseguir en la MVP, explicación de la API Rest y realización de pruebas y test. Un apartado centrado en la arquitectura de micro-frontales, y su aprovechamiento para conseguir la finalidad principal de publicar cambios en tiempo real, que en un desarrollo tradicional implicarían una versión nueva y pasar por las tiendas oficiales, y todo el tiempo y esfuerzo que esto implica. Este es uno de los apartados más importantes, donde se explica con más detalle y de forma técnica cada proyecto implicado, su puesta en marcha y como desarrollar en este tipo de arquitectura.

2. Diseño centrado en el usuario

El diseño centrado en el usuario, en adelante DCU, pretende cubrir todas las funcionalidades necesarias para el usuario. Para conseguirlo es necesario conocer el perfil del usuario, identificar correctamente el contexto de uso.

Las técnicas de investigación que van a utilizarse en el DCU para la App **Lookttery** son las siguientes:

- Análisis competitivo: Definición de una primera aproximación o punto de partida, mediante la comparación de otras Apps similares.
- Entrevistas a usuarios: Recopilación de información de usuarios de distintos perfiles.
- Shadowing: Recopilación de información mediante un seguimiento discreto de los hábitos diarios de los usuarios para conseguir información adicional de los usuarios.

2.1 Análisis competitivo

En el apartado *1.1 Contexto y justificación del trabajo* de la anterior PEC1 expliqué de forma resumida las motivaciones de realizar la App **Lookttery** y algunas funcionalidades de la competencia existente. Partiendo de esta información y, después de realizar un análisis en mayor profundidad de la competencia se han obtenido los siguientes puntos de partida:

- El usuario debe ser capaz de usar la App sin necesidad de registros y tramites tediosos e innecesarios.
- El procedimiento de escaneo de tickets debe realizarse de manera continuada y sin requerir más de una o dos acciones de usuario.
- Los datos de los sorteos realizados deben mostrarse al iniciar la App con la información más relevante en el menor espacio posible.
- Cada sorteo realizado debe ofrecer la posibilidad de ver toda la información accediendo desde su detalle.
- Los tickets escaneados por el usuario deben mostrarse por orden cronológico.
- La App debe estar traducida en todos los idiomas considerados de interés para los usuarios.
- La App solo debe ofrecer los formularios imprescindibles, la finalidad principal es la de consulta de información por parte del usuario.
- La funcionalidad de sincronizar varios dispositivos debe ser sencilla e intuitiva, por lo que se realizará mediante el escaneo de un código QR.
- La privacidad de los usuarios debe ser máxima y el usuario debe conocer que está en un sitio seguro y anónimo.
- Las funcionalidades principales deben ser accesibles en todo momento desde una zona reservada en la pantalla como un menú al píe.


Se han realizado entrevistas a usuarios con las que se pretende conseguir información sobre las necesidades principales, además de aplicar la técnica de shadowing y la realización de sketches para el diseño de la aplicación. Esta información puede verse en el apartado "1. Diseño centrado en el usuario", del anexo de esta memoria.

2.2 Perfiles de usuario

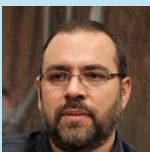
Según los datos obtenidos mediante las técnicas de investigación anteriores podemos realizar los siguientes perfiles de usuario.

Usuario en activo y estudiando sin hijos	Personalidad <ul style="list-style-type: none">- Utiliza la tecnología a diario- Dedicar su tiempo libre al ocio- Prefiere teletrabajar- Hace deporte habitualmente
---	---

	
Demografía <ul style="list-style-type: none"> - 25 a 34 años - Soltera - Trabaja y estudia - Independizada - Sin hijos 	Necesidades y objetivos <ul style="list-style-type: none"> - Crecer en su empresa - Terminar de formarse - Ganar competencias - Viajar

Usuario en activo sin hijos 	Personalidad <ul style="list-style-type: none"> - Utiliza la tecnología a diario - Realiza voluntariado - Prefiere espacios rurales - Es metódico
Demografía <ul style="list-style-type: none"> - 25 a 44 años - Soltero - Trabaja - Independizado - Sin hijos 	Necesidades y objetivos <ul style="list-style-type: none"> - Montar su propio huerto - Mejorar su comunidad - Viajar - Cambiar su trabajo actual

Usuario con hijos desempleado 	Personalidad <ul style="list-style-type: none"> - Dedicada a su familia - Ideales conservadores - No usa mucho la tecnología - Lee muchos libros
Demografía <ul style="list-style-type: none"> - 45 a 54 años - Casada - Desempleada - Independizada - Más de 3 hijos 	Necesidades y objetivos <ul style="list-style-type: none"> - Gestiona cuentas familiares - Club de lectura - Educación de sus hijos - Cambiar su trabajo actual

Usuario en activo con hijos 	Personalidad <ul style="list-style-type: none"> - Lidera sus proyectos - Intelectual - Usa nuevas tecnologías - Lee muchos libros
Demografía <ul style="list-style-type: none"> - 45 a 54 años 	Necesidades y objetivos <ul style="list-style-type: none"> - Hacer crecer su negocio

<ul style="list-style-type: none"> - Casado - Trabaja - Independizado - 1 hijo 	<ul style="list-style-type: none"> - Educación de su hijo - Montar en bici - Dar charlas tecnológicas
--	--

2.3 Evaluación

Para evaluar los diseños realizados se han realizado reuniones presenciales y telemáticas utilizando el prototipo y planteando las siguientes preguntas a parte de la opinión general:

- (Q-1) Has comprado una participación en un sorteo que aún no se ha realizado y quieres almacenarlo en tu móvil para consultar el premio cuando el sorteo sea realizado.
- (Q-2) Tienes varias participaciones antiguas en un cajón y quieres comprobar si alguna estaba premiada.
- (Q-3) Tienes un nuevo dispositivo móvil y quieres mover clonar los tickets que tienes escaneados en tu dispositivo antiguo.
- (Q-4) No tienes físicamente una participación, pero si sabes el número y quieres comprobar si tiene premio.

Una vez completadas estas tareas, cada usuario podrá plantear sus propias sugerencias y se le realizarán las siguientes preguntas en caso de que el usuario no haya comentado nada sobre ellas en sus sugerencias.

- ¿Te parece una App sencilla e intuitiva?
- ¿Qué mejorarías?
- ¿Recomendarías esta App?
- ¿Echas en falta alguna funcionalidad?

	Usuario en activo y estudiando sin hijos Compañera de Trabajo	
	Inputs positivos	Inputs negativos
Concepto general	Piensa que puede ser una App útil.	Sin comentarios.
Q-1	Le gusta la sencillez del proceso.	Le cuesta entender que se ha almacenado el ticket.
Q-2	Le parece un proceso muy rápido y eficiente.	Sin comentarios.
Q-3	Le parece una funcionalidad muy útil.	Le cuesta saber dónde está el código de usuario y cómo funciona la primera vez.
Q-4	Le parece un proceso muy sencillo y útil.	Sin comentarios.
Otros	Sin comentarios	<p>Le parece que la opción de clonar un dispositivo o es muy intuitiva y debería incluirse algún tipo de instrucciones.</p> <p>Mejoraría la interfaz para ser algo más clara y atractiva.</p> <p>Recomendaría esta App a compañeros del trabajo, pero echa en falta opciones para compartir mediante redes sociales.</p>

	Usuario en activo sin hijos Amigo
--	--

	Inputs positivos	Inputs negativos
Concepto general	Le parece buena idea.	Cree que le faltan funcionalidades adicionales que deberían sacarse en la siguiente versión.
Q-1	Le parece muy útil, sobre todo para no tener que llevar encima el resguardo en todo momento.	Sin comentarios
Q-2	Le parece un proceso muy rápido y que funciona muy bien.	Sin comentarios.
Q-3	Le parece una funcionalidad interesante también para compartir la cuenta con su pareja.	Le gustaría poder revertir este proceso.
Q-4	Le parece un proceso muy sencillo y útil.	Para resguardos muy antiguos le gustaría disponer de un calendario o de un buscador de sorteos.
Otros	Propone incluir publicidad para monetizar la App.	Cree que sería necesario un sistema que notifique al usuario cuando exista nueva información de sorteos que interesen al usuario o de sus propios tickets guardados.

	Usuario desempleado con hijos Amiga	
	Inputs positivos	Inputs negativos
Concepto general	Le parece una App que puede interesar a mucha gente.	Sin comentarios.
Q-1	Le parece muy útil, sobre todo para no tener que llevar encima el resguardo. en todo momento.	Sin comentarios.
Q-2	Le parece un proceso muy rápido y que funciona muy bien.	Le gustaría ver la información más animada si existe un premio.
Q-3	Sin comentarios.	Cree que no es una opción interesante.
Q-4	Le parece un proceso muy sencillo y útil.	Sin comentarios.
Otros	Sin comentarios.	Mejoraría el aspecto de la App.

	Usuario en activo con hijos Familiar	
	Inputs positivos	Inputs negativos
Concepto general	Cree que puede ser una App que use en su día a día	Sin comentarios.
Q-1	Sin comentarios	Sin comentarios.
Q-2	Le parece un proceso muy rápido y que funciona muy bien.	A veces le cuesta enfocar correctamente el código.
Q-3	Sin comentarios.	Sin comentarios.
Q-4	Le parece interesante ya que le gusta consultar números premiados de vez en cuando.	Sin comentarios.

Otros	Sin comentarios.	Mejoraría el aspecto de la App.
-------	------------------	---------------------------------

Después de las primeras pruebas, los usuarios han encontrado algunos puntos de mejora:

- Mejorar la apariencia de la App.
- Incluir sistemas de ayuda, tutoriales o guías para explicar el proceso de clonado.
- Animar la información de los premios.
- Notificaciones de nuevos sorteos y datos relevantes.

Se valorarán las siguientes propuestas, aunque inicialmente no se pretende incluirlas en el MVP:

- Calendario, filtros y buscadores de sorteos.
- Publicidad para monetizar la App

2.4 Flujos de interacción

A continuación, se muestra un diagrama con los principales flujos de interacción del usuario. Algunos de ellos son transparentes para el usuario, pero son relevantes para entender el flujo correctamente.

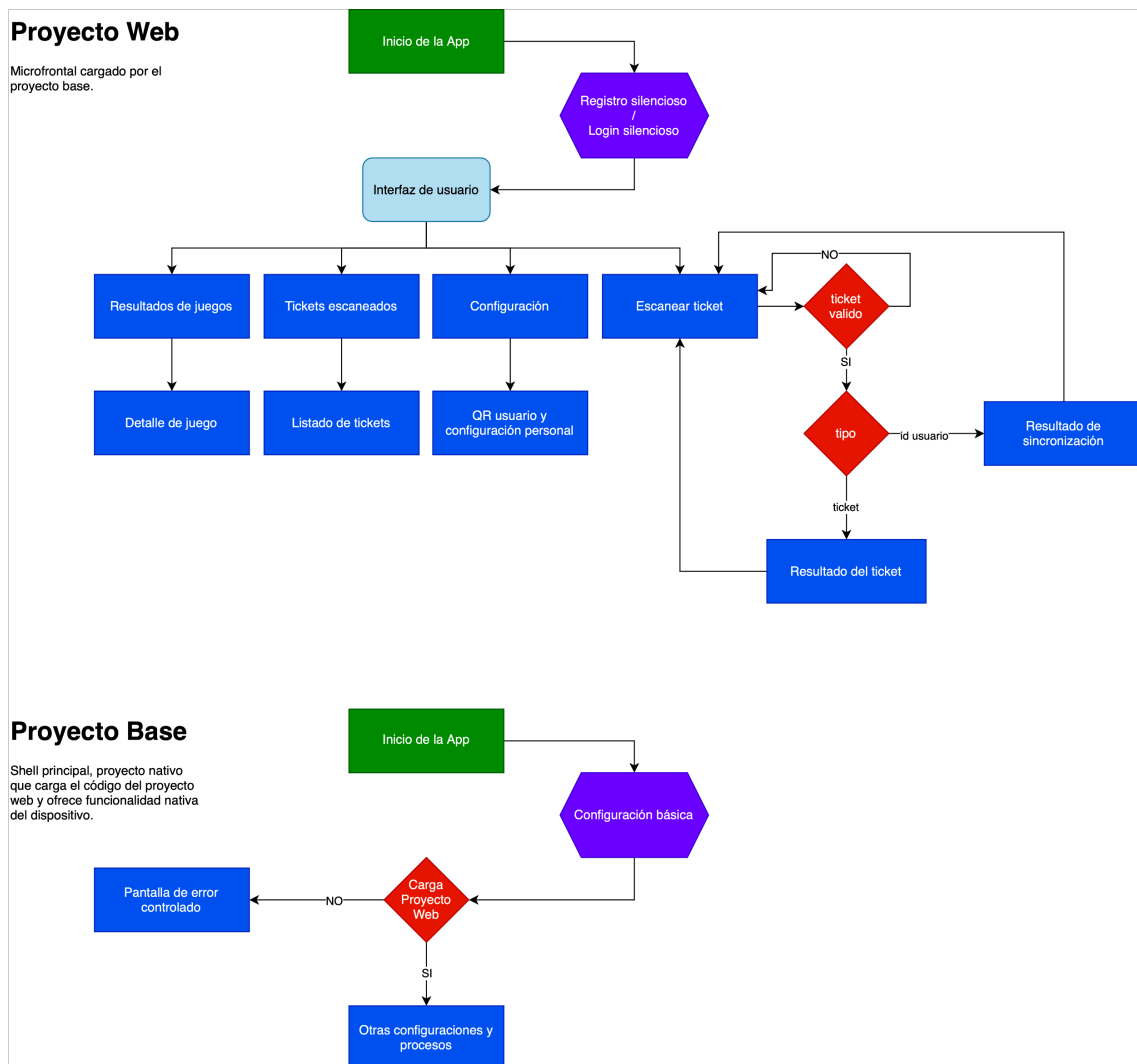


Ilustración 2 - Flujos de interacción

2.5 Prototipo

Se ha realizado un prototipo de alta fidelidad mediante la herramienta [Figma](#), el enlace público de acceso al prototipo es el siguiente:

<https://www.figma.com/proto/asATgl4OliJK20UNpIXRZB/Untitled?type=design&node-id=1-3&t=v3oz1cVIs7zqWctb-1&scaling=scale-down&page-id=0%3A1&starting-point-node-id=1%3A3&mode=design>

Como modelo para el diseño se ha tenido en cuenta un iPhone SE debido a que es uno de los modelos con menor tamaño de pantalla y considero que es mejor diseñar con el menor espacio posible inicialmente.

Se puede consultar información adicional sobre el prototipo en el apartado “2. Prototipo”, del anexo de esta memoria.

La pantalla inicial pretende ser totalmente informativa, inicialmente se mostrarán los distintos sorteos existentes en nuestra base de datos y su información más relevante. El listado cargará más resultados, si existen, cuando el usuario se desplace hacia abajo.



Ilustración 2 - Prototipo 1

El usuario puede interactuar con cada elemento de la lista, para acceder al detalle con toda la información del sorteo.



Ilustración 3 - Prototipo 2

Además, dependiendo del juego, se mostrará una interfaz que permita buscar un resultado concreto.



Ilustración 4 - Prototipo 3

Desde la opción “Escanear” se ofrecerá una interfaz sencilla abriendo la cámara en su totalidad y mostrando un puntero sencillo para ayudar al usuario a apuntar bien sobre el código de barras del ticket. Además, se muestran las opciones principales inferiores para poder salir de esta opción.



Ilustración 5 - Prototipo 4

Cuando se detecta un código de barras se muestran sus datos, este código puede ser correcto, incorrecto o de usuario. Si el código es correcto y tiene premio mostrará su premio, si todavía no tiene premio entonces solo mostrará sus datos y avisará de que es un ticket de una fecha sin sorteo realizado. Si el ticket no se reconoce avisará al usuario mediante un mensaje simple.



Ilustración 6 - Prototipo 5

Si el usuario escanea un código de otro dispositivo mostrará un mensaje del estado de la sincronización de los datos.



Ilustración 7 - Prototipo 6

Mediante la opción de "Tickets", el usuario accederá a un listado con todos los tickets escaneados, y podrá ver los premios de cada uno de ellos si el sorteo ha sido realizado. Si el sorteo ha sido realizado y el usuario toca el ticket será enviado a la pantalla de detalle del sorteo.



Ilustración 8 - Prototipo 7

Desde la opción de "Ajustes" el usuario accede a los parámetros de configuración de la App, donde aparecerán el idioma y el código del dispositivo.



Ilustración 9 - Prototipo 8

3. Diseño técnico

3.1 Definición de los casos de uso

Los casos de uso aquí explicados trabajan sobre una API que también se desarrollará y que será segura mediante la necesidad de un token de usuario, por lo que para usar la App en todo momento se necesitará estar registrado, salvo para registrar un usuario y hacer login que son acciones transparentes para el usuario.

Identificador	CU-001
Nombre	Registro silencioso
Prioridad	Alta
Descripción	El usuario necesita registrarse en la App para poder vincular su información a su usuario, pero debe hacerse de forma transparente, sin formularios, a nivel de dispositivo.
Actores	Usuario
Precondiciones	Ninguna
Iniciado por	Usuario
Flujo	1. Abrir la App
Postcondiciones	Internamente se crea el nuevo usuario basado en el UUID del dispositivo.
Notas	<p>El UUID es único por instalación, si el usuario borra completamente la App y la vuelve a instalar el UUID cambia. Se acepta ese inconveniente en favor de todas las ventajas de no molestar al usuario con registros con formularios, tampoco queremos trabajar con demasiada información personal de los usuarios.</p> <p>Al usuario se le proporcionará un código QR que podrá guardar para recuperar su información sin problemas.</p>

Identificador	CU-002
Nombre	Login silencioso
Prioridad	Alta
Descripción	El usuario registrado necesita logarse para recuperar su información.
Actores	Usuario
Precondiciones	Estar registrado (CU-001)
Iniciado por	Usuario
Flujo	1. Abrir la App
Postcondiciones	Si el UUID existe en la base de datos se recuperan sus datos.

Notas	-
-------	---

Identificador	CU-003
Nombre	Mostrar información de sorteos
Prioridad	Alta
Descripción	Estar registrado (CU-001) El usuario tiene que poder consultar la información de los sorteos realizados.
Actores	Usuario
Precondiciones	Estar registrado (CU-001)
Iniciado por	Usuario
Flujo	1. Abrir la App 2. Acceder a la opción de "Juegos", opción por defecto
Postcondiciones	Se muestran los sorteos realizados por orden de fecha de realización descendente. Los más nuevos primero.
Notas	Inicialmente se trabajará solo con el sorteo de Lotería Nacional, pero se pretende incluir muchos otros juegos por lo que la estructura de la información debe ser genérica y escalable.

Identificador	CU-004
Nombre	Escáner de códigos
Prioridad	Alta
Descripción	El usuario debe poder escanear códigos mediante la App
Actores	Usuario
Precondiciones	Estar registrado (CU-001) Haber aceptado los permisos necesarios
Iniciado por	Usuario
Flujo	1. Abrir la App 2. Acceder a la opción de "Escáner"
Postcondiciones	La primera vez se solicitan los permisos necesarios, si se aceptan se abre la cámara para escanear códigos.
Notas	Si no se aceptasen los permisos o el dispositivo no tuviese cámara no se podrá usar esta funcionalidad.

Identificador	CU-005
Nombre	Recuperar tickets escaneados
Prioridad	Alta

Descripción	El usuario debe poder recuperar los tickets que ha escaneado.
Actores	Usuario
Precondiciones	Estar registrado (CU-001)
Iniciado por	Usuario
Flujo	1. Abrir la App 2. Acceder a la opción de "Tickets"
Postcondiciones	Si el usuario tiene tickets almacenados en la base de datos se recuperan y se muestran.
Notas	-

Identificador	CU-006
Nombre	Eliminar tickets escaneados
Prioridad	Alta
Descripción	El usuario debe poder eliminar los tickets que ha escaneado.
Actores	Usuario
Precondiciones	Estar registrado (CU-001)
Iniciado por	Usuario
Flujo	1. Abrir la App 2. Acceder a la opción de "Tickets" 3. Tocar el icono de borrar 4. Aceptar la pregunta de confirmación
Postcondiciones	El ticket será eliminado de la base de datos y desaparecerá de la pantalla.
Notas	-

Identificador	CU-007
Nombre	Escanear un ticket
Prioridad	Alta
Descripción	El usuario debe recibir información cada vez que escanea un ticket.
Actores	Usuario
Precondiciones	Estar registrado (CU-001)
Iniciado por	Usuario
Flujo	1. Abrir la App 2. Acceder a la opción de "Escáner" 3. Escanear un ticket

Postcondiciones	Si el ticket es reconocido será almacenado en la base de datos y cruzado con los datos del sorteo que deberá existir en la base de datos. Se muestra al usuario la información encontrada.
Notas	Cada ticket escaneado es almacenado en la base de datos, si ya existe no se almacena de nuevo.

Identificador	CU-008
Nombre	Escanear un UUID
Prioridad	Normal
Descripción	El usuario debe recibir información cada vez que escanea un UUID.
Actores	Usuario
Precondiciones	Estar registrado (CU-001)
Iniciado por	Usuario
Flujo	1. Abrir la App 2. Acceder a la opción de “Escáner” 3. Escanear un UUID
Postcondiciones	Se asigna el nuevo UUID al usuario en la base de datos y se hace login de nuevo para sincronizar el dispositivo.
Notas	En adelante existirá más de un dispositivo con el mismo UUID y compartirán todos los tickets existentes y que sean escaneados posteriormente. Se ofrecerá una opción para que este proceso pueda ser revertido.

Identificador	CU-009
Nombre	Configuración de usuario
Prioridad	Alta
Descripción	El usuario debe tener un apartado donde configurar la App
Actores	Usuario
Precondiciones	Estar registrado (CU-001)
Iniciado por	Usuario
Flujo	1. Abrir la App 2. Acceder a la opción de “Configuración”
Postcondiciones	Se accede a un panel con distintas opciones de configuración, aquí estará el código con el UUID del dispositivo y al menos una opción de cambiar idioma.
Notas	-

3.2 Diagrama UML de la base de datos y entidades

Para este desarrollo se ha tomado la decisión de usar una base de datos no relacional, MongoDB, el principal motivo para ello es que la mayoría de las operaciones a realizar van a ser de lectura en vez de escritura, aunque también se ha tenido en cuenta que existen pocas entidades relevantes y relaciones sencillas entre ellas.

Hay que tener en cuenta que este tipo de bases de datos trabaja con colecciones de objetos JSON, en vez de usar relaciones y separar datos complejos en tablas, es común ver objetos y arrays de objetos formando parte de la estructura de la colección en la base de datos. A continuación, se muestra un diagrama UML donde se ven las colecciones necesarias en azul, que comparten la estructura con las entidades y, en rojo, se ven las estructuras internas de la entidad.

Las colecciones de la base de datos son idénticas a las entidades y clases que se utilizarán en el código ya que ambos son objetos JSON, por eso lo muestro todo en un solo diagrama.

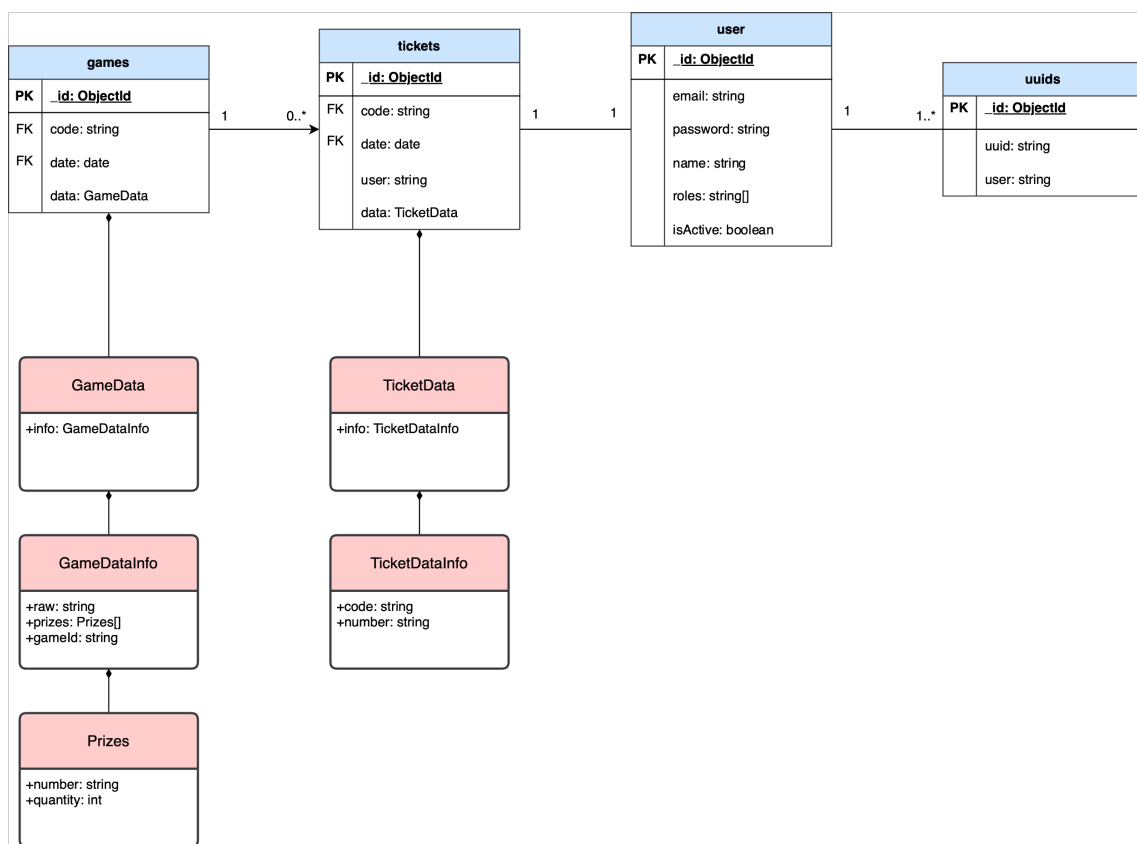


Ilustración 10 - Diagrama UML Base de datos y Entidades

3.3 Diagrama explicativo de la arquitectura

El diagrama de la arquitectura es bastante sencillo para el MVP.

La base de datos MongoDB se alojará en [MongoDB Atlas](#), que ofrece una capa gratuita más que suficiente y, en caso de crecer ofrece servicios profesionales a precios competitivos.

La API se alojará en [Railway](#), que ofrece un servicio muy completo y cómodo para alojar, desplegar y escalar aplicaciones basadas en Node. Existe una capa gratuita limitada. He valorado usar otros servicios con capas gratuitas más amplias como [f10](#), pero el problema es que las maquinas se apagan por inactividad y hay que habilitarlas manualmente, entre otros puntos negativos.

El proyecto Web (Micro-frontal con la lógica de negocio) se alojará en una [GitHub-Page](#), que ofrece alojamiento para webs estáticas. Se está valorando publicar el proyecto web además de la App, este alojamiento lo permitiría, pero realmente solo es necesario alojar el código compilado del proyecto.

El proyecto App será publicado en las principales tiendas, App Store y Play Store.

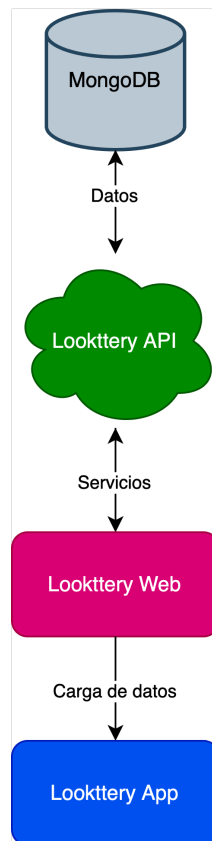


Ilustración 11 - Diagrama de arquitectura

4. Tecnologías utilizadas

En este apartado se presentan las tecnologías implicadas en todas las partes del desarrollo. En cada apartado se explica brevemente su función, motivos de usar esta tecnología y no otra, de qué forma se han utilizado para conseguir desarrollar con éxito la App y las complicaciones y limitaciones existentes.

Solo pretendo explicar las más importantes y las que no se consideran obligatorias en el desarrollo de aplicaciones multiplataforma de este estilo como el uso de Android Studio o Xcode.

4.1 – Capacitor



- <https://capacitorjs.com>
- <https://github.com/ionic-team/capacitor>

Es la pieza principal que nos permite conseguir Apps multiplataforma para Android y iOS a partir de un proyecto web realizado con HTML, JavaScript y CSS y, acceder a funcionalidades nativas del dispositivo físico desde este mismo proyecto web.

Capacitor es un proyecto de código abierto realizado por el equipo de Ionic. Inicialmente surgió como alternativa a Cordova, un producto que existía con anterioridad, desarrollado inicialmente por Nitobi, adquirido posteriormente por Adobe, el cual lo renombró como PhoneGap y a su vez liberó una versión de código abierto, con algunas pequeñas diferencias imperceptibles, que se conoce actualmente como Apache Cordova.

Aunque durante aquella época PhoneGap y Cordova se consideraban lo mismo lo cierto es que eran proyectos distintos y ofrecían distintas funcionalidades, PhoneGap fue famoso por su servicio online que permitía compilar en la nube tus proyectos para Android y iOS, incluso publicarlos en las tiendas oficiales, sin necesidad de tener el hardware necesario instalado en tu equipo.

Actualmente PhoneGap no tiene mantenimiento desde hace varios años y Apache Cordova, aunque se sigue usando para nuevos proyectos y existen muchos proyectos que funcionan sobre esta tecnología, solo hay que echar un vistazo a sus repositorios públicos para comprobar que cada se utiliza menos y recibe actualizaciones importantes con menos frecuencia.

La decisión de usar Capacitor es casi indiscutible, es la opción más adecuada por muchos motivos, además mi experiencia personal con ambas tecnologías me ha ayudado a tomar esta decisión.

La principal ventaja sobre Capacitor respecto a Apache Cordova es la filosofía de mantener los proyectos nativos abiertos a su modificación directa por parte del desarrollador. Con Apache Cordova, es necesario trabajar mediante scripts propios para modificar algunos ficheros nativos, ya que el recompilado de los proyectos nativos es habitual y, si se trabaja de forma directa, los cambios se pierden. Por ejemplo, a la hora de añadir un permiso nuevo en el

fichero manifest.xml del proyecto Android o en el fichero info.plist del proyecto iOS. En cambio, mediante Capacitor lo habitual es generar una única vez la estructura del proyecto nativo y solamente actualizar los ficheros del proyecto web cuando es necesario. Si se necesita añadir un permiso nuevo en el manifest.xml o en el fichero info.plist se hace directamente en el proyecto nativo.

Otra gran ventaja es la gran variedad de plugins existentes para trabajar con las funcionalidades del dispositivo físico. Existe una amplia comunidad que mantiene estas soluciones y funcionan bastante bien. Concretamente, para esta App he necesitado utilizar un plugin para escanear códigos de barras y, para Apache Cordova existe poca variedad y por lo general su mantenimiento está discontinuado y hay que utilizar un fork de algún particular, lo que no da mucha seguridad. Por el contrario, el plugin existente para Capacitor tiene una comunidad fuerte detrás.

También hay que tener en cuenta que Capacitor es muy fácil de añadir y eliminar de un proyecto web. Para el desarrollo de la App he utilizado un proyecto nuevo basado en Angular 17 al que le he incluido Capacitor. Toda la lógica de funcionalidades nativas existirá en este proyecto.

4.2 – Angular 17



- <https://angular.io>
- <https://github.com/angular/angular>

Para el desarrollo de la lógica de negocio de la App y todas las funcionalidades de los proyectos web implicados en el desarrollo de la App he decidido usar Angular en su versión más moderna hasta la fecha. He tenido la suerte de que ha coincidido en el tiempo una versión de Angular con grandes cambios respecto a versiones anteriores y he podido aprovechar grandes novedades como su sintaxis para directivas estructurales como el hecho de que ya no usa Webpack en favor del uso de Vite, lo que permite desarrollar de forma mucho más rápida ya que los tiempos de compilación son muy inferiores.

Aunque existen muchos frameworks para el desarrollo web, en este caso me he decantado por Angular debido a que me gusta su forma de estructurar los proyectos, actualmente mediante el uso de componentes standalone y sin la necesidad de usar módulos la estructura de los proyectos se simplifica aún más, ganando libertad, pero sin perder el orden de un framework robusto y testado en muchos proyectos.

También hay que reconocer que he estado un poco obligado a usar Angular es por el hecho de que Native Federation está pensado para Angular y, en caso de haber usado otro framework habría tenido que adaptar el código mediante Web Components y se me habría complicado mucho el desarrollo.

La App finalmente está formada por dos proyectos Angular 17, el primero es denominado core-shell, con la lógica genérica para aprovechar las funcionalidades nativas del dispositivo y el segundo es denominado lookttery-app y es el que contiene toda la lógica de negocio de la App.

4.3 – Native Federation



- <https://www.npmjs.com/package/@angular-architects/native-federation>
- <https://github.com/angular-architects/module-federation-plugin/blob/main/libs/native-federation/README.md>

Sin duda esta es la tecnología más puntera y especial utilizada en el desarrollo de la App. Es la evolución directa de la solución para trabajar con micro-frontales conocida como Module Federation, pero de forma “nativa para el navegador”, las comillas las ponen sus propios creadores en el texto que usan para explicar este proyecto. La gran diferencia entre Module Federation y Native Federation es la limitación de la versión de Angular a utilizar, para usar Angular 17 y superiores debe usarse Native Federation. Internamente otra gran diferencia es que deja de usarse Webpack.

Gracias a esta tecnología podemos instalar unas dependencias en nuestros proyectos Angular para otorgarles unas capacidades propias como un micro-frontal o un proyecto Shell, se entiende por proyecto Shell el proyecto base que va a orquestar todos los micro-frontales existentes.

Existen diferentes maneras de comunicar los micro-frontales entre sí, lo ideal es tener un diseño de nuestro producto eficiente y que los micro-frontales no necesiten comunicarse entre sí, simplemente enviar y recoger datos del servidor o trabajar con la Shell común. No obstante, todos los proyectos implicados comparten la misma memoria local y de sesiones, pueden definir y escuchar eventos como si fuesen un mismo proyecto web y pueden aprovechar Angular para compartir servicios entre sí.

Ambos proyectos pueden ser ejecutados de forma independiente y, el proyecto Shell, puede cargar cuando considere oportuno, en la carga inicial de la App, o cuando lo vaya a necesitar, el micro-frontal.

En el caso particular del desarrollo de la App, como he comentado con anterioridad, existen dos proyectos Angular 17, el denominado como core-shell será el encargado de orquestar los micro-frontales y, el denominado como lookttery-app, será un micro-frontal.

He decidido usar solo un único micro-frontal debido a que no necesito aprovechar toda la complejidad de los micro-frontales, mi proyecto no es lo suficientemente grande ni complejo, si usase varios micro-frontales solo añadiría problemas y dificultades al desarrollo. La justificación de usar esta tecnología es básicamente la potencia de poder cargar un proyecto cuando lo considere necesario. Esto se traduce en la posibilidad de cargar toda la lógica de mi App desde una dirección web cada vez que arranque mi aplicación, de esta forma puedo tener actualizada la lógica de negocio de mi App siempre que lo necesite sin pasar por las tiendas oficiales y sin necesidad de compilar los proyectos nativos y esperar por revisiones de Apple o de Google.

En cambio, sí que se debe compilar la App de nuevo y seguir el camino tradicional, subiendo la versión oficial de la App, cada vez que tenga que modificar mi proyecto Shell, por ejemplo, si

necesito añadir una nueva funcionalidad nativa, o compilar el proyecto para una nueva versión de iOS o Android.

Durante mi experiencia como desarrollador he sufrido (y sigo sufriendo) el cuello de botella de publicar una versión de una App cada vez que se quiere añadir una nueva sección, un nuevo filtro para un buscador o, simplemente, cambiar un color. En ocasiones he implementado soluciones específicas para proyectos concretos, por ejemplo un componente capaz de crear formularios a medida mediante un JSON para poder crear formularios de forma dinámica en tiempo real, o mediante el uso de XHTML, algo que se acercaban bastante a esta funcionalidad de actualizar contenido de la App en tiempo real, pero que necesitaba aun de unas etiquetas muy concretas en el proyecto y una rigidez importante, además de ser muy engorroso para lógicas complejas de JavaScript.

Ahora, mediante el proyecto base core-shell puedo tener un proyecto con lógica genérica para trabajar con funcionalidades nativas de dispositivos y cargar lógica de distintos negocios para crear algo parecido a una Super-App y ganar velocidad y libertad en mis desarrollos separando y encapsulando la parte compleja de una App de la lógica de negocio Web.

La parte negativa de esta tecnología y, de la que me he dado cuenta durante el desarrollo de la App, es la limitación a versiones recientes de ECMAScript ya que al usar imports dinámicos se requiere una versión ES2020 como mínimo, algo que aceptan todos los navegadores actuales.

En el caso de dispositivos Android he podido comprobar que es necesario actualizar Chrome para que la App funcione, por ejemplo, un dispositivo con Android 8, la versión mínima necesaria para usar Chrome en Android no carga correctamente el micro-frontal en su configuración de fábrica, ya que la versión de Android es inferior a la 94, versión a partir de la cual se aceptan imports dinámicos. Pero si se actualiza Chrome en el mismo dispositivo la App carga sin problemas. En el caso de iOS esta limitación va ligada a la versión del sistema operativo y es a partir de iOS 16.4 cuando se permiten los imports dinámicos.

Reconozco que estoy intentando buscar una solución por mi cuenta, intentando traducir la misma funcionalidad a ES2015, pero aún no he dado con ello.

Otro inconveniente menor que hay que tener en cuenta, relacionados con el uso que hago de esta tecnología, pero que esta más relacionado con mi decisión personal de exponer solo las rutas del micro-frontal, es la problemática de evitar repetir código en el proyecto Shell y en el proyecto micro-frontal.

Al exponer y cargar solo las rutas de ciertos componentes quedan al margen otras funcionalidades existentes fuera de la estructura de directorios expuesta, en este caso son los assets (imágenes y recursos multimedia) y los interceptores (en Angular 17 se usan de forma general fuera del directorio src y sin módulos).

La solución fácil a este problema es la de repetir el código en ambos proyectos, pero es justo lo que queremos evitar, por lo que la solución que yo he realizado es diferente para cada problema.

En el caso de los assets es necesario incluir el dominio del recurso siempre que este sea usado, en vez de usar una ruta relativa se debe incluir el dominio del entorno, en el caso de producción es <http://lookttery.com/> y en el caso de desarrollo puede ser <http://localhost:4201/> o <http://10.0.2.2:4201>, etc. Por lo que para ello existen ficheros de entorno donde se configura cada valor para una misma variable que se usa en el código y, a la hora de generar código, elegimos el entorno en el que queremos trabajar. Esta solución es similar a lo que ya sucede en la mayoría de los proyectos con la URL de la API de negocio, dependiendo del entorno suele cambiar y se usa una variable de entorno en el código.

En el caso de los interceptores he decidido crear un servicio propio, llamado ApiService, que hace de fachada sobre los verbos utilizados para llamar a la API (GET, POST, PUT, PATCH, DELETE...) de este modo puedo interceptar cada llamada e incluir o modificar lo necesario antes de llamar a la API. En este caso necesito incluir un token de autenticación en cada llamada, ya que mi API está protegida mediante autenticación en la mayoría de sus servicios.

Por otro lado, siempre incluyo el usuario en las cabeceras, de esta forma evito pasarlo como parámetro siempre y puedo ofrecer respuestas personalizadas de una forma más sencilla.

4.4 – Ionic 7



- <https://ionicframework.com>
- <https://ionicframework.com/docs/components>

Ionic es uno de los frameworks con más recorrido y más conocidos para el desarrollo de Apps híbridas multiplataforma. Inicialmente fue concebido como un framework de componentes adecuados visualmente para el desarrollo de Apps para móviles. Fue pionero ofreciendo componentes con diseños distintos y similares a los componentes nativos para Android o a iOS. Lo habitual era crear el proyecto con Ionic y empaquetarlo con Cordova, posteriormente crearon Capacitor y lo incluyeron como opción por defecto a la hora de empaquetar proyectos realizados con Ionic.

Actualmente Ionic ofrece varios productos interesantes y relacionados con el motivo de mi trabajo. Uno de ellos es Ionic Portals, que permite usar micro-frontales de una forma sencilla e intuitiva, pero enfocada al trabajo principal sobre los proyectos nativos de cada App y la realización de componentes basados en proyectos web. Además, es una solución de pago.

Otro producto interesante es AppFlow, un servicio que ofrece la capacidad de actualizar en tiempo real una App sin pasar por las tiendas oficiales. Funciona mediante la inclusión de unas dependencias en el proyecto de Ionic y también es una solución de pago. He tenido la posibilidad de trabajar con esta herramienta y su funcionamiento es bastante complejo, funciona con canales y entornos de una forma un poco enrevesada, y limitado, tiene una cuota máxima de usuarios a actualizar que suele ser de 5000 dispositivos, aunque se puede ampliar pagando más.

He hecho hincapié en estos dos productos para recalcar que **la solución ofrecida con mi trabajo aún estas dos soluciones de pago en una sola, ofreciendo micro-frontales y actualización en tiempo real de Apps móviles de una forma mucho más sencilla, sin costes adicionales y sin limitaciones de usuarios.**

En lo que se refiere a la App que he desarrollado no he utilizado el framework completo, solamente he usado los componentes de interfaz de usuario que ofrece Ionic. Estos componentes se usan en el proyecto micro-frontal sobre Angular 17.

Existen varias justificaciones para no usar directamente Ionic como framework para el desarrollo de la App. La principal es el hecho de que la tecnología para usar micro-frontales está preparada para proyectos de Angular y los proyectos de Ionic tienen una estructura y procesos diferentes. Por otro lado, Ionic suele ir alguna versión por detrás de Angular y en este caso quería probar a usar Angular 17 y la nueva sintaxis de directivas estructurales que ofrece (@if, @for, @defer...), también quería evitar hacer que la App dependa demasiado de Ionic puesto que cada vez ofrece más productos de pago y quizás en un futuro quiera utilizar otro empaquetador.

Durante el desarrollo he encontrado algunos inconvenientes a la hora de incluir los componentes de Ionic en el proyecto micro-frontal, el principal es el hecho de que el proyecto que carga inicialmente es el proyecto Shell y posteriormente carga el micro-frontal, pero

necesito tener cargados los componentes de Ionic previamente a esta carga, pero además no quiero incluir esta lógica en el proyecto Shell porque quiero tener la libertad de actualizarlos sin necesitar pasar por las tiendas oficiales. Para ello hago una carga dinámica en el proyecto micro-frontal, de esta forma puedo cargar librerías JavaScript y estilos CSS cuando lo necesito.

4.5 – NestJS 3



- <https://nestjs.com>

Los que nos dedicamos al desarrollo de Apps sabemos que una de las partes más importantes y esenciales, sin las que una App no sería nada, es la lógica de servidor conocida como “backend”. Todo lo que puede o no puede hacer una App depende en gran parte de los procesos realizados en uno o varios servidores y, en el caso de esta App, la gran parte de la lógica del negocio y, por lo tanto, el éxito de la App depende del backend.

Esta App puede ser considerada como un mero cliente que se comunica con una API Rest a la que envía datos para conseguir información que presentará al usuario.

Todo el backend ha sido desarrollado de cero sobre NestJS 3 y esta decisión ha sido tomada teniendo en cuenta muchos motivos de peso. Sobre la mesa existía la posibilidad de hacerlo en alguna tecnología distinta a NodeJS como Java o PHP, pero fueron descartadas ya que con NestJS trabajaría en todos los proyectos con TypeScript, un lenguaje con el que me defiendo bastante bien, además tengo más experiencia desplegando este tipo de proyectos y conozco más herramientas y proveedores adecuados. Una vez decidido el hecho de que trabajaría en un entorno NodeJS tenía sobre la mesa otras dos opciones, usar NestJS que no lo conocía demasiado o usar directamente ExpressJS que si lo conocía más. Finalmente, después de documentarme tome la decisión en base a que NestJS permite crear CRUDs de entidades en pocos minutos, ofrece seguridad básica en los servicios de forma muy sencilla de configurar y la estructura del proyecto es muy similar a Angular, ExpressJS ofrece mucha libertad, quizás demasiada y te obliga a tener que desarrollar toda la lógica por tu parte. A fin de cuentas, NestJS utiliza ExpressJS por debajo y la lógica de decoradores ofrecida por NestJS es tan potente y tan sencilla que me convenció por completo.

Trabajando con NestJS me encontré con bastantes problemas debido a mi propio desconocimiento, pero existe una buena comunidad en la red donde todos estos problemas estaban solucionados y explicados.

Lo más interesante es la posibilidad de desarrollar rápidamente nuevos servicios si fuese necesario y el desarrollo de un decorador @Auth propio que me permite autenticar un servicio rápidamente para que solo funcione si se recibe un token valido en la cabecera de la llamada.

De momento, el CORS está configurado, pero permite todos los orígenes, al ser una App realizada con Capacitor al menos tendré que permitir los orígenes <http://localhost> e <ionic://localhost> que son los schemas que utiliza. Pero de momento, por comodidad del desarrollo no está limitado.

A parte de la lógica de API Rest ofrecida, el backend tiene una lógica de negocio esencial que es la de recopilar la información oficial de los juegos. Actualmente solo existe la lógica para

recopilar resultados de los juegos de Lotería Nacional, pero la idea es añadir una lógica diferente para cada tipo de juego.

Esto se consigue mediante una tarea CRON que se ejecuta los días en las que los sorteos son realizados y en las horas donde suelen estar disponibles la nueva información.

4.6 – MongoDB



- <https://www.mongodb.com>

La persistencia de los datos es otra de las patas esenciales en el desarrollo de una App, existen Apps que pueden funcionar con una persistencia de datos local, pero en este caso es necesaria una persistencia de datos fuera del propio dispositivo móvil.

La elección de una base de datos no relacional en vez de una solución relacional está motivada por el hecho de que esta App está pensada sobre todo para lectura de datos, necesitaba sencillez ya que las estructuras de datos necesarias son simples, al trabajar con JavaScript el hecho de que mongoDB trabaje con JSON es una ventaja y además está el hecho de que me apetecía probar este tipo de bases de datos.

Es la API Rest la que trabaja directamente con la base de datos, para ello uso la librería de Moongoose (<https://mongoosejs.com>) que me permite realizar operaciones de lectura y escritura de una forma sencilla.

La base de datos esta publicada en mongoDB-Atlas, el servicio oficial de mongoDB que, con su capa gratuita, consigo cubrir las necesidades iniciales de sobra y, en caso de necesitar escalar los requisitos, puede hacerse de forma rápida y sencilla.

Para trabajar en local utilizo el cliente mongoDB Compass.

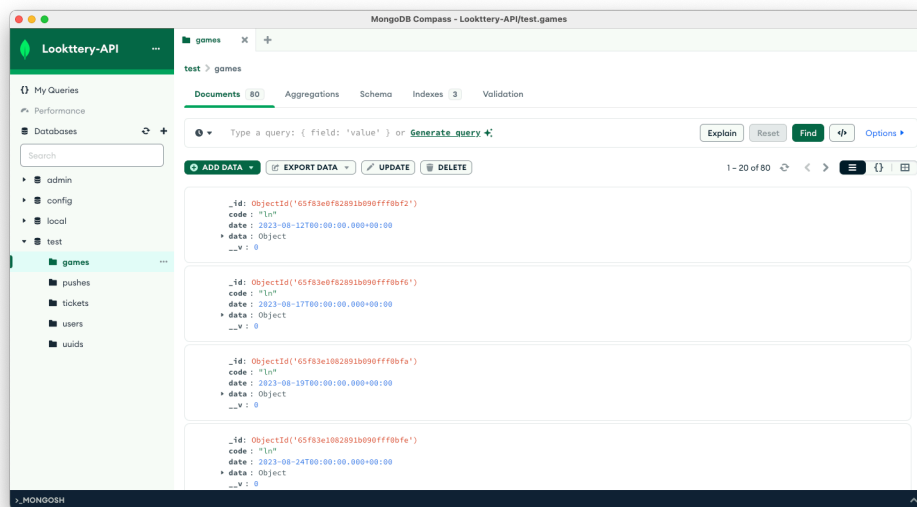


Ilustración 12 - MongoDB Compass

MongoDB Atlas ofrece paneles de métricas que me permiten ver el consumo realizado de la App.

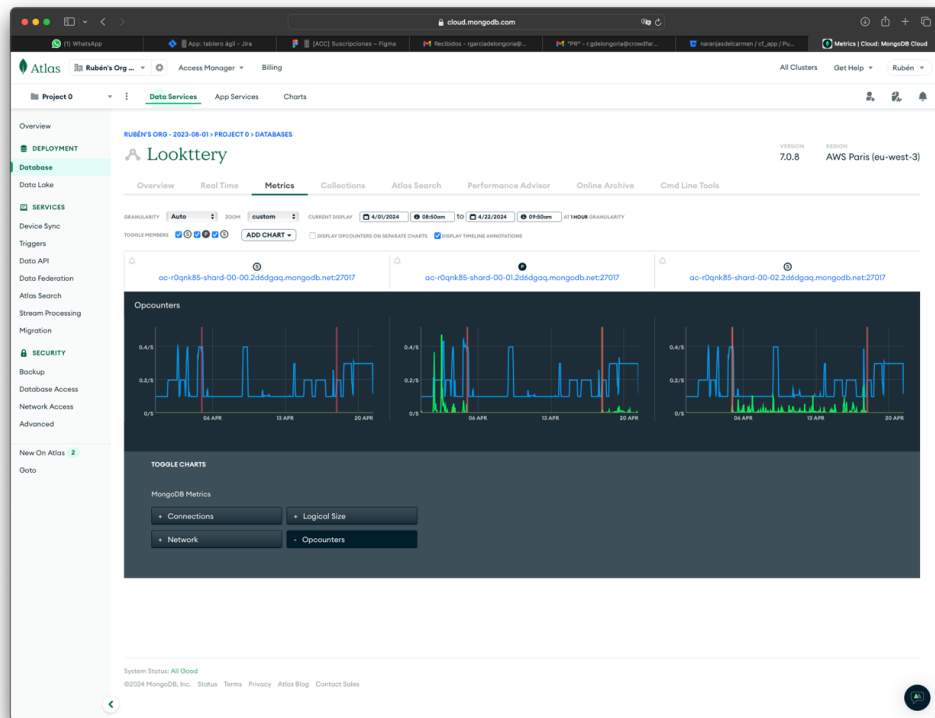


Ilustración 13 - MongoDB Atlas Métricas

4.7 – Github Pages



Para poder desplegar el proyecto micro-frontal de la App he utilizado la posibilidad que ofrece GitHub de publicar proyectos web estáticos de forma gratuita y estable mediante sus GitHub Pages.

Al principio pensé en publicar el proyecto en un hosting corriente, pero me di cuenta de que era demasiado para lo que necesitaba, de hecho, solo es necesario publicar unos ficheros JavaScript y el fichero remoteEntry.json, por lo que me planteé finalmente esta solución o las páginas web estáticas de Firebase. Al final me decanté por GitHub Pages ya que son muy sencillas de utilizar y me permiten realizar despliegue continuo desde el propio repositorio. Actualmente, para publicar una nueva versión del micro-frontal, solo es necesario subir un

nuevo cambio a la rama master del repositorio y a los pocos minutos el cambio estará en producción.

El principal inconveniente es que no quiero compartir el repositorio de código de forma pública, por lo que he creado un nuevo repositorio público llamado lookttery-static que contiene el código compilado y preparado para producción (<https://github.com/rgarciadelongoria/lookttery-static>), además, por si el día de mañana prefiero cambiar de proveedor de servicios de hosting, dispongo de un dominio propio (lookttery.com) para el micro-frontal y la API Rest, de esta forma el código no cambiará.

Aunque no es parte del objetivo, desde www.lookttery.com puede verse el proyecto de micro-frontal desde un navegador corriente.

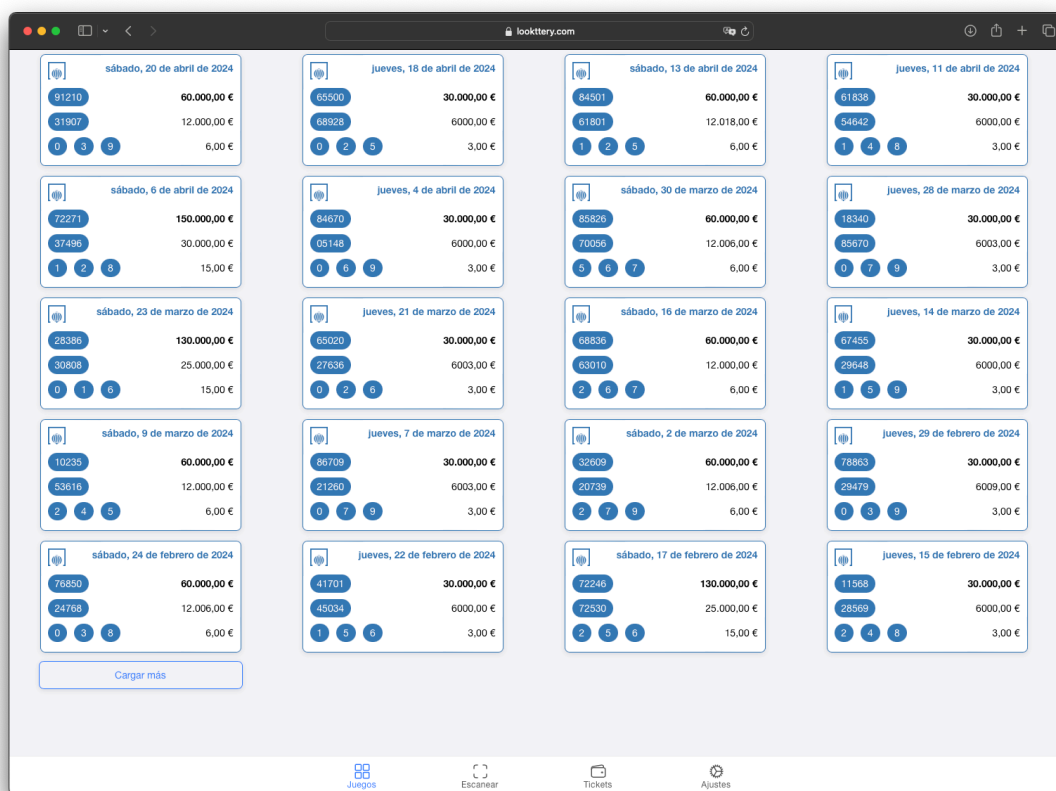


Ilustración 14 - Lookttery.com web

4.8 – Firebase



- <https://firebase.google.com>

Firebase es una de las plataformas para el desarrollo de Apps más conocidas, ofrece muchos servicios que se usan de forma habitual en Apps como gestión de usuarios, almacenamiento, gestión de errores, integración con servicios de Google, entre otros.

En el caso de la App que estoy desarrollando actualmente solo necesito Firebase para utilizar Notificaciones Push. Podría haber utilizado El servicio SNS de Amazon como alternativa, pero me parece algo más complicado de configurar en local para el desarrollo. En cambio, es probable que utilice más servicios a futuro de Firebase como por ejemplo Crashlytics y prefiero tener todo en una misma plataforma, además la configuración es bastante sencilla y la API para enviar notificaciones desde el backend es bastante intuitiva.

No me he encontrado muchos problemas con Firebase, quizás el mayor problema lo he tenido cuando estaba realizando la lógica de envío de notificaciones desde la API de Google, ya que el formato es distinto para Android y para iOS y, al internacionalizar los textos de la notificación, confundí el nombre exacto del parámetro de título y texto de ambas plataformas y no conseguía ver el error.

4.9 – Railway



Railway

- <https://railway.app>

Railway es la plataforma Cloud en la que despliego la API Rest utilizada por la App.

A la hora de elegir una plataforma para desplegar la API Rest basada en NodeJS me centré en servicios que no fuesen excesivamente complicados y que cubriesen mis necesidades principales. Una de las opciones fue usar Amazon, pero por motivos similares de complejidad inicial y de exceso de productos decidí buscar servicios más sencillos pero potentes. Probé muchos servicios con capas gratuitas solventes y al final me decanté por Railway que ofrece despliegue continuo directamente desde el repositorio y los planes de pago, en caso de necesitarlos, son económicos y ofrecen unas características más que suficientes.

Railway también fue una opción inicialmente sobre la que desplegar el proyecto micro-frontal, pero lo descarté por complejidad, ya que previamente había que configurar en servidor de aplicaciones web para desplegarlo y no quería perder demasiado tiempo en esta parte.

También aprovecho la configuración de secretos que ofrece y la separación de proyectos por entornos. Con un solo repositorio, puedo desplegar en diferentes entornos que apuntan a diferentes secretos de una forma tan sencilla como tener dos ramas distintas en el repositorio.

No he tenido ningún problema usando Railway, de hecho, avisa con cada problema de backend y reintenta los despliegues en caso de error, los logs que ofrece están muy claros y tiene buena documentación.

5 – Lookttery App



- Play Store: https://play.google.com/store/apps/details?id=com.lookttery.app&hl=es_PE
- App Store: <https://apps.apple.com/es/app/lookttery/id6482973016>
- Lookttery-App: <https://github.com/rgarciadelongoria/uoc-tfg-web>
- Lookttery-Shell: <https://github.com/rgarciadelongoria/uoc-tfg-base>
- Lookttery-API: <https://github.com/rgarciadelongoria/uoc-tfg-api>

5.1 – Introducción

Lookttery es una App para usuarios que juegan a juegos de lotería, el MVP solo pretende alcanzar unas funcionalidades iniciales que se detallan a continuación, no obstante, tengo en mente que esta App amplíe su funcionalidad y la cantidad de información con la que trabajar, ampliando su capacidad a juegos de loterías de otros países.

5.2 – MVP

El objetivo principal es conseguir todas las funcionalidades del MVP que son las siguientes:

- Mostrar información en tiempo real y de forma cronológica los sorteos de Lotería Nacional.
- Poder buscar el premio de un número concreto para cada sorteo de Lotería Nacional.
- Poder escanear décimos reales de Lotería Nacional y mostrar su premio o si son de sorteos futuros.
- Guardar los décimos escaneados y mostrarlos al usuario.
- Recibir notificaciones push cuando se realice el sorteo de algún decimo guardado.
- App multi-idioma.

Actualmente la App realiza todas estas funcionalidades.

El tiempo de descarga de la App es bastante reducido, ya que gracias a la tecnología de micro-frontales que permite cargar la lógica de la App en remoto cada vez que se inicia la App hace que el paquete final ocupe muy poco. Por lo que al descargar la App por primera vez se ofrece una agradable experiencia de usuario.

Una vez que el usuario abre la App por primera vez se encontrará con una pantalla modal de bienvenida. En esta pantalla se le permite elegir el idioma de la App en primer lugar y, posteriormente, se le explican y solicitan los permisos para ser notificado y poder usar la cámara del dispositivo para poder escanear los décimos de lotería.

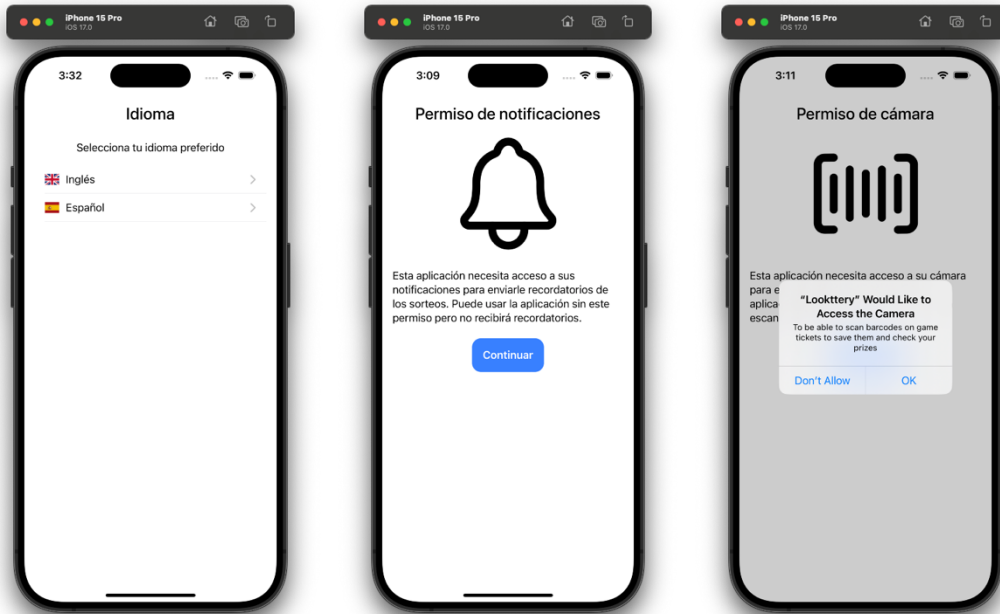


Ilustración 15 - Onboarding

Esto es una buena práctica solicitada por Apple en una de las revisiones previas a la publicación de la App en producción. También es importante comentar que el idioma elegido en la App puede ser distinto al que se encuentre configurado en el dispositivo, el cual es utilizado por las notificaciones push, o para mostrar mensajes de solicitud de permisos para internacionalizar su contenido.

El proceso de configuración de la pantalla de bienvenida puede lanzarse de nuevo desde la pantalla de configuración de la App, que se detalla más adelante, pero es importante saber que los permisos, una vez aceptados o rechazados, solo se pueden modificar desde el propio sistema operativo del dispositivo, en la configuración propia de la App.

Una vez finalizado el proceso de bienvenida el usuario puede ver una barra con cuatro opciones principales: juegos, escanear, tickets y configuración. Inicialmente se accede a la pantalla de juegos donde puede ver el listado de sorteos realizados por orden cronológico. Actualmente solo existen datos del sorteo de Lotería Nacional, pero en un futuro aparecerán mezclados más sorteos y se tendrán que incluir filtros y configuraciones para que el usuario pueda encontrar fácilmente la información que buscar.

El usuario puede acceder al detalle de uno de los sorteos tocando un elemento de la lista, en este caso accede a una pantalla con la información principal y un listado completo de todos los premios del sorteo, que podrá filtrar mediante un campo buscador que le permitirá encontrar un número o varios resultados que contengan un valor concreto.

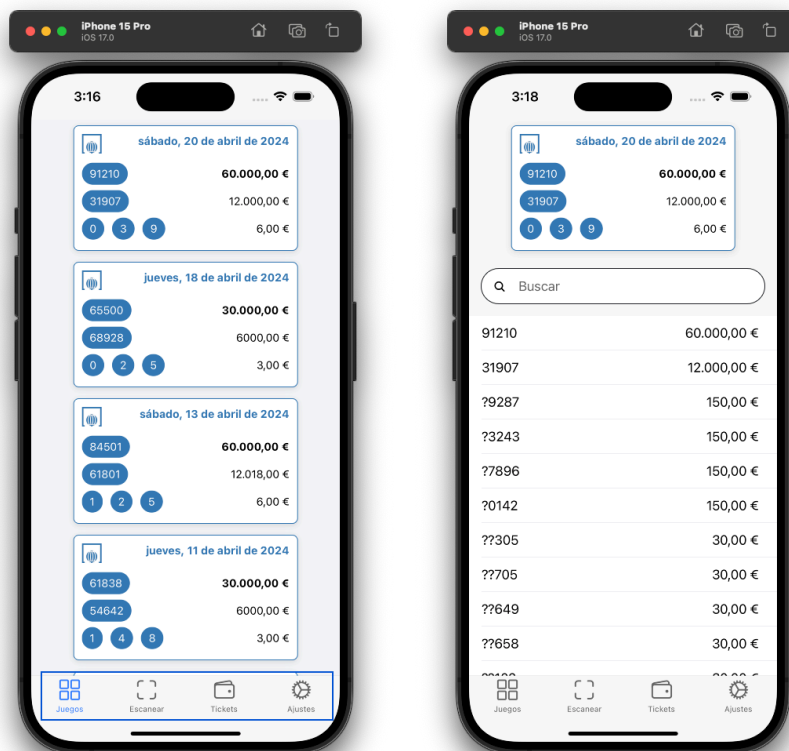


Ilustración 16 - Pantalla inicial y detalle del sorteo

Para salir de esta pantalla el usuario deberá tocar alguna de las opciones principales, si quiere volver tocará la “juegos”.

Cuando el usuario toca la opción de “escanear”, se enciende la cámara del dispositivo donde puede verse un puntero para usar como referencia a la hora de escanear un código de barras.

El simulador de iOS no ofrece la posibilidad de usar una cámara real por lo que las capturas de esta parte se han realizado con un dispositivo real.

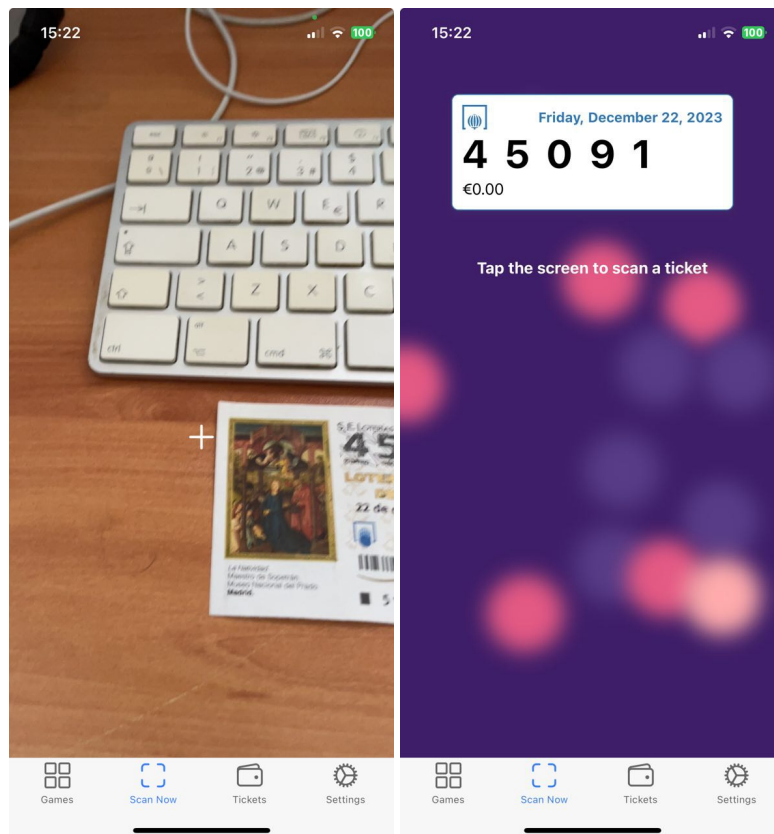


Ilustración 17 - Función para escanear décimos

Esta es una de las funcionalidades principales de la App y, ahora solo entiende códigos de barra de décimos de Lotería Nacional, pero la idea es que entienda de todos los sorteos posibles como un único punto de entrada y muestra información para cada tipo de sorteo en base al código escaneado.

Si el código escaneado es válido se mostrará una pantalla con los datos del décimo y el premio conseguido, se entiende que si el premio es de 0€ es que no se ha conseguido ningún premio. Si el usuario ve el mensaje sorteo pendiente es porque el décimo escaneado pertenece a un sorteo que aún no ha sido realizado. Esta pantalla se cierra tocando en cualquier parte de la pantalla lo que devuelve al usuario a la pantalla de escaneado de décimos. Si el usuario quiere salir de esta pantalla debe tocar alguna de las opciones principales.

Cuando el usuario toca la opción "tickets", accede a un listado con todos los tickets escaneados de forma cronológica según la fecha del sorteo de cada décimo, primero los de mayor fecha. Aquí pueden coexistir tickets de sorteos ya realizados con tickets de sorteos que aún no han sido realizados. Si el usuario toca un elemento de la lista y este elemento es de un sorteo realizado navegará automáticamente al detalle del sorteo.

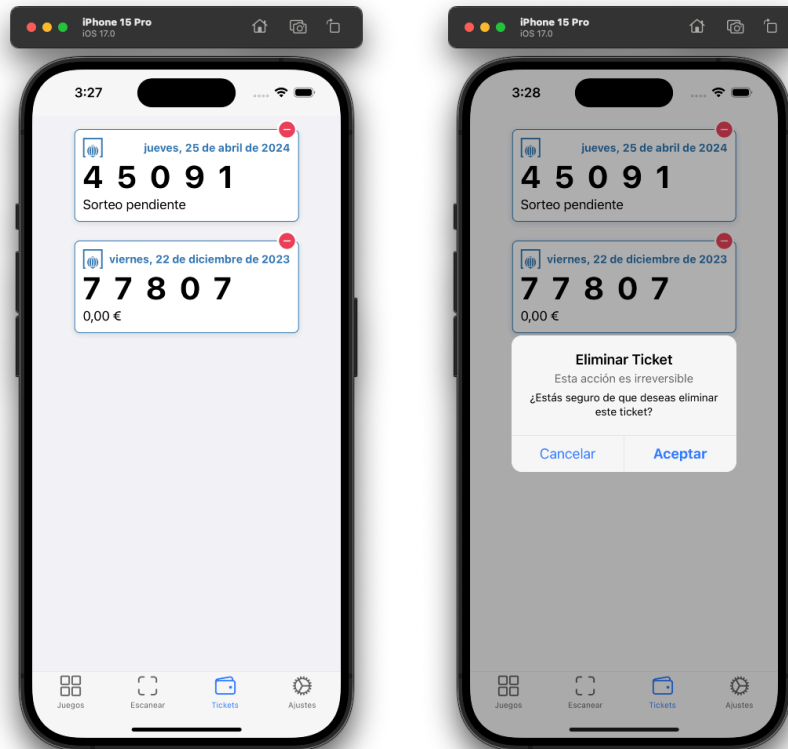


Ilustración 18 - Listado de tickets

La información de cada elemento depende de la fecha en que se accede al listado, si el usuario tiene tickets pendientes, pero accede de nuevo cuando el sorteo ya ha sido realizado, los tickets mostrarán el premio que se ha conseguido.

Cada elemento de la lista ofrece un botón que permite eliminar el ticket guardado.

Cuando un sorteo es realizado el backend de la App detectará el nuevo sorteo y comprobará los tickets existentes para este sorteo en la base de datos en ese momento, posteriormente enviará una notificación push a cada dispositivo del usuario relacionado con esos tickets. De esta forma el usuario puede saber que ya hay datos nuevos para alguno de los tickets que tiene guardados.



Ilustración 19 - Notificación Push

Para salir de esta pantalla hay que tocar alguna de las opciones principales.

La última opción principal es la de “configuración”, que ofrece al usuario información y opciones. Desde esta pantalla de momento se puede ver el código QR con el identificador único del dispositivo, una opción para cambiar de idioma y otra opción para relanzar el proceso de bienvenida.

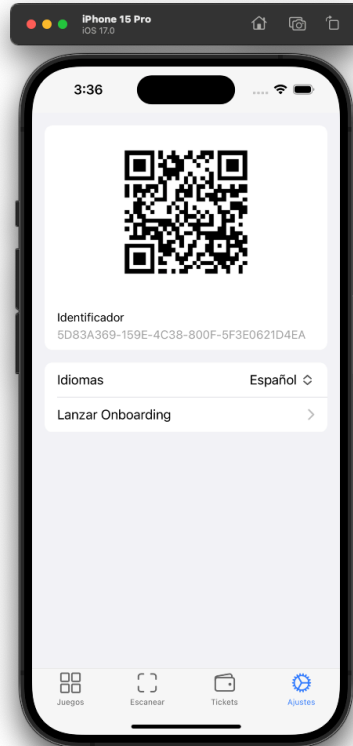


Ilustración 20 - Pantalla de configuración

En esta pantalla existirá una opción denominada “desvincular dispositivo” que solo aparece cuando nuestro dispositivo está vinculado con otra cuenta de otro dispositivo.

Looktery permite que un usuario con distintos dispositivos tenga la misma información en todos ellos. Para ello el usuario debe escanear desde la App el código de QR del identificador único del dispositivo, el cual aparece en la pantalla de “configuración”. El usuario será informado de que se va a sincronizar este dispositivo y deberá aceptar o cancelar esta acción.

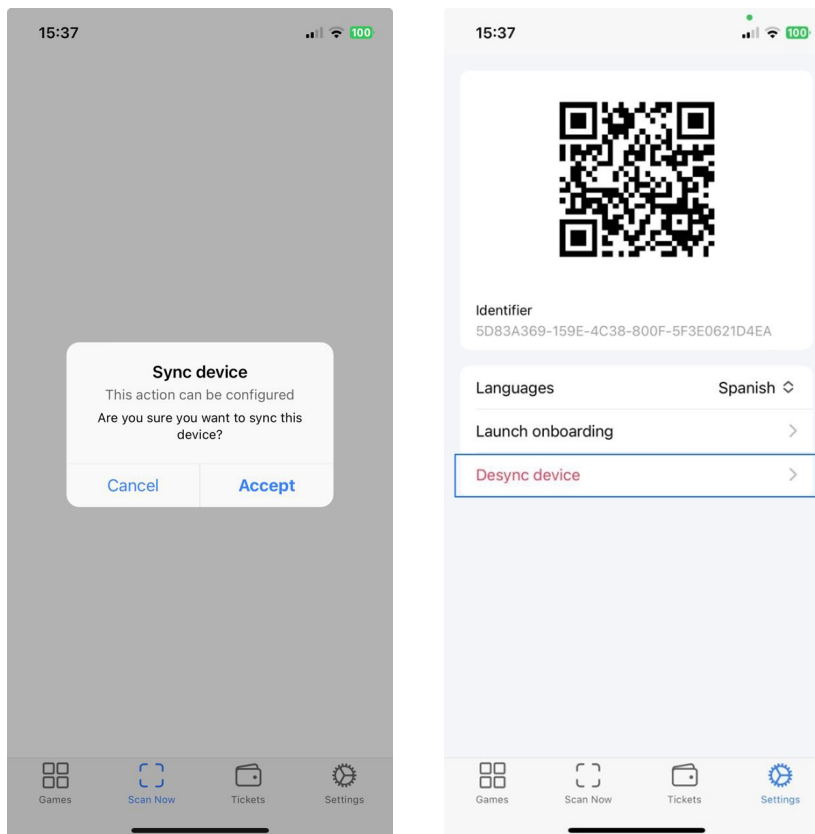


Ilustración 21 - Sincronizar dispositivos

Una vez aceptada la acción el dispositivo que ha escaneado el código QR se convierte en algo parecido a una copia del otro dispositivo y compartirán los mismos tickets. Los tickets existentes ya no se verán y en cambio veremos los del otro dispositivo. Si borramos un ticket en un dispositivo este ticket desaparecerá en el otro dispositivo y, si escaneamos un décimo nuevo, este aparecerá en el otro dispositivo. Por supuesto hay que recargar la información para que la App actualice correctamente la nueva información.

Las notificaciones push también se ven afectadas y ahora recibiremos las mismas notificaciones push de los tickets existentes en ambos dispositivos.

Esta acción se puede deshacer desde la opción de “desvincular dispositivo” del menú de configuración, recuperando la información existente en el momento en que se realizó la sincronización.

5.3 – API Rest

La infraestructura sobre la que se apoya la API Rest que utiliza Lookttery es un servidor NodeJS alojado en la nube de Railway. Mediante este servicio puedo desplegar fácilmente y de forma automática cada cambio que subo a la rama “master” del proyecto, además ofrece recuperación de instancias desplegadas con anterioridad por se despliega una versión con errores.

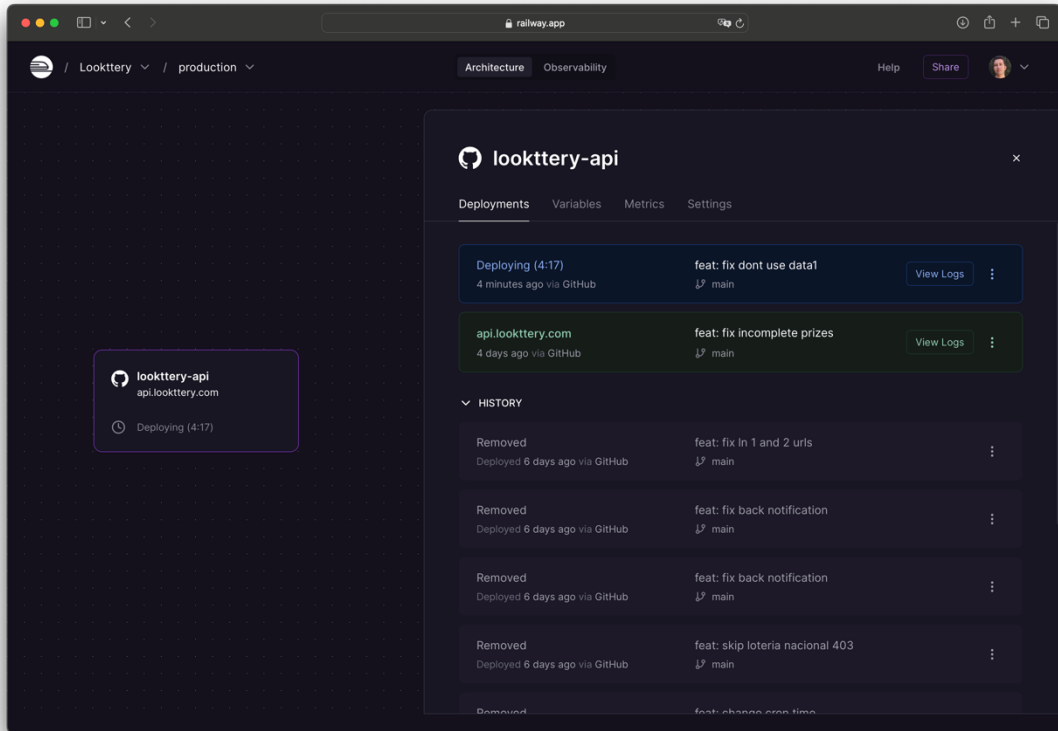


Ilustración 22 - Panel principal de Railway

Como se aprecia en la imagen he configurado el dominio personalizado para la API Rest como `api.lookttery.com`, de esta forma el código es independiente del servicio que se utilice.

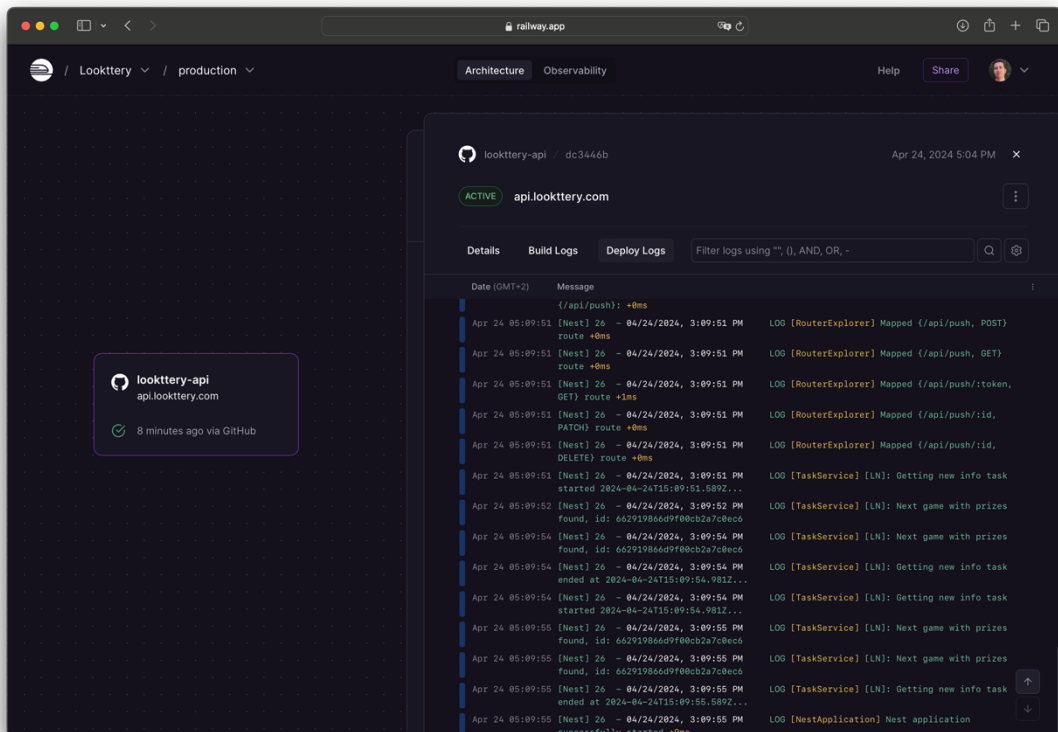


Ilustración 23 - Logs de la API Rest

Railway nos ofrece la posibilidad de ver fácilmente los logs de despliegue de la aplicación de NodeJS, de esta forma puedo saber en qué momento se ha lanzado la tarea CRON que recoge la información externa y conocer si ha habido algún problema en la recuperación de los datos.

Para trabajar en las funcionalidades de la API Rest utilizo el plugin de Postman para Visual Studio Code el cual me permite, en un mismo entorno de desarrollo, probar las funcionalidades que desarrollo en diferentes entornos.

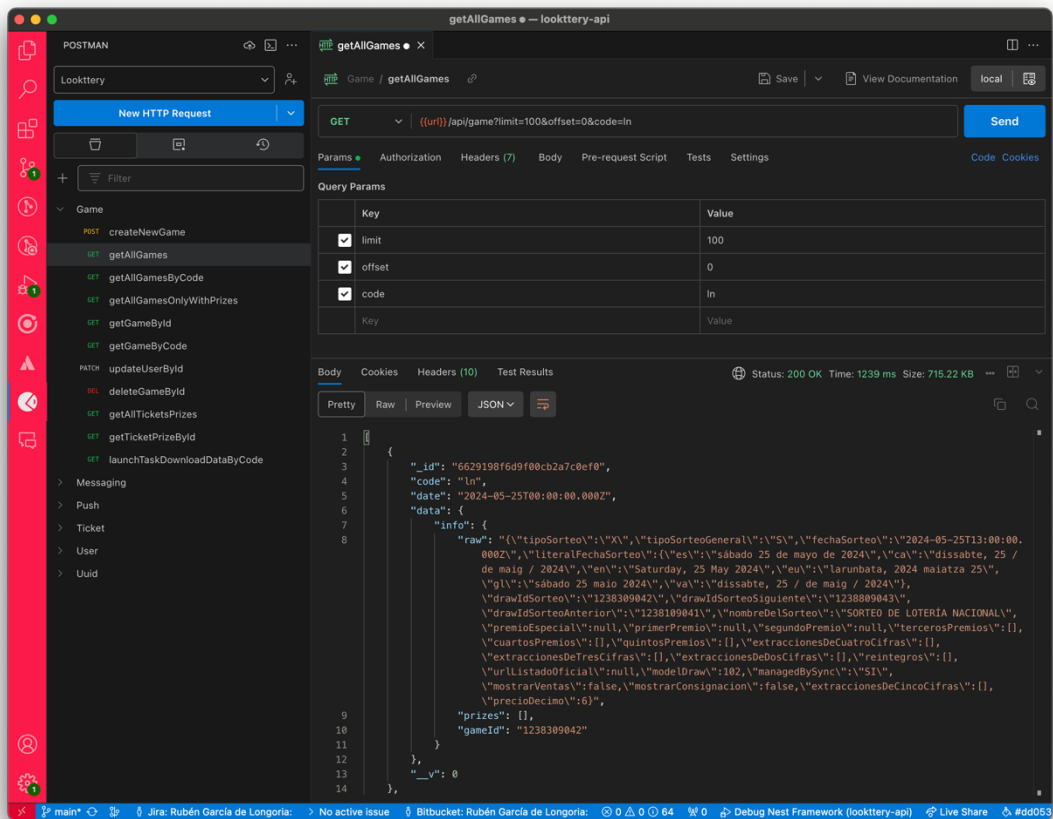


Ilustración 24 - Postman VSCode

La API Rest ofrece funcionalidades CRUD para cada entidad definida, actualmente existen Game, Push, Ticket, User y Uuid, cada una de ellas es una colección distinta en la base de datos.

5.4 – Pruebas de usuario y despliegue en producción

El ciclo de desarrollo de toda App finaliza en la puesta en producción de la misma, que suele ser la publicación en las tiendas oficiales. Para asegurarme de que a producción llegan las funcionalidades sin errores he seguido los pasos habituales en el ciclo de desarrollo de una App.

5.5 – Test unitarios

Para el correcto desarrollo de la App y con el fin de evitar en la medida de lo posible errores he realizado test unitarios en algunas partes del código. Lo ideal sería llegar a tener una cobertura real del 90% del código, no obstante, en este caso, para el MVP he desarrollado una menor cantidad de test que cubra alguna de las funcionalidades esenciales. Antes de continuar con el desarrollo de funcionalidades adicionales de la próxima versión debería conseguir el objetivo de test para asegurarme que con los nuevos desarrollos no rompo funcionalidad existente.

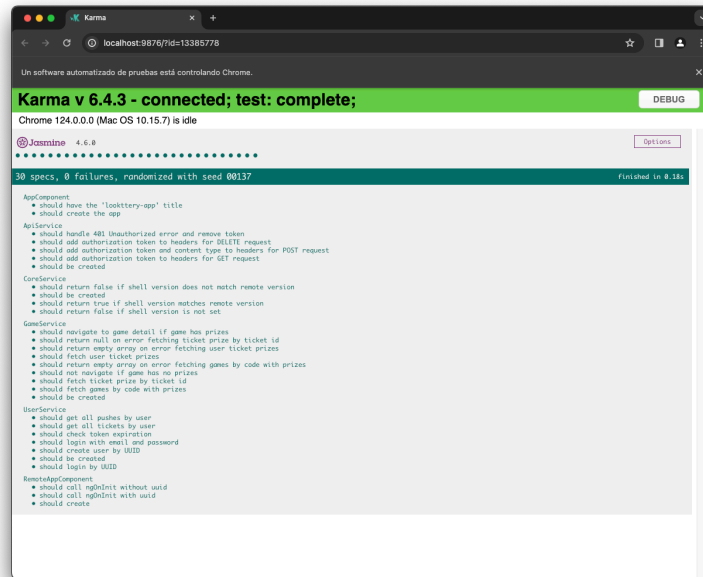


Ilustración 25 - Test unitarios micro-frontal

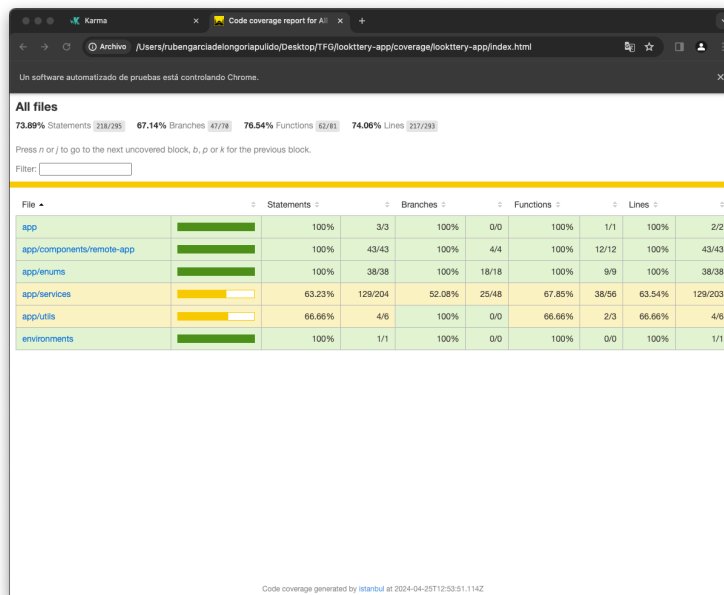


Ilustración 26 - Cobertura test micro-frontal

6 – Desarrollo y ejecución del proyecto en local

En este apartado pretendo explicar cómo trabajar sobre el proyecto en un equipo local basándome en la forma en la que yo mismo he trabajado y desarrollado el MVP.

Para poder trabajar es necesario tener los recursos y tecnologías básicas necesarias instaladas y configuradas previamente en el equipo.

Personalmente prefiero usar VSCode como entorno de desarrollo y aprovechar el terminal embebido.

6.1 – Proyecto lookttery-api

Este proyecto contiene la lógica de la API Rest utilizada por la App. Está desarrollado mediante el framework NestJS.

6.1.1 – Ejecución en local

Los pasos básicos para utilizar el proyecto están descritos en el fichero `readme.md` del proyecto.

Lo primero que debemos hacer es instalar las dependencias mediante el comando `npm i`.

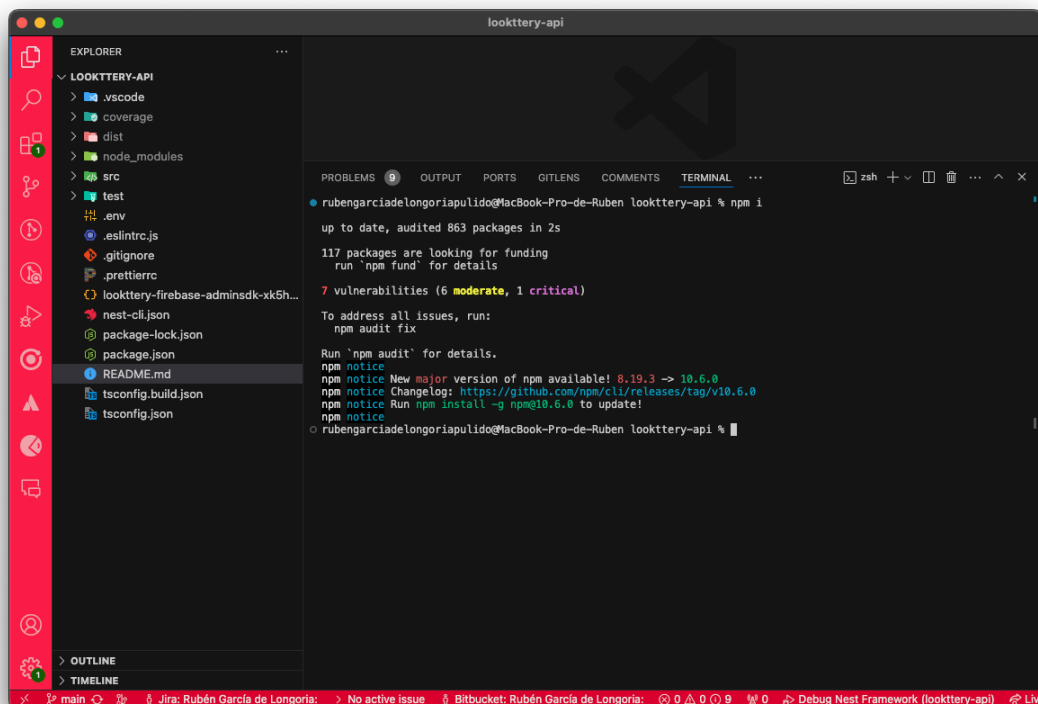


Ilustración 27 - Lookttery-api instalación de dependencias

Una vez las dependencias estén descargadas hay que configurar las variables de entorno del fichero `.env` que se encuentra en la raíz del proyecto:

- MONGO_DB: Cadena de conexión de la base de datos.
- PORT: Puerto de ejecución.
- JWT_SECRET: Clave secreta de generación de tokens de autenticación.
- RUN_TASKS: Si queremos que se ejecuten tareas cron.

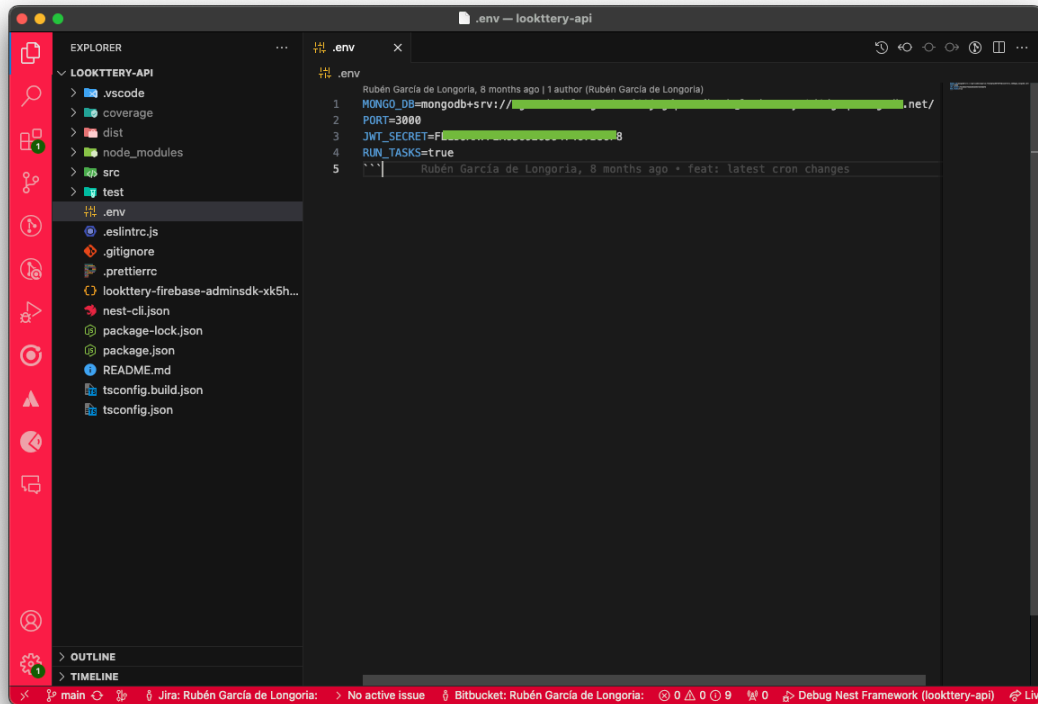


Ilustración 28 - Lookttery-api configuración de secretos

En mi caso particular he trabajado con una cadena de conexión externa para la BBDD en vez de trabajar directamente en local.

Una vez realizada esta sencilla configuración podemos lanzar el proyecto mediante el comando `npm run start:dev` el cual nos permite ver los logs de despliegue, donde podremos ver si la conexión con la base de datos se ha realizado correctamente o si ha ocurrido un error.

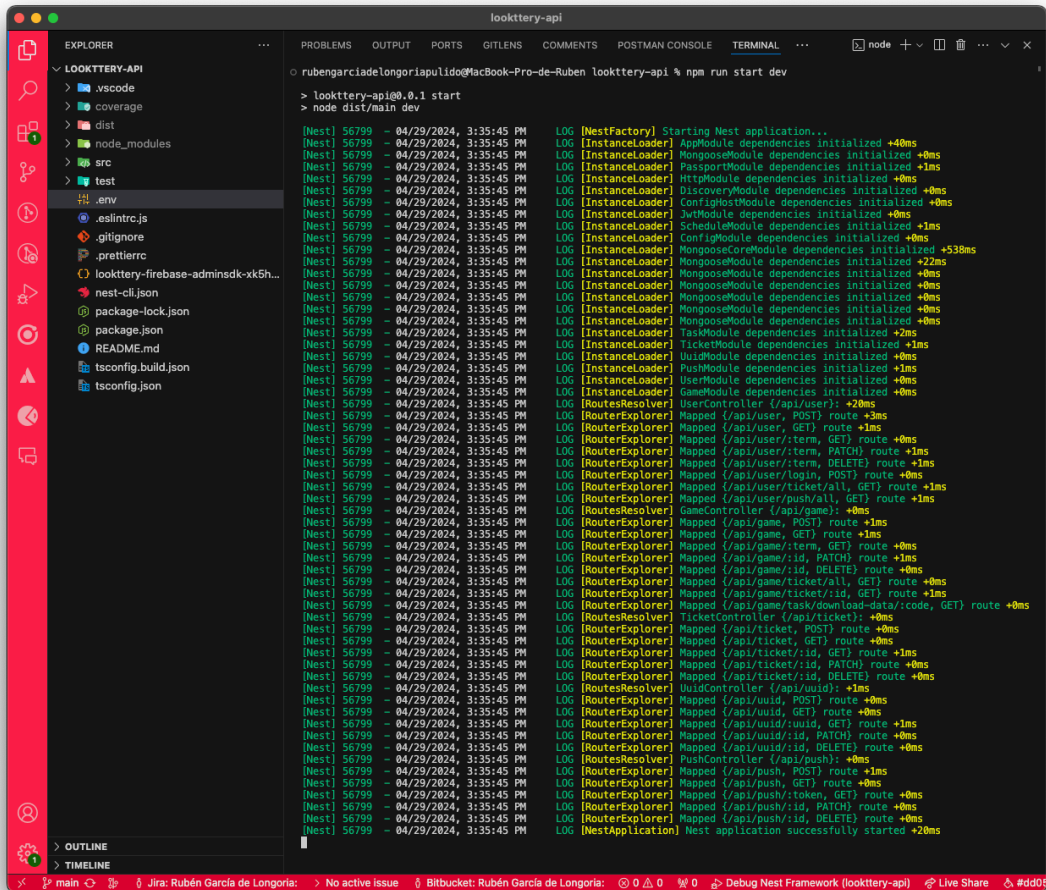


Ilustración 29 - Looktty api ejecución en local

6.1.2 – Herramientas de debug

Durante el desarrollo siempre es necesario disponer la capacidad de depurar nuestro código, para ello no podemos lanzar el proyecto con el comando anterior. Para este caso necesitamos arrancar el proyecto en modo debug desde la opción run & debug de VSCode.

Esta herramienta nos permite depurar con un control total durante el tiempo de ejecución. Una de las funcionalidades que considero más importantes y que más me han ayudado durante el desarrollo ha sido la posibilidad de editar los puntos de ruptura para incluir condiciones y así conseguir detener el programa en el momento exacto de un estado concreto.

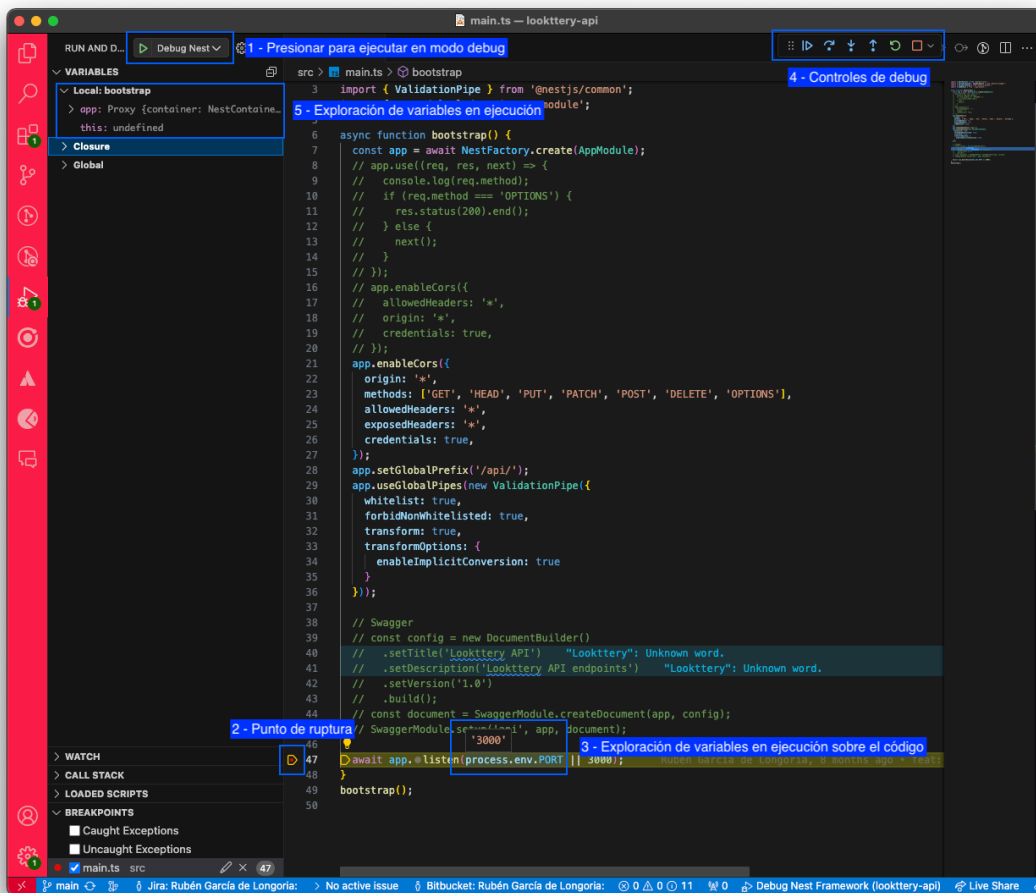


Ilustración 30 - Lookttery api debug

6.1.3 – Colección de endpoints

Una vez tenemos lanzada la app en local podemos realizar llamadas a los distintos endpoints que ofrece. Para tener un mejor control y evitar el exceso de programas distintos abiertos he utilizado la extensión de Postman para VSCode, de esta forma puedo tener mis colecciones de llamadas ordenadas y en un mismo sitio.

Como se ve en la siguiente imagen tengo todas las funcionalidades principales de Postman y puedo tener ordenada mi colección de llamadas para realizar pruebas en distintos entornos. En este caso me ha sido muy útil poder elegir entre mi entorno local (localhost) o de producción (api.lookttery.com) además de la facilidad para realizar llamadas con cabeceras de autenticación.

La mayoría de los endpoints de la API Rest están autenticados, requieren que el usuario este autenticado, este control se realiza mediante la inclusión de la cabecera Bearer Token en la llamada.

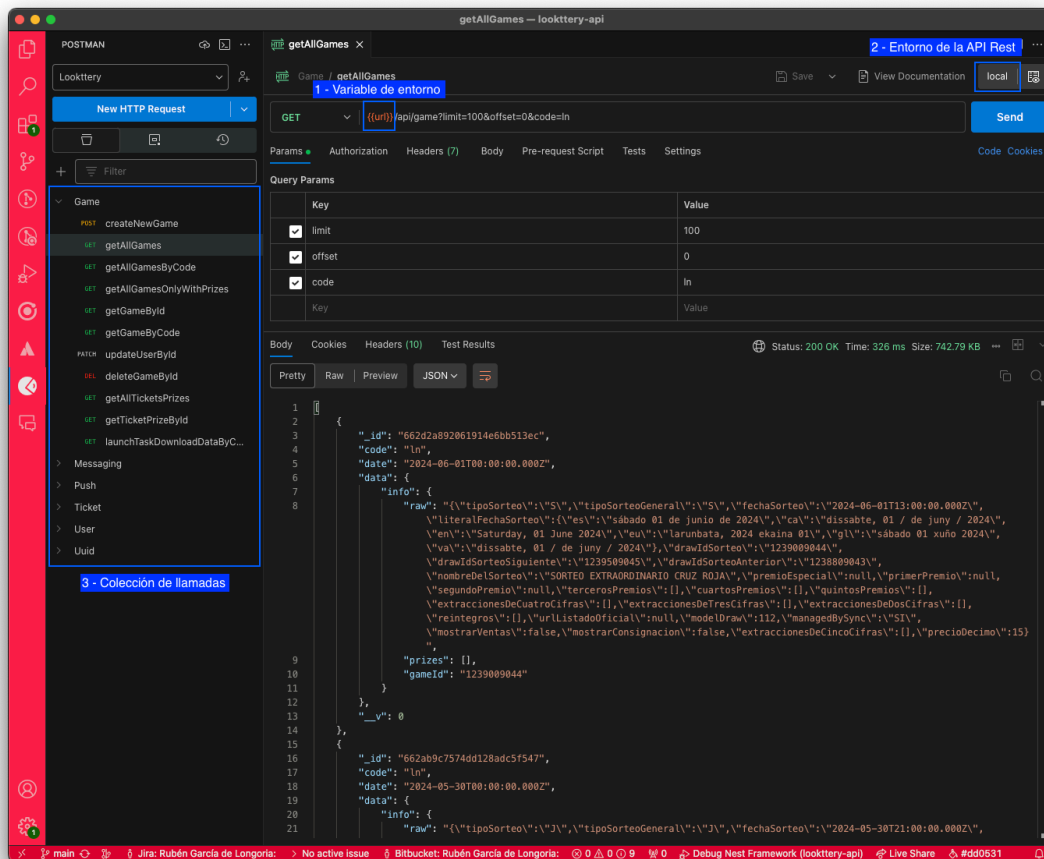


Ilustración 31 - Lookttery api colección de llamadas

6.1.4 – Entidades de negocio

Existen cinco entidades principales que son las piedras angulares de la lógica de la App:

- Game: Colección de sorteos, contiene la información de cada sorteo como las fechas y los premios.
- Push: Colección de tokens de notificaciones de cada dispositivo. Contiene el token de notificaciones del dispositivo y el usuario al que pertenece.
- Ticket: Colección de tickets, contiene la información de cada ticket como el código de barras, el tipo de sorteo al que pertenece y la fecha de su sorteo.
- User: Colección de usuarios, contiene la información de cada usuario como su nombre, email, password o marca de activo.
- Uuid: Colección de identificadores únicos de dispositivos, contiene el identificador y el usuario al que pertenece.

En la siguiente figura se muestra de forma esquemática las distintas entidades y como quedan relacionadas entre sí.

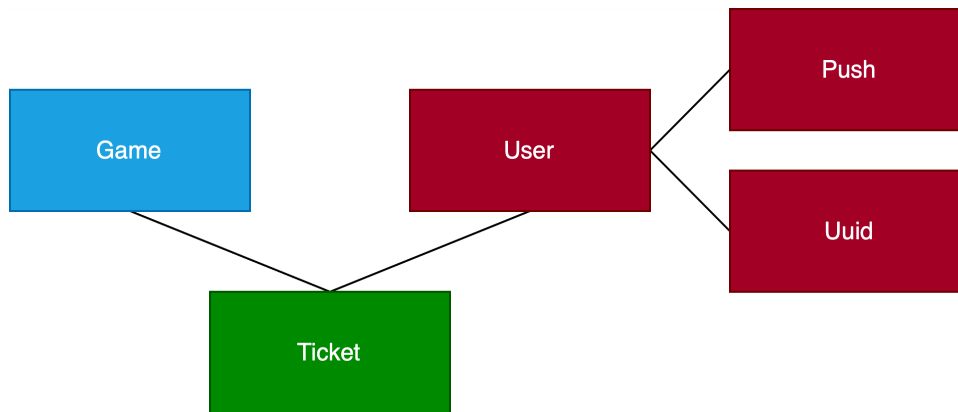


Ilustración 32 - Diagrama de entidades

De forma superficial y, como se aprecia por los tres colores que las diferencian, hay tres grupos diferenciados, las entidades User, Push y Uuid están relacionadas con el usuario, la entidad Game es independiente y la entidad Ticket relaciona al usuario, dueño del ticket y a un sorteo, el que nos indicará si tiene premio o no.

6.1.5 – Registro y login silencioso en una App sin usuarios

En los apartados anteriores he mencionado características como usuario, email, password o autenticación de endpoints, pero desde el principio ha quedado claro que Lookttery no es una App que requiera registro explícito de usuarios.

Internamente Lookttery está preparada para incluir un registro explícito de usuarios en un futuro, quizás más adelante queramos ofrecer funcionalidad de pago, un registro para evitar una más que probable publicidad o simplemente una nueva funcionalidad que requiera usuarios y aun no tenga en mente.

La colección User contiene los campos de email, password y name que actualmente se están completando con el identificador único del dispositivo.

```

_id: ObjectId('6621147f957d4055e1c471d0')
email: "a2d57d6b-6772-4b3f-92a5-100a53e4072a@uuid.app"
password: "$2b$10$uR0Qj6RGv5cYnL/HycpFdepE1FcmEiVph0GgMcsLqs3RAv6t5tM0m"
name: "A2D57D6B-6772-4B3F-92A5-100A53E4072A"
roles: Array (1)
isActive: true
__v: 0
  
```

Ilustración 33 - Ejemplo entidad User

Cada vez que se instala de nuevo la App se obtiene un identificador único con el que se registra de forma silenciosa una especie de usuario en la base de datos. Mediante este usuario se hace login en la API Rest y se obtiene un token de autenticación con una duración aproximada de dos horas.

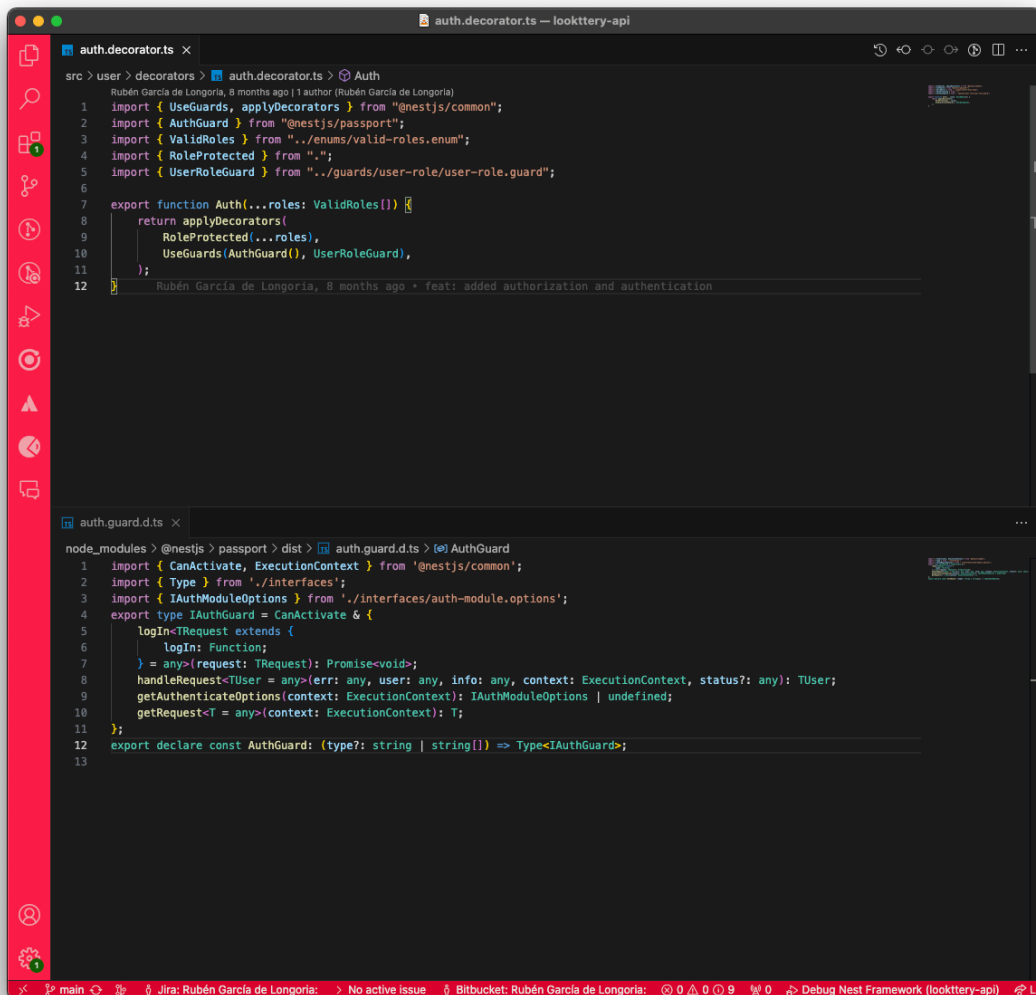
Una lógica propia de la App hace que con cada llamada se compruebe si el token esta caducado y, en este caso, hacer un nuevo login para conseguir otro token.

Actualmente los usuarios solamente sirven para poder acceder a la API Rest, todos son iguales y carecen de datos sensibles.

Con este sistema se pretende conseguir una barrera adicional de seguridad en la API Rest. A diferencia de un cliente web, las Apps realizadas con Capacitor tienen un origen (en la

documentación oficial se le llama schema) que suele ser <http://localhost> por lo que todas las llamadas a la API Rest se hacen con este origen y por lo tanto no se puede securizar mediante CORS que, aunque es un sistema de seguridad fácilmente evitable mediante proxis o clientes diferentes de navegadores web, ayuda a dificultar la explotación de nuestros recursos por parte de terceros desde fuera de la App.

Para facilitar mi desarrollo he creado un decorador `@Auth` propio que puede ser usado en cada endpoint de forma sencilla.



```
src > user > decorators > auth.decorator.ts > Auth
Rubén García de Longoria, 8 months ago | 1 author (Rubén García de Longoria)
1 import { UseGuards, applyDecorators } from '@nestjs/common';
2 import { AuthGuard } from '@nestjs/passport';
3 import { ValidRoles } from '../enums/valid-roles.enum';
4 import { RoleProtected } from './';
5 import { UserRoleGuard } from '../guards/user-role/user-role.guard';
6
7 export function Auth(...roles: ValidRoles[]) {
8   return applyDecorators(
9     RoleProtected(...roles),
10    UseGuards(AuthGuard(), UserRoleGuard),
11  );
12 }
Rubén García de Longoria, 8 months ago * feat: added authorization and authentication

node_modules > @nestjs > passport > dist > auth.guard.d.ts > AuthGuard
1 import { CanActivate, ExecutionContext } from '@nestjs/common';
2 import { Type } from './interfaces';
3 import { IAuthModuleOptions } from './interfaces/auth-module.options';
4 export type IAuthGuard = CanActivate & {
5   |
6   | login<TRequest> extends {
7     | login: Function;
8   } = any>(request: TRequest): Promise<void>;
9   handleRequest<TUser> = any>(err: any, user: any, info: any, context: ExecutionContext, status?: any): TUser;
10  getAuthenticateOptions(context: ExecutionContext): IAuthModuleOptions | undefined;
11  getRequest<T> = any>(context: ExecutionContext): T;
12 };
13 export declare const AuthGuard: (type?: string | string[]) => Type<IAAuthGuard>;
```

Ilustración 34 - Lookttery api decorador @Auth

Mediante este decorador y otro decorador propio llamado `@GetUser`, puedo tener conocimiento en cada endpoint del usuario del token enviado por la cabecera de autenticación. De esta forma no necesito enviar esta información en el cuerpo de la llamada y así ofrecer una respuesta adaptada al usuario que solicita la información.

En la imagen siguiente puede verse una llamada para recuperar los tickets de un usuario, pero solo pasamos el token. La API sabe que usuario es, que esta correctamente logado y, en consecuencia, le devuelve sus tickets.

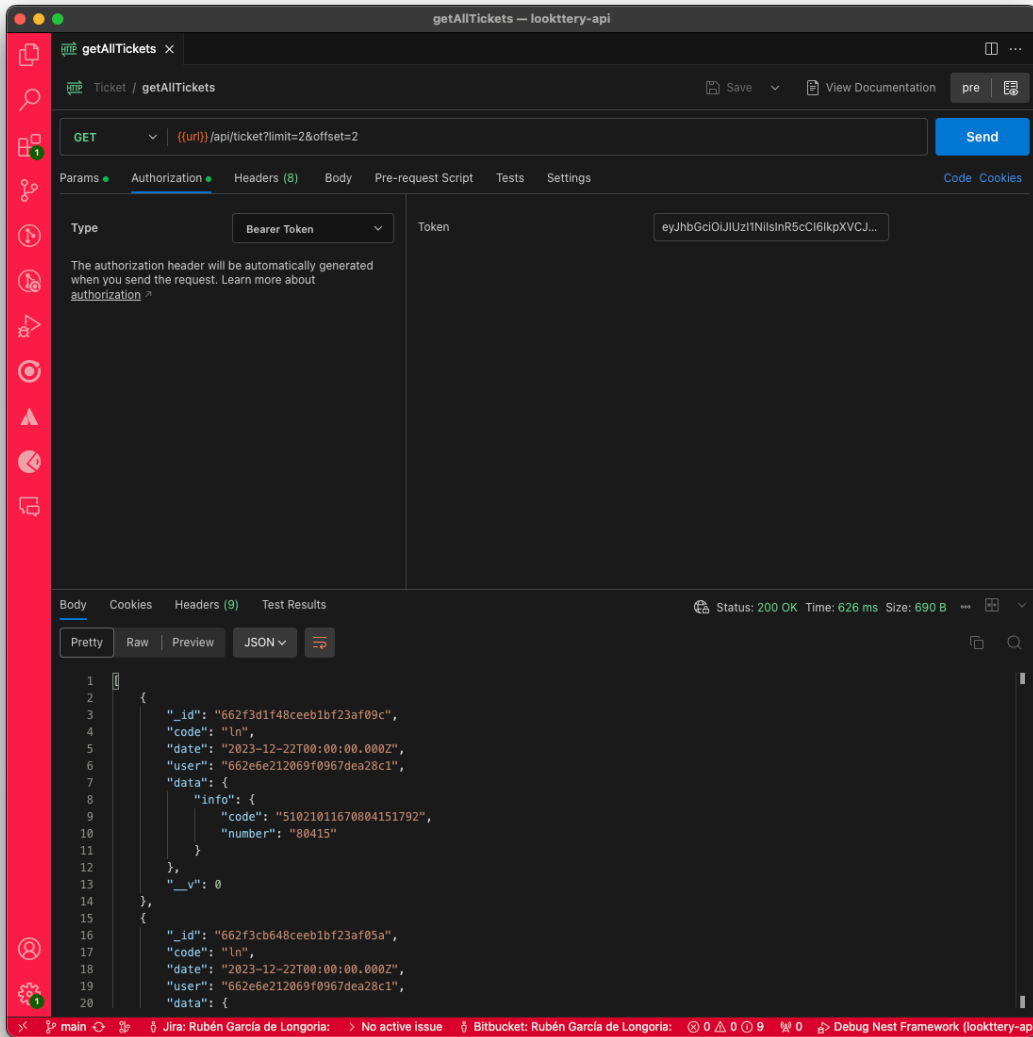


Ilustración 35 - Lookttery api ejemplo de llamada @Auth y @GetUser

6.1.6 – Tarea Cron para conseguir datos de los sorteos

Para que Lookttery sea una App competitiva tiene que disponer de la información de una forma veraz e inmediata. Para conseguirlo existe una tarea Cron que se ejecuta cada día de sorteo en las horas adecuadas, para conseguir la información en cuanto esté disponible, pero sin realizar un esfuerzo exagerado y que consuma demasiados recursos.

```
src > task > task.service.ts > ...
18 export class TaskService {
19     // ...
20     // ...
21     private hasWorking = false;
22     // ...
23     /*
24     LN variables
25     */
26     // ...
27     constructor(
28         @InjectModel(Game.name)
29         private readonly gameModel: Model<Game>,
30         @InjectModel(Ticket.name)
31         private readonly ticketModel: Model<Ticket>,
32         @InjectModel(Push.name)
33         private readonly pushModel: Model<Push>,
34         private readonly httpService: HttpService
35     ) {}
36     // ...
37     async onModuleInit() {
38         // await this.launchTaskLN();
39     }
40     // ...
41     /*
42     LN tasks
43     */
44     // ...
45     // Runs every 15 minutes the thursdays and saturdays
46     @Cron('0 */15 * * * 4,6', { name: 'handleIntervalLN'})
47     async handleIntervalLN() {
48         await this.launchTaskLN();
49     }
50     // ...
51     public async launchTaskLN() {
52         if (!process.env.RUN_TASKS && !this.hasWorking) {
53             this.logger.log('[LN]: Getting new info task started $(new Date().toISOString())...');
54             this.hasWorking = true;
55             await this.gettingInfoWithPrizesLN(); // Get new info with prizes
56             await this.gettingInfoWithoutPrizesLN(); // Get new info without prizes
57             this.hasWorking = false;
58             this.logger.log('[LN]: Getting new info task ended at $(new Date().toISOString())...');
59         }
60     }
61     // ...
62     // WITH PRIZES
63     private async gettingInfoWithPrizesLN() {
64         let latestFameIdWithPrizes = '';
65         let stop = false;
66         do {
67             const nextGameIdWithPrizes = await this.findNextGameIdWithPrizesLN();
68             stop = (latestFameIdWithPrizes === nextGameIdWithPrizes);
69             if (!cron) {
70                 // ...
71             }
72         } while (!stop);
73     }
74     // ...
75     // ...
76     // ...
77     // ...
78     // ...
79     // ...
80     // ...
81     // ...
82     // ...
83     // ...
84     // ...
85     // ...
86     // ...
87     // ...
88     // ...
89     // ...
90     // ...
91     // ...
92     // ...
93     // ...
94     // ...
95     // ...
96     // ...
97     // ...
98     // ...
99     // ...
100    // ...
101    // ...
102    // ...
103    // ...
104    // ...
105    // ...
106    // ...
107    // ...
108    // ...
109    // ...
110    // ...
111    // ...
112    // ...
113    // ...
114    // ...
115    // ...
116    // ...
117    // ...
118    // ...
119    // ...
120    // ...
121    // ...
122    // ...
123    // ...
124    // ...
125    // ...
126    // ...
127    // ...
128    // ...
129    // ...
130    // ...
131    // ...
132    // ...
133    // ...
134    // ...
135    // ...
136    // ...
137    // ...
138    // ...
139    // ...
140    // ...
141    // ...
142    // ...
143    // ...
144    // ...
145    // ...
146    // ...
147    // ...
148    // ...
149    // ...
150    // ...
151    // ...
152    // ...
153    // ...
154    // ...
155    // ...
156    // ...
157    // ...
158    // ...
159    // ...
160    // ...
161    // ...
162    // ...
163    // ...
164    // ...
165    // ...
166    // ...
167    // ...
168    // ...
169    // ...
170    // ...
171    // ...
172    // ...
173    // ...
174    // ...
175    // ...
176    // ...
177    // ...
178    // ...
179    // ...
180    // ...
181    // ...
182    // ...
183    // ...
184    // ...
185    // ...
186    // ...
187    // ...
188    // ...
189    // ...
190    // ...
191    // ...
192    // ...
193    // ...
194    // ...
195    // ...
196    // ...
197    // ...
198    // ...
199    // ...
200    // ...
201    // ...
202    // ...
203    // ...
204    // ...
205    // ...
206    // ...
207    // ...
208    // ...
209    // ...
210    // ...
211    // ...
212    // ...
213    // ...
214    // ...
215    // ...
216    // ...
217    // ...
218    // ...
219    // ...
220    // ...
221    // ...
222    // ...
223    // ...
224    // ...
225    // ...
226    // ...
227    // ...
228    // ...
229    // ...
230    // ...
231    // ...
232    // ...
233    // ...
234    // ...
235    // ...
236    // ...
237    // ...
238    // ...
239    // ...
240    // ...
241    // ...
242    // ...
243    // ...
244    // ...
245    // ...
246    // ...
247    // ...
248    // ...
249    // ...
250    // ...
251    // ...
252    // ...
253    // ...
254    // ...
255    // ...
256    // ...
257    // ...
258    // ...
259    // ...
260    // ...
261    // ...
262    // ...
263    // ...
264    // ...
265    // ...
266    // ...
267    // ...
268    // ...
269    // ...
270    // ...
271    // ...
272    // ...
273    // ...
274    // ...
275    // ...
276    // ...
277    // ...
278    // ...
279    // ...
280    // ...
281    // ...
282    // ...
283    // ...
284    // ...
285    // ...
286    // ...
287    // ...
288    // ...
289    // ...
290    // ...
291    // ...
292    // ...
293    // ...
294    // ...
295    // ...
296    // ...
297    // ...
298    // ...
299    // ...
300    // ...
301    // ...
302    // ...
303    // ...
304    // ...
305    // ...
306    // ...
307    // ...
308    // ...
309    // ...
310    // ...
311    // ...
312    // ...
313    // ...
314    // ...
315    // ...
316    // ...
317    // ...
318    // ...
319    // ...
320    // ...
321    // ...
322    // ...
323    // ...
324    // ...
325    // ...
326    // ...
327    // ...
328    // ...
329    // ...
330    // ...
331    // ...
332    // ...
333    // ...
334    // ...
335    // ...
336    // ...
337    // ...
338    // ...
339    // ...
340    // ...
341    // ...
342    // ...
343    // ...
344    // ...
345    // ...
346    // ...
347    // ...
348    // ...
349    // ...
350    // ...
351    // ...
352    // ...
353    // ...
354    // ...
355    // ...
356    // ...
357    // ...
358    // ...
359    // ...
360    // ...
361    // ...
362    // ...
363    // ...
364    // ...
365    // ...
366    // ...
367    // ...
368    // ...
369    // ...
370    // ...
371    // ...
372    // ...
373    // ...
374    // ...
375    // ...
376    // ...
377    // ...
378    // ...
379    // ...
380    // ...
381    // ...
382    // ...
383    // ...
384    // ...
385    // ...
386    // ...
387    // ...
388    // ...
389    // ...
390    // ...
391    // ...
392    // ...
393    // ...
394    // ...
395    // ...
396    // ...
397    // ...
398    // ...
399    // ...
400    // ...
401    // ...
402    // ...
403    // ...
404    // ...
405    // ...
406    // ...
407    // ...
408    // ...
409    // ...
410    // ...
411    // ...
412    // ...
413    // ...
414    // ...
415    // ...
416    // ...
417    // ...
418    // ...
419    // ...
420    // ...
421    // ...
422    // ...
423    // ...
424    // ...
425    // ...
426    // ...
427    // ...
428    // ...
429    // ...
430    // ...
431    // ...
432    // ...
433    // ...
434    // ...
435    // ...
436    // ...
437    // ...
438    // ...
439    // ...
440    // ...
441    // ...
442    // ...
443    // ...
444    // ...
445    // ...
446    // ...
447    // ...
448    // ...
449    // ...
450    // ...
451    // ...
452    // ...
453    // ...
454    // ...
455    // ...
456    // ...
457    // ...
458    // ...
459    // ...
460    // ...
461    // ...
462    // ...
463    // ...
464    // ...
465    // ...
466    // ...
467    // ...
468    // ...
469    // ...
470    // ...
471    // ...
472    // ...
473    // ...
474    // ...
475    // ...
476    // ...
477    // ...
478    // ...
479    // ...
480    // ...
481    // ...
482    // ...
483    // ...
484    // ...
485    // ...
486    // ...
487    // ...
488    // ...
489    // ...
490    // ...
491    // ...
492    // ...
493    // ...
494    // ...
495    // ...
496    // ...
497    // ...
498    // ...
499    // ...
500    // ...
501    // ...
502    // ...
503    // ...
504    // ...
505    // ...
506    // ...
507    // ...
508    // ...
509    // ...
510    // ...
511    // ...
512    // ...
513    // ...
514    // ...
515    // ...
516    // ...
517    // ...
518    // ...
519    // ...
520    // ...
521    // ...
522    // ...
523    // ...
524    // ...
525    // ...
526    // ...
527    // ...
528    // ...
529    // ...
530    // ...
531    // ...
532    // ...
533    // ...
534    // ...
535    // ...
536    // ...
537    // ...
538    // ...
539    // ...
540    // ...
541    // ...
542    // ...
543    // ...
544    // ...
545    // ...
546    // ...
547    // ...
548    // ...
549    // ...
550    // ...
551    // ...
552    // ...
553    // ...
554    // ...
555    // ...
556    // ...
557    // ...
558    // ...
559    // ...
560    // ...
561    // ...
562    // ...
563    // ...
564    // ...
565    // ...
566    // ...
567    // ...
568    // ...
569    // ...
570    // ...
571    // ...
572    // ...
573    // ...
574    // ...
575    // ...
576    // ...
577    // ...
578    // ...
579    // ...
580    // ...
581    // ...
582    // ...
583    // ...
584    // ...
585    // ...
586    // ...
587    // ...
588    // ...
589    // ...
590    // ...
591    // ...
592    // ...
593    // ...
594    // ...
595    // ...
596    // ...
597    // ...
598    // ...
599    // ...
600    // ...
601    // ...
602    // ...
603    // ...
604    // ...
605    // ...
606    // ...
607    // ...
608    // ...
609    // ...
610    // ...
611    // ...
612    // ...
613    // ...
614    // ...
615    // ...
616    // ...
617    // ...
618    // ...
619    // ...
620    // ...
621    // ...
622    // ...
623    // ...
624    // ...
625    // ...
626    // ...
627    // ...
628    // ...
629    // ...
630    // ...
631    // ...
632    // ...
633    // ...
634    // ...
635    // ...
636    // ...
637    // ...
638    // ...
639    // ...
640    // ...
641    // ...
642    // ...
643    // ...
644    // ...
645    // ...
646    // ...
647    // ...
648    // ...
649    // ...
650    // ...
651    // ...
652    // ...
653    // ...
654    // ...
655    // ...
656    // ...
657    // ...
658    // ...
659    // ...
660    // ...
661    // ...
662    // ...
663    // ...
664    // ...
665    // ...
666    // ...
667    // ...
668    // ...
669    // ...
670    // ...
671    // ...
672    // ...
673    // ...
674    // ...
675    // ...
676    // ...
677    // ...
678    // ...
679    // ...
680    // ...
681    // ...
682    // ...
683    // ...
684    // ...
685    // ...
686    // ...
687    // ...
688    // ...
689    // ...
690    // ...
691    // ...
692    // ...
693    // ...
694    // ...
695    // ...
696    // ...
697    // ...
698    // ...
699    // ...
700    // ...
701    // ...
702    // ...
703    // ...
704    // ...
705    // ...
706    // ...
707    // ...
708    // ...
709    // ...
710    // ...
711    // ...
712    // ...
713    // ...
714    // ...
715    // ...
716    // ...
717    // ...
718    // ...
719    // ...
720    // ...
721    // ...
722    // ...
723    // ...
724    // ...
725    // ...
726    // ...
727    // ...
728    // ...
729    // ...
730    // ...
731    // ...
732    // ...
733    // ...
734    // ...
735    // ...
736    // ...
737    // ...
738    // ...
739    // ...
740    // ...
741    // ...
742    // ...
743    // ...
744    // ...
745    // ...
746    // ...
747    // ...
748    // ...
749    // ...
750    // ...
751    // ...
752    // ...
753    // ...
754    // ...
755    // ...
756    // ...
757    // ...
758    // ...
759    // ...
760    // ...
761    // ...
762    // ...
763    // ...
764    // ...
765    // ...
766    // ...
767    // ...
768    // ...
769    // ...
770    // ...
771    // ...
772    // ...
773    // ...
774    // ...
775    // ...
776    // ...
777    // ...
778    // ...
779    // ...
780    // ...
781    // ...
782    // ...
783    // ...
784    // ...
785    // ...
786    // ...
787    // ...
788    // ...
789    // ...
790    // ...
791    // ...
792    // ...
793    // ...
794    // ...
795    // ...
796    // ...
797    // ...
798    // ...
799    // ...
800    // ...
801    // ...
802    // ...
803    // ...
804    // ...
805    // ...
806    // ...
807    // ...
808    // ...
809    // ...
810    // ...
811    // ...
812    // ...
813    // ...
814    // ...
815    // ...
816    // ...
817    // ...
818    // ...
819    // ...
820    // ...
821    // ...
822    // ...
823    // ...
824    // ...
825    // ...
826    // ...
827    // ...
828    // ...
829    // ...
830    // ...
831    // ...
832    // ...
833    // ...
834    // ...
835    // ...
836    // ...
837    // ...
838    // ...
839    // ...
840    // ...
841    // ...
842    // ...
843    // ...
844    // ...
845    // ...
846    // ...
847    // ...
848    // ...
849    // ...
850    // ...
851    // ...
852    // ...
853    // ...
854    // ...
855    // ...
856    // ...
857    // ...
858    // ...
859    // ...
860    // ...
861    // ...
862    // ...
863    // ...
864    // ...
865    // ...
866    // ...
867    // ...
868    // ...
869    // ...
870    // ...
871    // ...
872    // ...
873    // ...
874    // ...
875    // ...
876    // ...
877    // ...
878    // ...
879    // ...
880    // ...
881    // ...
882    // ...
883    // ...
884    // ...
885    // ...
886    // ...
887    // ...
888    // ...
889    // ...
890    // ...
891    // ...
892    // ...
893    // ...
894    // ...
895    // ...
896    // ...
897    // ...
898    // ...
899    // ...
900    // ...
901    // ...
902    // ...
903    // ...
904    // ...
905    // ...
906    // ...
907    // ...
908    // ...
909    // ...
910    // ...
911    // ...
912    // ...
913    // ...
914    // ...
915    // ...
916    // ...
917    // ...
918    // ...
919    // ...
920    // ...
921    // ...
922    // ...
923    // ...
924    // ...
925    // ...
926    // ...
927    // ...
928    // ...
929    // ...
930    // ...
931    // ...
932    // ...
933    // ...
934    // ...
935    // ...
936    // ...
937    // ...
938    // ...
939    // ...
940    // ...
941    // ...
942    // ...
943    // ...
944    // ...
945    // ...
946    // ...
947    // ...
948    // ...
949    // ...
950    // ...
951    // ...
952    // ...
953    // ...
954    // ...
955    // ...
956    // ...
957    // ...
958    // ...
959    // ...
960    // ...
961    // ...
962    // ...
963    // ...
964    // ...
965    // ...
966    // ...
967    // ...
968    // ...
969    // ...
970    // ...
971    // ...
972    // ...
973    // ...
974    // ...
975    // ...
976    // ...
977    // ...
978    // ...
979    // ...
980    // ...
981    // ...
982    // ...
983    // ...
984    // ...
985    // ...
986    // ...
987    // ...
988    // ...
989    // ...
990    // ...
991    // ...
992    // ...
993    // ...
994    // ...
995    // ...
996    // ...
997    // ...
998    // ...
999    // ...
1000   // ...

```

Ilustración 36 - - Lookttery api tarea Cron

Esta tarea lo único que hace es llamar a los mismos servicios existentes de forma pública en la web de Lotería Nacional, como si fuese un usuario más que realiza una consulta. Una vez recopilamos la información la adaptamos a nuestra colección games de la base de datos.

6.1.7 – Despliegue en producción

El despliegue en Railway se hace de forma automática con cada subida de código a la rama master del repositorio, que está alojado en GitHub.

Railway tiene permisos de acceso a este repositorio y cada vez que detecta una nueva subida despliega el código en una nueva instancia y, solo cuando esta instancia ha arrancado correctamente, detiene la instancia anterior. De esta forma no existen tiempos de desconexión.

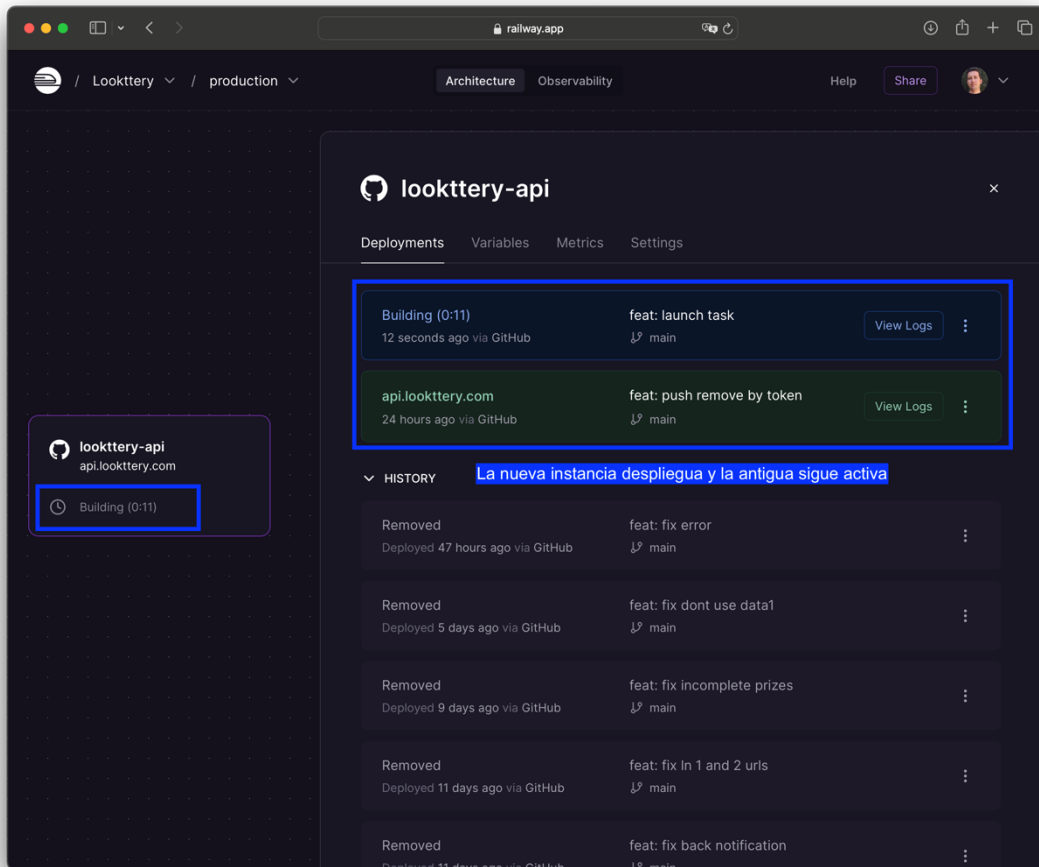


Ilustración 37 - Lookttery api puesta en producción

6.2 – Proyecto lookttery-app

Denominado también como el proyecto web o micro-frontal, este es el proyecto con la lógica de negocio de Lookttery.

Esta realizado con el framework Angular en su versión 17 y utiliza componentes visuales de Ionic para la interfaz de usuario.

6.2.1 – Ejecución en local

Los pasos básicos para utilizar el proyecto están descritos en el fichero readme.md del proyecto.

Lo primero que debemos hacer es instalar las dependencias mediante el comando **npm i**.

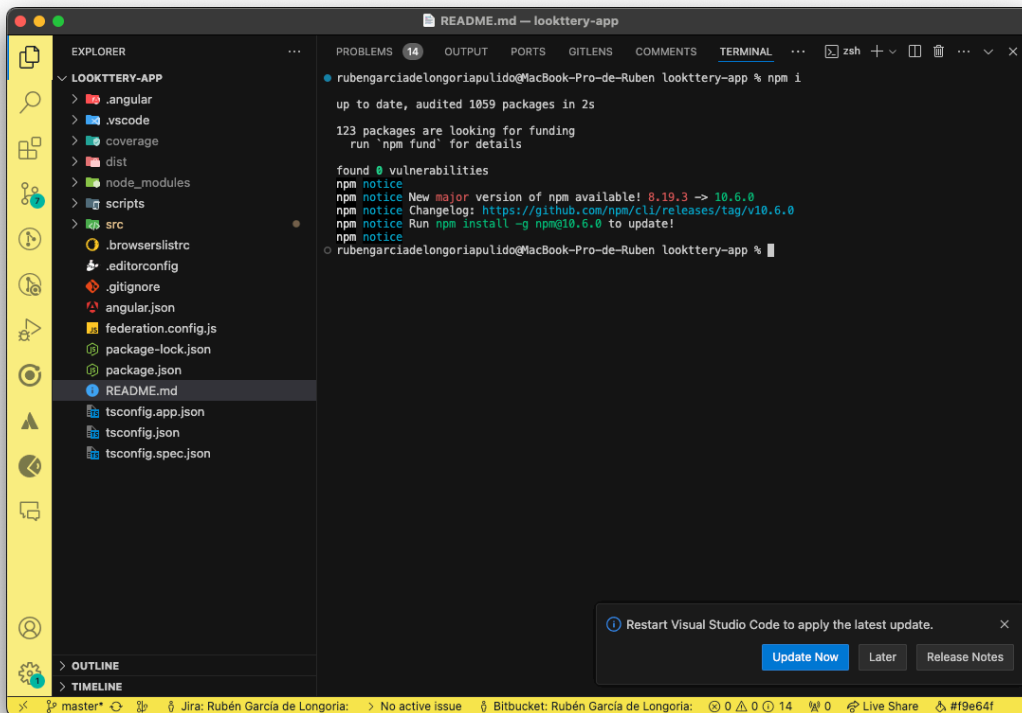


Ilustración 38 - Lookttery app instalación de dependencias

El siguiente paso a realizar es el de configurar los ficheros de variables de entorno dependiendo de donde queramos lanzar la ejecución. El proyecto está preparado para lanzarse en el puerto 4201 y la API Rest en el puerto 3000, pero tenemos que definir la URL en ambos casos. Dependiendo de si vamos a ejecutar el micro-frontal dentro de una Shell en un emulador Android, simulador iOS, navegador Web o dispositivo Real deberemos elegir la dirección IP adecuada.

web.host	
Dispositivo	URL
Emulador Android	http://10.0.2.2:4201
Simulador iOS	http://localhost:4201
Navegador web	http://localhost:4201
Dispositivo Real	http://192.168.1.40:4201
Producción	http://lookttery.com

api.host	
Dispositivo	URL
Emulador Android	http://10.0.2.2:3000/api
Simulador iOS	http://localhost:3000/api
Navegador web	http://localhost:3000/api

Dispositivo Real	http://192.168.1.40:3000/api
Producción	https://api.lookttery.com/api

Una vez realizada esta configuración básica podemos ejecutar el proyecto mediante el comando **ng serve**.

Si el proceso finaliza exitosamente veremos una pantalla similar a la que se muestra en la siguiente imagen.

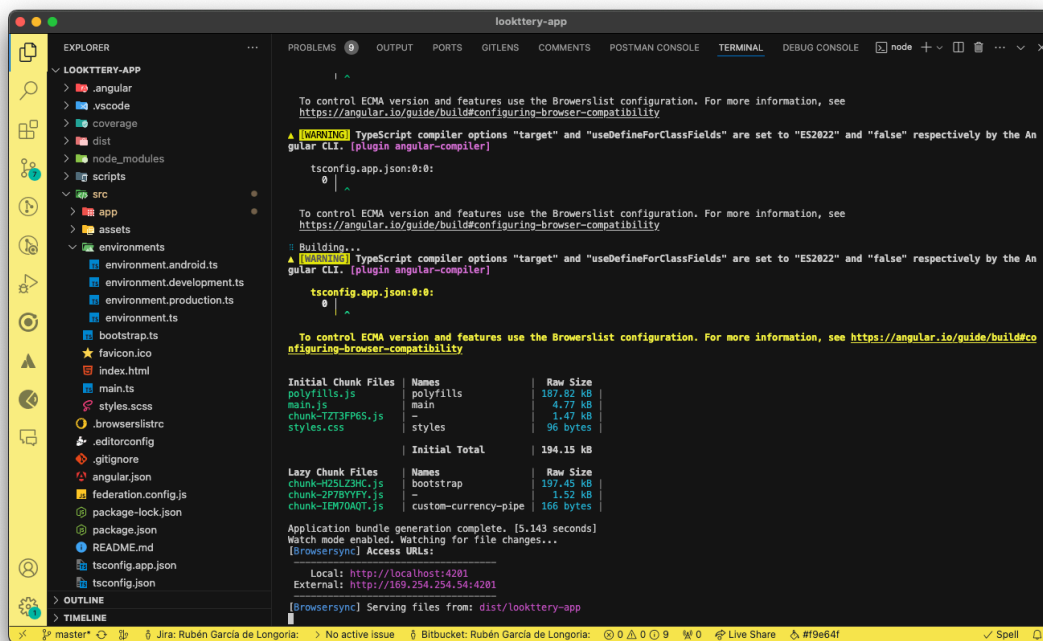


Ilustración 39 - Lookttery app ejecución en local

Como norma general el navegador configurado por defecto abrirá en una pestaña la dirección <http://localhost:4201> y se mostrará el proyecto ejecutándose.

Si hemos configurado una URL local y nuestra API no está funcionando el proyecto se abrirá, pero no funcionará al no poder realizar las llamadas necesarias.

Es importante que, si estamos en un entorno local, primero se ejecute el proyecto lookttery-api en local para disponer de la API Rest funcionando.

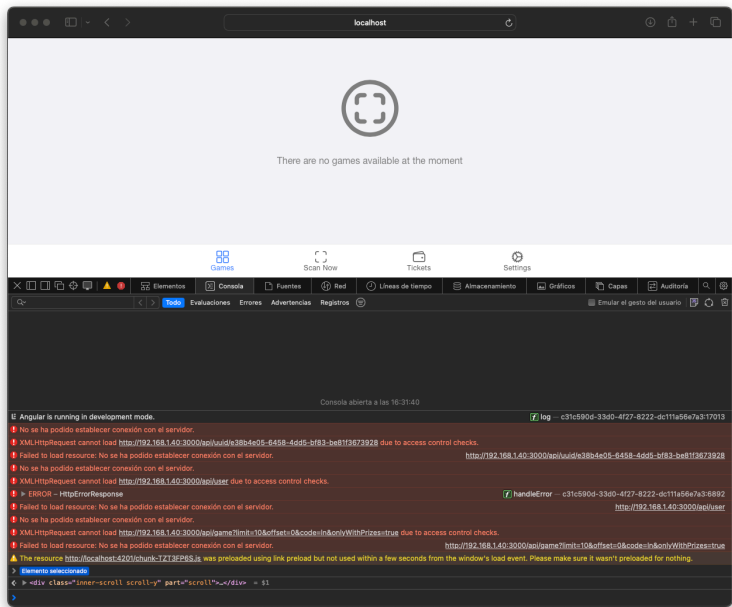


Ilustración 40 - - Looktty App ejecución en local sin API funcionando

6.2.2 – Herramientas de debug

Durante el desarrollo es imprescindible contar con unas herramientas de debug adecuadas. En este caso, la gran ventaja de trabajar la lógica de negocio con micro-frontales es que se trabaja de forma similar a como si lo hiciésemos en una web. Para hacer debug podemos usar las herramientas de inspección de nuestro navegador. Personalmente suelo utilizar las Chrome Dev Tools, aunque para casos concretos de iOS necesitaremos trabajar con el entorno de desarrollador de Safari.

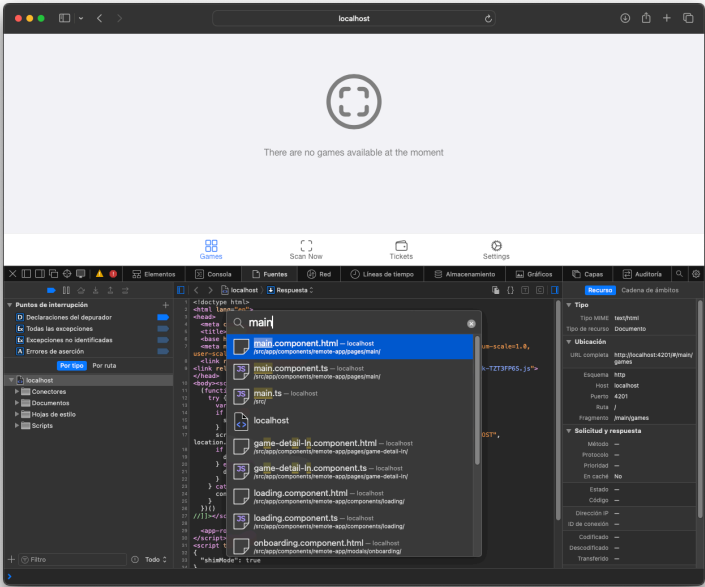


Ilustración 41 - Looktty app debug safari

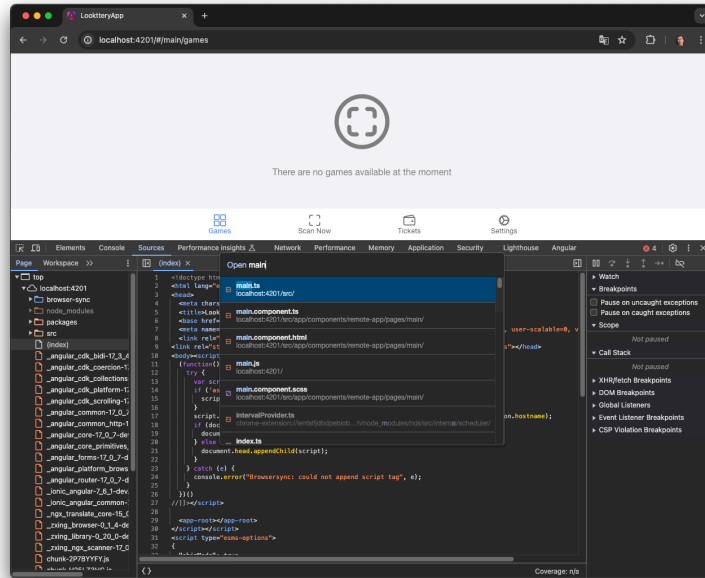


Ilustración 42 - Lookttery app debug Chrome

6.2.3 – Estructura del proyecto

La finalidad de trabajar con un proyecto como micro-frontal es la de poder aprovechar la carga en remoto del proyecto desde una App y así poder realizar actualizaciones de código en tiempo real sin pasar por las tiendas oficiales. Para conseguir esto necesitamos tener una estructura de proyecto concreta, no funciona un proyecto Angular cualquiera, se necesita trabajar de una determinada manera y readaptar la forma de trabajar con algunas funcionalidades habituales de Angular.

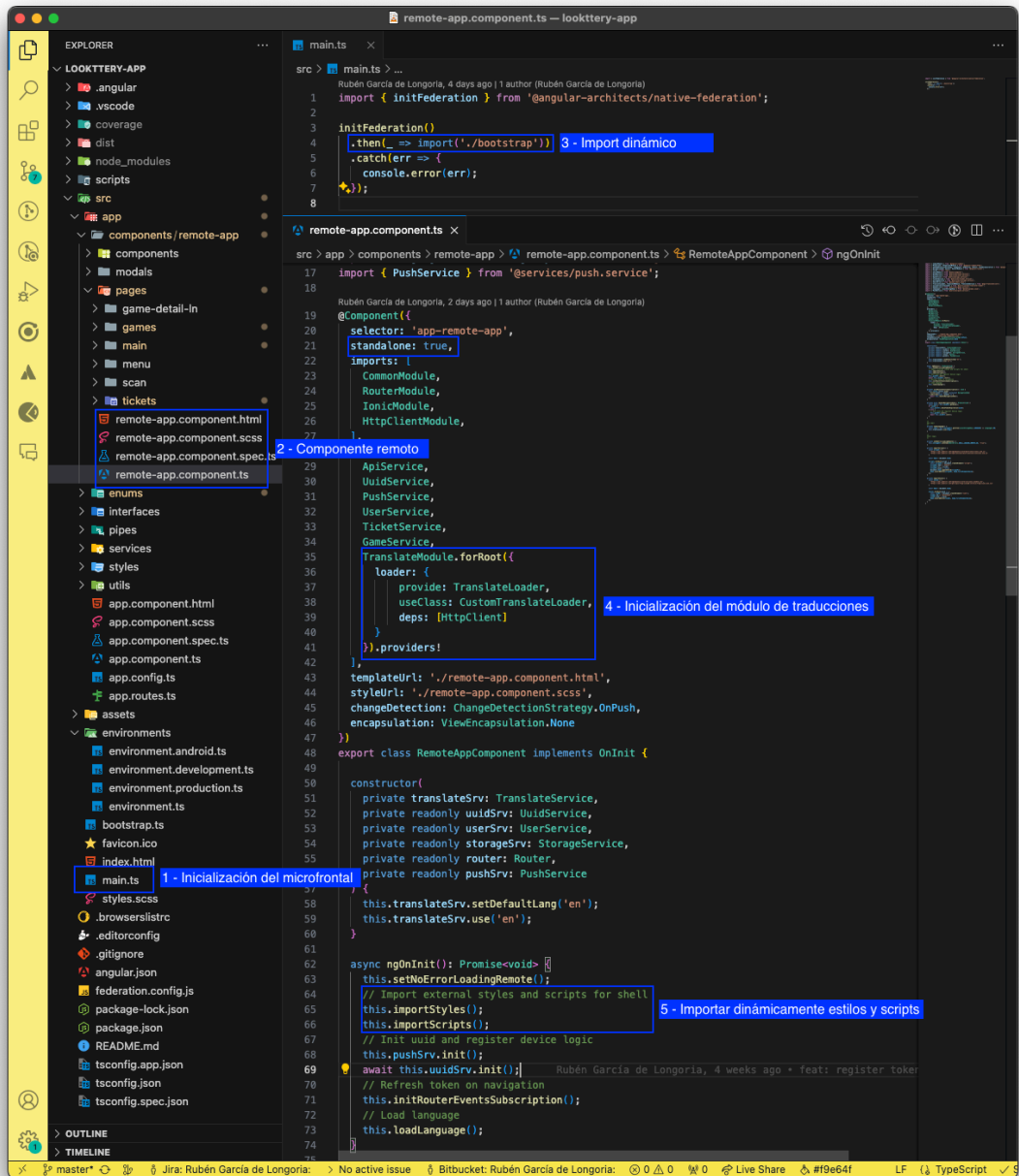


Ilustración 43 - Lookttery app estructura principal

En la imagen anterior se ven algunos de los elementos principales de la estructura especial que debe tener este proyecto. He resaltado los más importantes.

El fichero main.ts es donde se inicializa la lógica del micro-frontend por parte del plugin de Native Federation y, si todo ha ido bien, arranca una instancia de la aplicación Angular y representa un componente independiente, en este caso el AppComponent, tal como indica en los comentarios oficiales del código de Angular.

```

export declare type ApplicationConfig = ApplicationConfig_2;

/**
 * Bootstraps an instance of an Angular application and renders a standalone component as the
 * application's root component. More information about standalone components can be found in [this
 * guide](guide/standalone-components).
 */

```

Ilustración 44 - Lookttery app explicación oficial del método bootstrapApplication

En el código del fichero maint.ts he resaltado la línea 4 en la que se realiza el import dinámico del fichero bootstrap.ts, una línea tan importante como problemática, ya que **esta funcionalidad está disponible solo a partir de ES2020** y no funciona en versiones navegadores que, aunque son antiguas, su uso sigue muy extendido.

Esto limita las versiones en las que Lookttery puede funcionar. Después de realizar muchas pruebas y, teniendo en cuenta que los navegadores pueden implementar parte de las funcionalidades incluidas en ES2020, pero pueden no incluir justamente esta, he podido acotar que Lookttery no puede funcionar en instancias de **navegadores Chrome inferiores a la versión 91** ni en versiones de Safari vinculadas a **sistemas operativos inferiores a iOS 16.4**.

En Android Chrome puede instalarse y actualizarse a su última versión desde Android 8, pero en esta versión suele venir Chrome preinstalado con una versión cercana a la 60, por lo que Lookttery no funcionará hasta que se actualice Chrome en el dispositivo.

Para hacer más amigable este inconveniente a los usuarios que quieran usar Lookttery, he desarrollado una lógica que, en caso de detectar un error con la interpretación del código JavaScript, mostrará un mensaje avisando de este problema y la única solución de actualizar en la medida de lo posible el navegador o el sistema operativo.

Este mensaje es una URL externa (<https://lookttery.com/error/error.html>) por lo que podré actualizarla para que muestre el mensaje de una forma más amigable en el futuro.

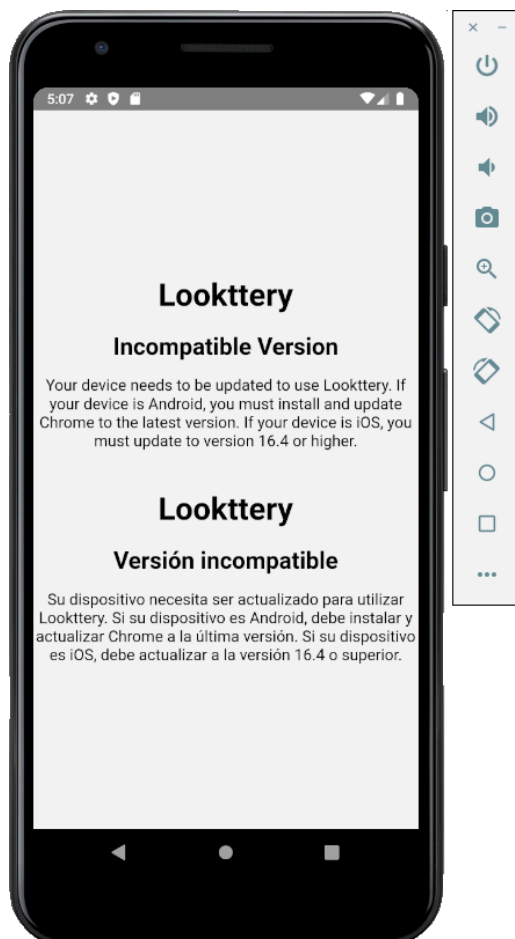


Ilustración 45 - Lookttery app error version ECMAScript

Otra parte importante de la estructura especial del proyecto es la existencia de un componente llamado remote-app, y que se encuentra alojado en el directorio src/app/components y que representa el componente base del proyecto. Este componente define los providers de forma

general, carga el módulo de traducciones, importa los estilos y scripts de Ionic de forma dinámica cuando el micro-frontal ya ha arrancado y realiza la lógica básica inicial de Lookttery.

El componente habitual para realizar estas acciones sería el AppComponent, pero en este caso, debido a la forma particular de nuestra arquitectura, este componente carece de valor.

6.2.4 - Exposición de rutas desde el micro-frontal

Esta es la clave que me permite aprovechar la carga dinámica del micro-frontal pudiendo tener un proyecto web o una App, con el mismo código, sin duplicidades y sin complicar el proyecto injustificadamente.

Con Native Federation los micro-frontales pueden exponer funcionalidad de forma concreta, por ejemplo, exponer unos componentes, unos módulos, unos servicios o unas rutas para que sean utilizados por la Shell o por otros micro-frontales. También pueden compartir dependencias, eventos, servicios o el propio almacenamiento local, sesiones o cookies.

En el caso de Lookttery, el micro-frontal solo expone las rutas del proyecto y, mediante eventos y almacenamiento local se comunicará con la Shell. De esta forma el micro-frontal puede funcionar por sí mismo como un proyecto Angular o dentro de una Shell.

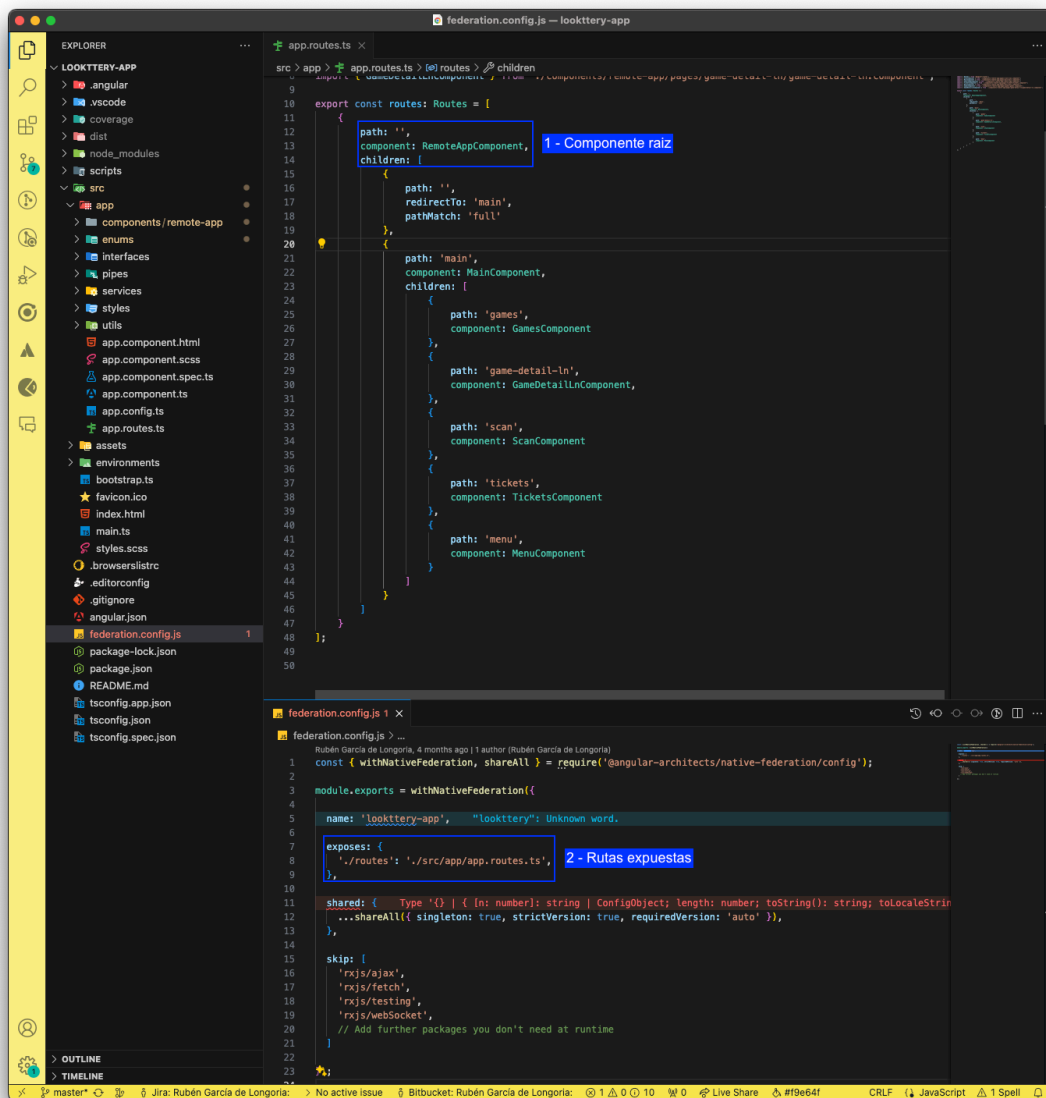


Ilustración 46 - Lookttery app rutas expuestas

Como se ve en la imagen anterior, el fichero de rutas utiliza como ruta base el componente `remoteApp` en vez del `AppComponent` y el resto de las rutas son rutas hijas. Con este planteamiento se puede entender que toda la funcionalidad existente en el directorio `src` será expuesto y podrá utilizarse dentro de un Shell, pero todo lo que esté fuera solo funcionará cuando se use el proyecto de forma independiente, pero no existirá cuando se use dentro de una Shell. Esto nos plantea al menos dos problemas que he encontrado en mi desarrollo, los interceptores y el uso de assets.

6.2.5 - Solución de problemas con interceptores y assets

En Angular 17 los módulos no son necesarios y Lookttery utiliza componentes independientes conocidos como standalone. Esto implica que los interceptores se implementan a nivel de proyecto fuera de la carpeta `src` y por tanto no se exponen cuando se utiliza dentro de una Shell.

Para solucionar este problema he trasladado la funcionalidad de los interceptores a un servicio propio que, si se encuentra dentro del directorio `src`, este servicio se llama `api.service` y ofrece una capa a modo de fachada entre el código y el uso de la librería `HttpClient` de Angular. De esta forma puedo controlar las llamadas antes de que se realicen y añadir cabeceras en ciertas circunstancias de igual manera que hace un interceptor.

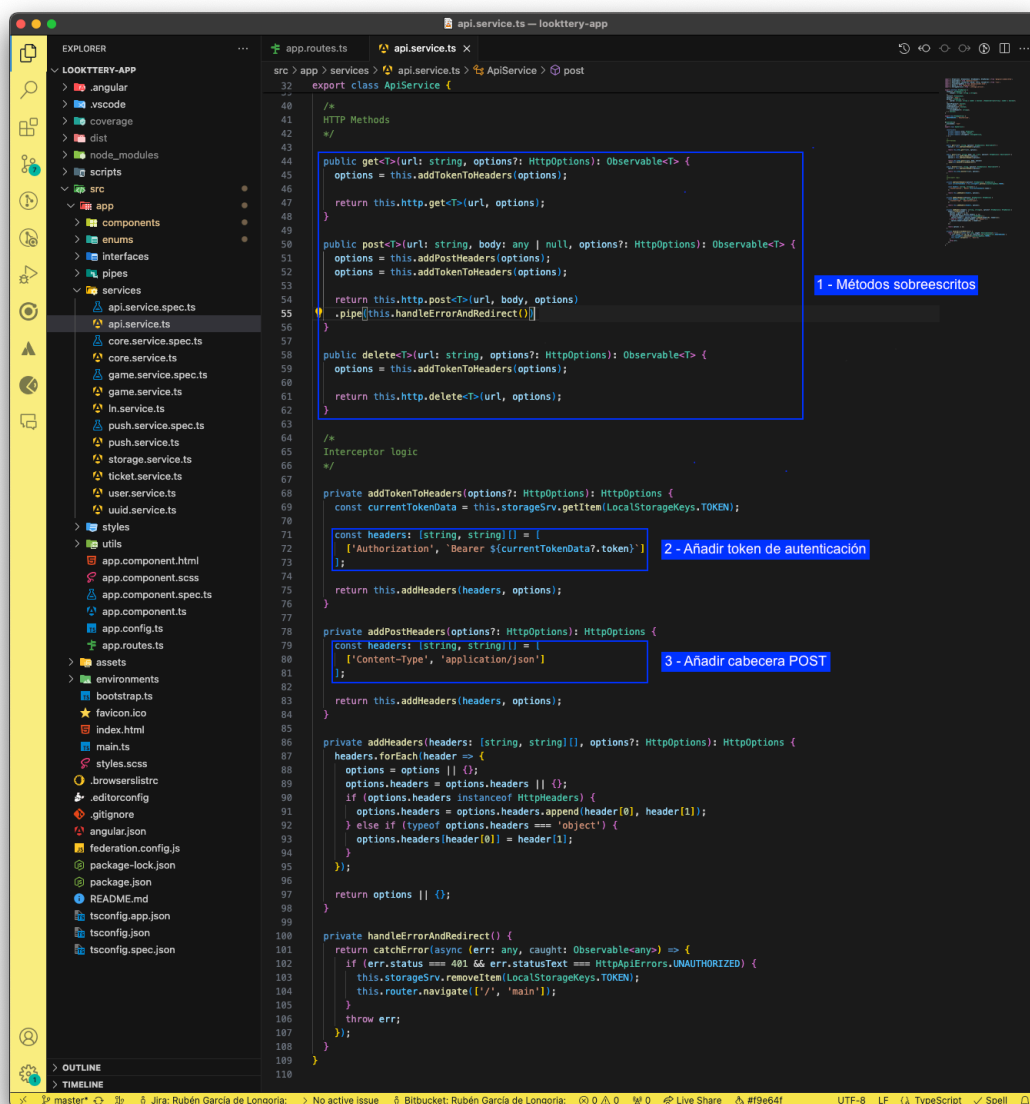


Ilustración 47 - Lookttery app api service

Para solucionar el problema de los recursos multimedia, assets he utilizado la variable global `web.host` que contiene la URL de nuestro micro-frontal y lo utilizo cada vez que enlazo el path de un recurso asset en un elemento HTML, de esta forma la ruta relativa se convierte en una ruta absoluta a nuestro micro-frontal y, en caso de estar en la Shell accede a los recursos del micro-frontal en vez de a los recursos de la Shell que, de otro modo, deberían estar duplicados y perderíamos la capacidad de añadir nuevos recursos en caliente sin pasar por las tiendas oficiales.

```
<section class="game">
  <article class="row">
    <img class="game-img"
      aria-hidden="true" slot="start" alt=""
      [src]='environment.web.host' + '/assets/logos/games/' + data.code + '.png'>
    <section class="game-data">
      <span class="game-data-date" color="medium">{{ data.date | date | async }}</span>
    </section>
  </article>
</section>
```

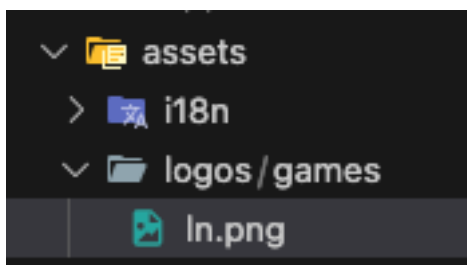


Ilustración 48 - Lookttery app uso de assets

6.2.6 – Despliegue en producción

El micro-frontal debe publicarse de forma similar a como se publica una web, en un hosting adecuado para ello. Las instrucciones están incluidas en el fichero `readme.md` del proyecto.

Previamente deben generarse los archivos de distribución finales mediante el comando ***npm run build-production***, este comando lanza un script personalizado que puede verse en el fichero `package.json` en el apartado de scripts.

```
"scripts": {
  "ng": "ng",
  "start": "ng serve",
  "build": "ng build",
  "watch": "ng build --watch --configuration development",
  "test": "ng test",
  "test-pro": "ng test --watch --code-coverage",
  "update-dist-files": "node scripts/updateDistFiles.js",
  "rename-dist-files": "node scripts/renameDistFiles.js",
  "build-production": "ng build --configuration production && npm run update-dist-files && npm run rename-dist-files"
},
```

Ilustración 49 - Lookttery app scripts

El script en primer lugar construye el proyecto con el parámetro `production`, que se encarga de que la construcción este optimizada, que no incluya los ficheros `.map` utilizados para hacer debug durante el desarrollo y actualiza las variables de entorno incluidas en el fichero `environment.ts` por las del fichero del mismo directorio llamado `environment.production.ts`.

```

},
"configurations": {
  "production": {
    "budgets": [
      {
        "type": "initial",
        "maximumWarning": "500kb",
        "maximumError": "1mb"
      },
      {
        "type": "anyComponentStyle",
        "maximumWarning": "2kb",
        "maximumError": "8kb"
      }
    ],
    "optimization": true,
    "extractLicenses": true,
    "sourceMap": false,
    "outputHashing": "all",
    "fileReplacements": [
      {
        "replace": "src/environments/environment.ts",
        "with": "src/environments/environment.production.ts"
      }
    ]
  }
},
}

```

Ilustración 50 - Lookttery app configuración de producción

Una vez terminan los procesos de generación del proyecto en modo producción se realizan dos tareas adicionales y necesarias para publicar el proyecto en el hosting que he elegido que es en forma de github-page.

El script update-dist-files se encarga de actualizar el contenido de los ficheros importmap.json y remoteEntry.json para eliminar las barras bajas que aparecen como prefijo en los nombres de los ficheros de las rutas que contienen estos ficheros.

El script rename-dist-files renombra los ficheros generados para que no empiecen por la barra baja.

Esto es necesario porque en este hosting los ficheros que comienzan por una barra baja son considerados ficheros privados y, cuando se accede mediante el dominio www.lookttery.com a ellos no son encontrados en las rutas de estos ficheros y ocurre un error. Básicamente estoy convirtiendo los ficheros en ficheros públicos.

Una vez finalizados todos estos procesos, veremos una pantalla similar a la de la imagen.

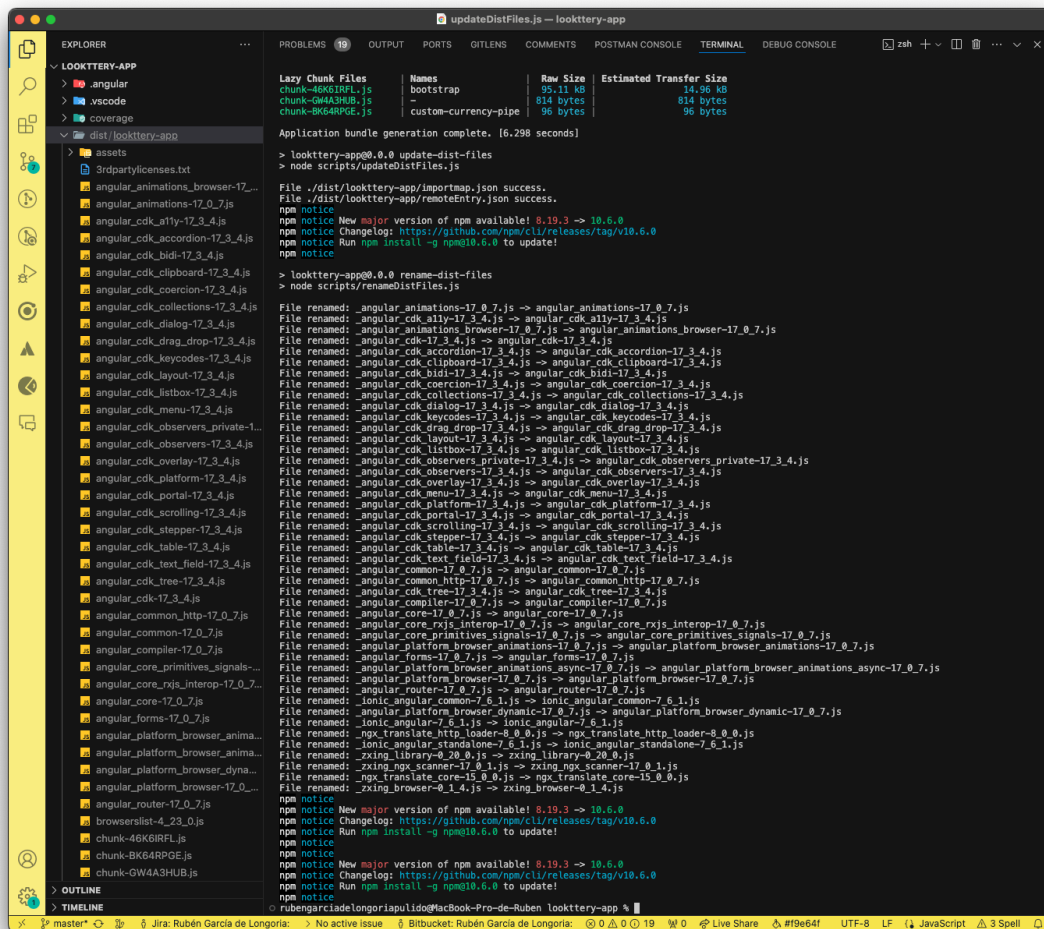


Ilustración 51 - Lookttery app generación de ficheros para distribución

Ahora el directorio `dist/lookttery-app` contiene todos los ficheros que deben publicarse como una github-page. Para ello he creado otro proyecto, por comodidad y privacidad, llamado `lookttery-static`. Las github-page requieren que el repositorio de código al que se vinculan sea público y mi código se aloja en un repositorio privado.

El proyecto `lookttery-static` es un proyecto muy sencillo que solo contiene los ficheros del directorio `dis/lookttery-app` y, con cada subida de código a la rama `master`, actualizará la github-page.

La url de este repositorio es: <https://github.com/rgarciadelongoria/lookttery-static>

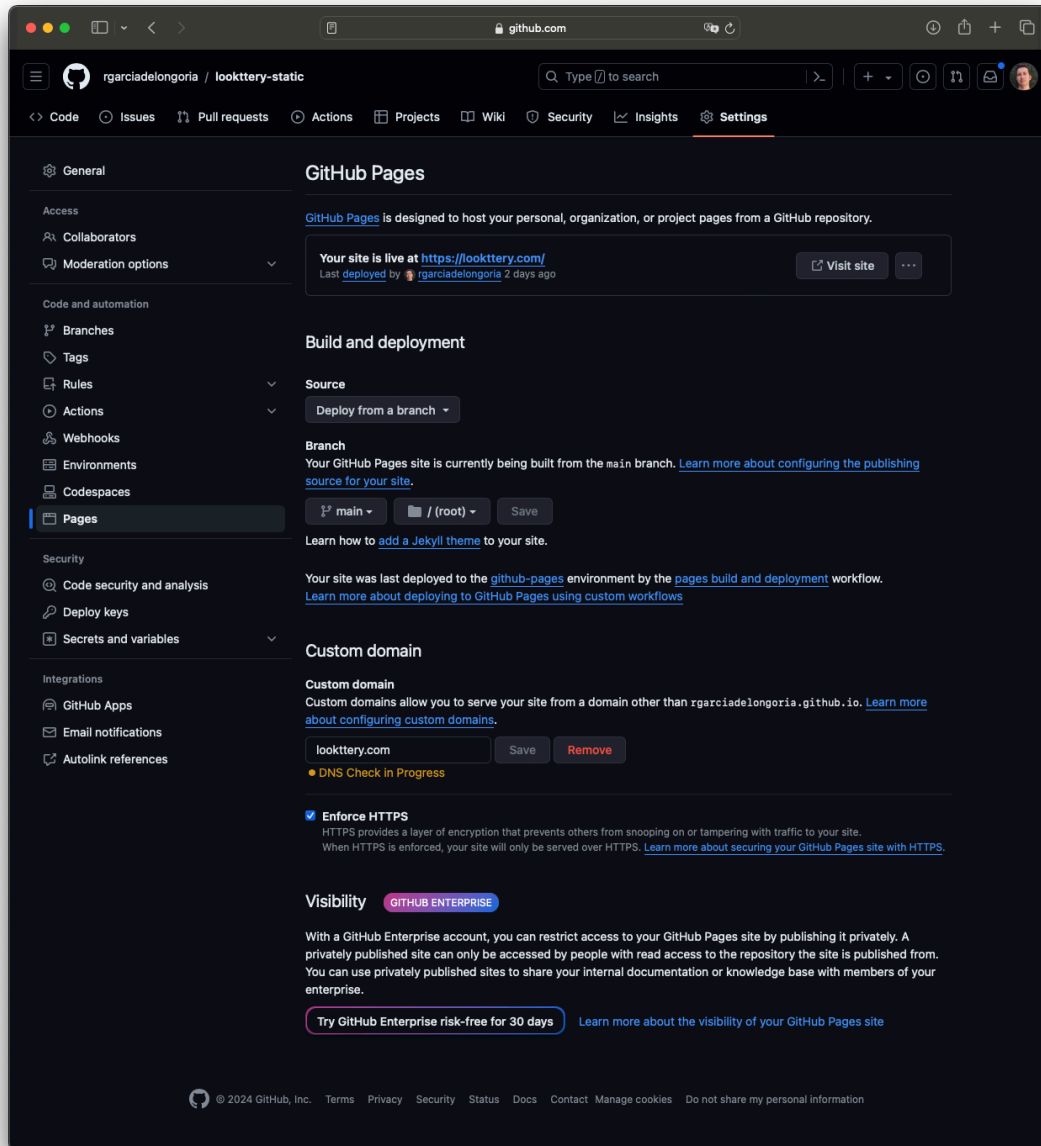


Ilustración 52 - Lookttery app configuración github-page

Como se ve en la imagen anterior, la github-page está asociada al dominio www.lookttery.com donde se expone el fichero remoteEntry.json, que es el fichero que la Shell descargará para obtener el código del micro-frontal.


```
{
  "name": "lookttery-app",
  "shared": [
    {
      "packageName": "@angular/animations",
      "outFileName": "angular_animations-17_0_7.js",
      "requiredVersion": "17.0.7",
      "singleton": true,
      "strictVersion": true,
      "version": "17.0.7"
    },
    {
      "packageName": "@angular/animations/browser",
      "outFileName": "angular_animations_browser-17_0_7.js",
      "requiredVersion": "17.0.7",
      "singleton": true,
      "strictVersion": true,
      "version": "17.0.7"
    },
    {
      "packageName": "@angular/cdk",
      "outFileName": "angular_cdk-17_3_4.js",
      "requiredVersion": "^17.3.1",
      "singleton": true,
      "strictVersion": true,
      "version": "17.3.4"
    },
    {
      "packageName": "@angular/cdk/ally",
      "outFileName": "angular_cdk_ally-17_3_4.js",
      "requiredVersion": "^17.3.1",
      "singleton": true,
      "strictVersion": true,
      "version": "17.3.4"
    },
    {
      "packageName": "@angular/cdk/accordion",
      "outFileName": "angular_cdk_accordion-17_3_4.js",
      "requiredVersion": "^17.3.1",
      "singleton": true,
      "strictVersion": true,
      "version": "17.3.4"
    },
    {
      "packageName": "@angular/cdk/bidi",
      "outFileName": "angular_cdk_bidi-17_3_4.js",
      "requiredVersion": "^17.3.1",
      "singleton": true,
      "strictVersion": true,
      "version": "17.3.4"
    },
    {
      "packageName": "@angular/cdk/clipboard",
      "outFileName": "angular_cdk_clipboard-17_3_4.js",
      "requiredVersion": "^17.3.1",
      "singleton": true,
      "strictVersion": true,
      "version": "17.3.4"
    },
    {
      "packageName": "@angular/cdk/coercion",
      "outFileName": "angular_cdk_coercion-17_3_4.js",
      "requiredVersion": "^17.3.1",
      "singleton": true,
      "strictVersion": true,
      "version": "17.3.4"
    },
    {
      "packageName": "@angular/cdk/collections",
      "outFileName": "angular_cdk_collections-17_3_4.js",
      "requiredVersion": "^17.3.1",
      "singleton": true,
      "strictVersion": true,
      "version": "17.3.4"
    },
    {
      "packageName": "@angular/cdk/dialog",
      "outFileName": "angular_cdk_dialog-17_3_4.js",
      "requiredVersion": "^17.3.1",
      "singleton": true,
      "strictVersion": true,
      "version": "17.3.4"
    },
    {
      "packageName": "@angular/cdk/drag-drop",
      "outFileName": "angular_cdk_drag_drop-17_3_4.js",
      "requiredVersion": "^17.3.1",
      "singleton": true,
      "strictVersion": true,
      "version": "17.3.4"
    }
  ]
}
```

Ilustración 53 - Lookttery app fichero remoteEntry.json en el navegador

Aunque no es obligatorio ni es la finalidad de mi proyecto, solo por añadir el fichero index.html a la github-page, he conseguido la versión web de Lookttery, es un mismo código, sin duplicidades que mantener y totalmente funcional.

Puede que en un futuro se interesante tener una versión web, así que, he adaptado ligeramente el código para que pueda funcionar igual que en la App, incluso puede sincronizarse mediante el código QR, escanear tickets y, en definitiva, funcionar igual que la App desde el navegador. Estas claves se explicarán mejor en el próximo punto que trata del proyecto Shell.



Ilustración 54 - Lookttery app como web

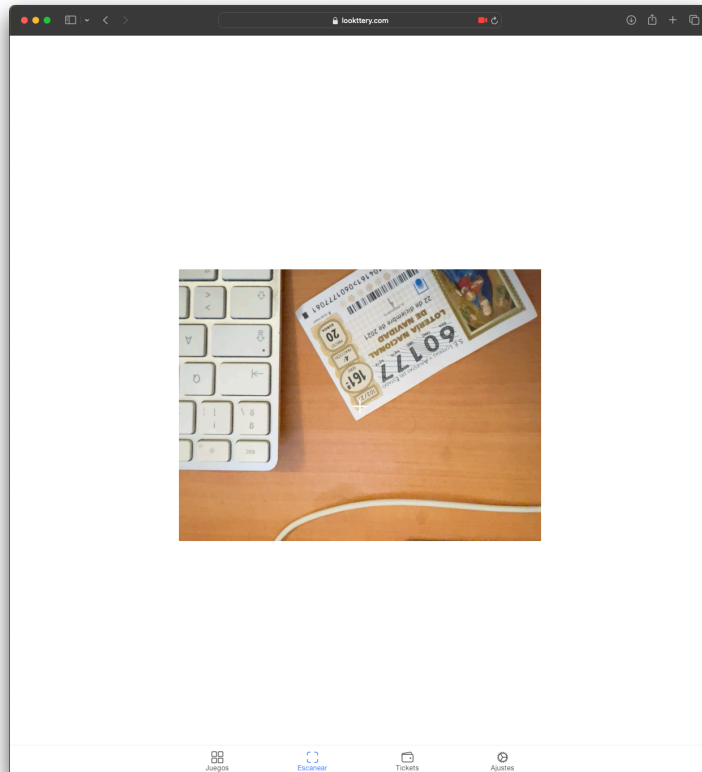


Ilustración 55 - Lookttery app escaneando desde la web

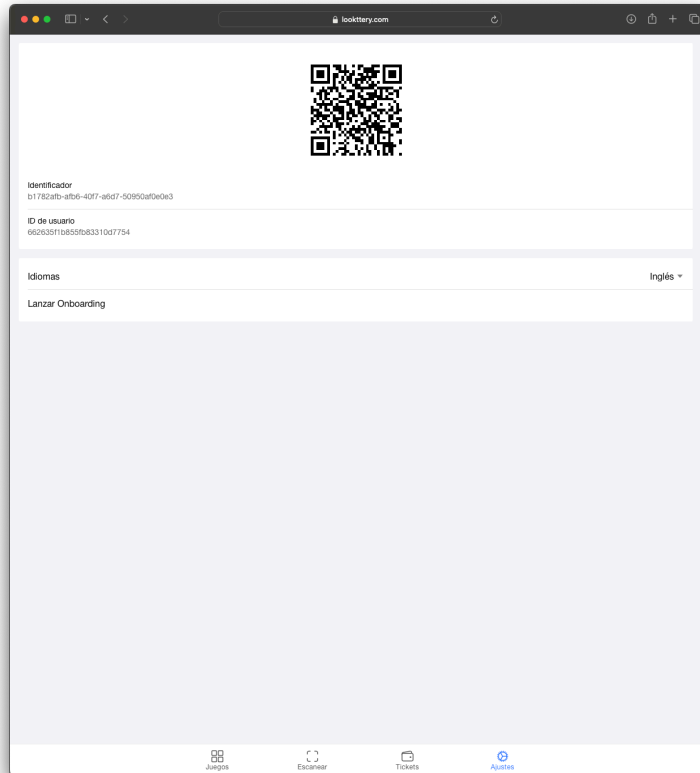


Ilustración 56 - Lookttery app ajustes desde la web, posibilidad de sincronizar

6.3 – Proyecto core-shell

Este proyecto se conoce también como Shell y su funcionamiento se puede comparar al de una carcasa capaz de cargar el proyecto lookttery-app y dotarlo de funcionalidad adicional. Se compone de un proyecto Angular 17 que, mediante Capacitor, se convierte en una App para Android y iOS.

El proyecto Shell ha sido diseñado la forma más genérica posible para poder reutilizarlo como si fuese un prototipo en otros proyectos.

Este proyecto es el que se compilara como una App y será el que se suba a las tiendas oficiales. Actualmente en su versión MVP ofrece lógica nativa para que el micro-frontal pueda obtener información del dispositivo, trabajar con la cámara y el escáner de códigos de barras, además de poder utilizar notificaciones Push.

Este proyecto comparte el almacenamiento local y los eventos con el micro-frontal, será mediante estos elementos como se realice la comunicación entre ambos proyectos.

6.3.1 – Ejecución en local

Las instrucciones a seguir para ejecutar el proyecto en local están recogidas en el fichero readme.md del proyecto.

El primer paso es instalar las dependencias mediante el comando ***npm i*** desde el terminal. A diferencia de los proyectos lookttery-api y lookttery-app, después de la instalación de dependencias se realiza el comando ***prepare*** que construye y sincroniza el proyecto para Android y iOS.

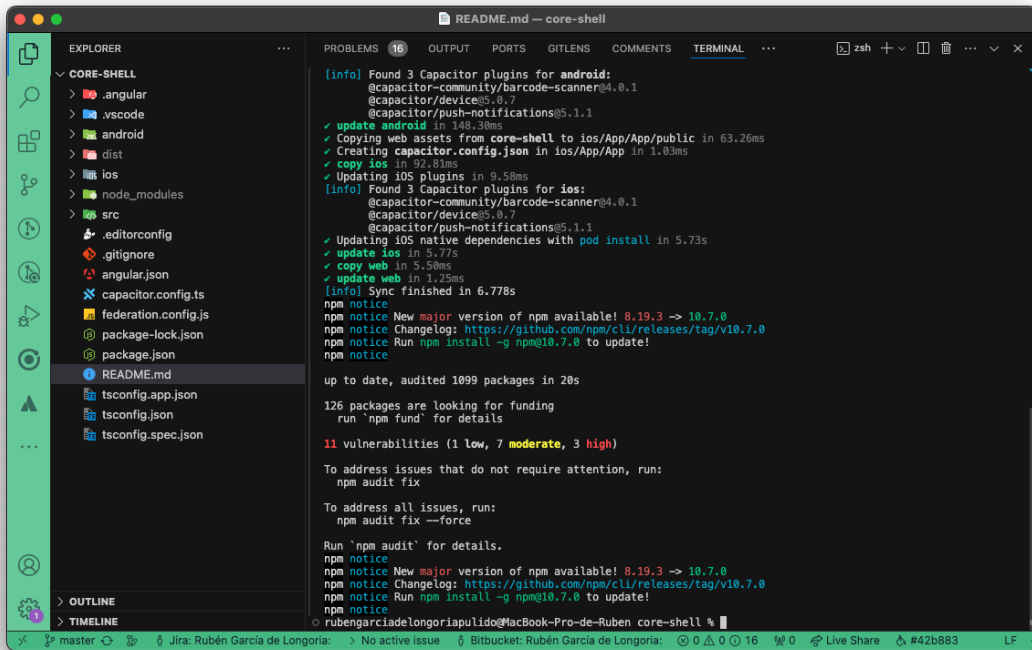


Ilustración 57 - Core shell instalación de dependencias

El siguiente paso es configurar en el fichero federation.manifest.json la ruta del fichero remoteEntry.json del micro-frontal. De esta forma el Shell es capaz de descargar el micro-frontal cuando arranque.

Por lo general las urls son similares a las comentadas en el proyecto lookttery-app, dependiendo de si nuestra App va a ejecutarse en un emulador Android, un simulador iOS, un navegador web o un dispositivo real Android o iOS.

federation.manifest.json	
Emulador Android	http://10.0.2.2:4201/remoteEntry.json
Simulador iOS	http://localhost:4201/remoteEntry.json
Navegador web	http://localhost:4201/remoteEntry.json
Dispositivo Real	http://192.168.1.40:4201/remoteEntry.json
Producción	https://api.lookttery.com/remoteEntry.json

Por lo general siempre vamos a asegurar la disponibilidad del proyecto lookttery-app en producción, no obstante, pueden surgir problemas como una mala conexión o una ausencia de acceso a Internet, en este caso la App nos mostrará un mensaje avisando de este problema y permitiendo reintentar la carga del proyecto lookttery-app, en una versión posterior al MVP tengo pensado mejorar este sistema para ofrecer más soluciones al usuario.

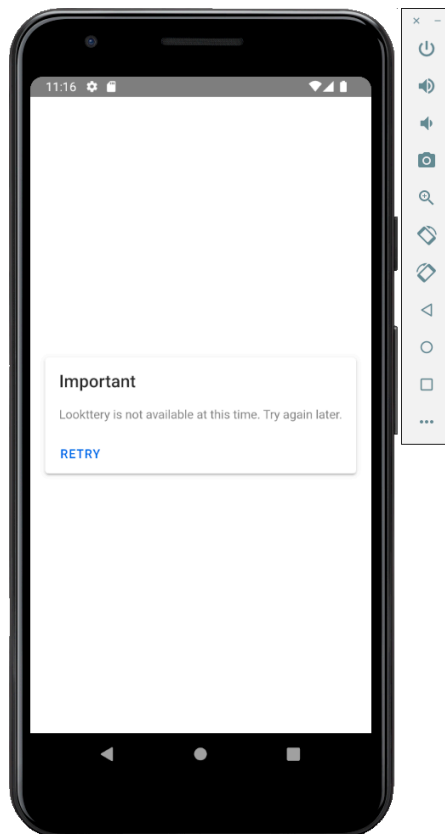


Ilustración 58 - Core shell sin conexión

6.3.2 – Herramientas de debug

Para realizar correctamente cualquier desarrollo es necesario disponer de herramientas de debug adecuadas. Este proyecto posiblemente sea el más complicado de depurar teniendo en cuenta que se trabaja tanto con la lógica de negocio del micro-frontal como con la lógica propia del Shell y, a su vez, con los proyectos nativos generados.

Para depurar código del Shell o del micro-frontal es necesario usar las herramientas de debug de Chrome o las de Safari, dependiendo de si estamos trabajando en un dispositivo Android o iOS.

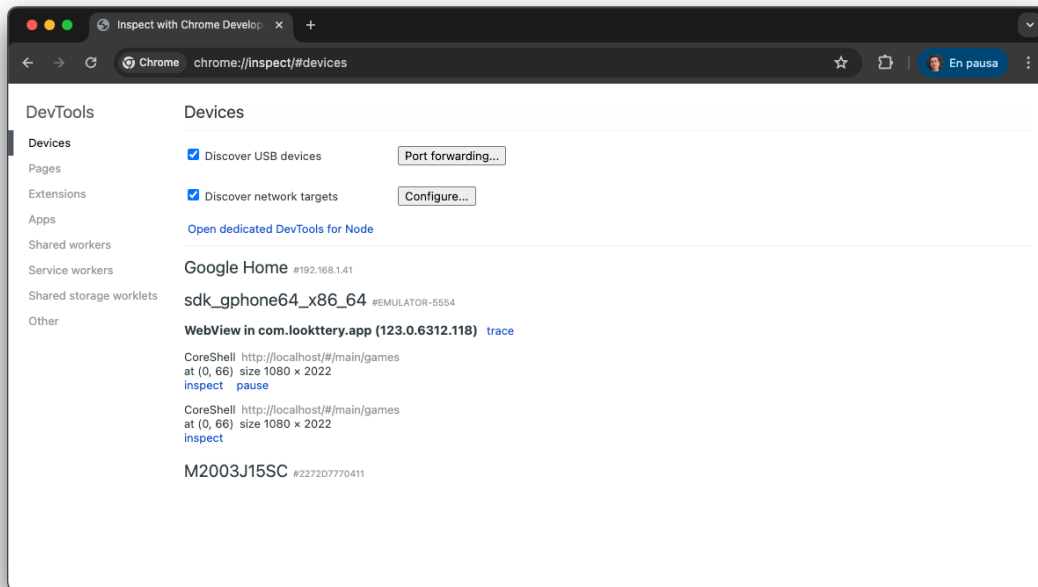


Ilustración 59 - Core shell Chrome inspect desde emulador Android

Como se aprecia en la imagen anterior, podemos ver como aparece la instancia Web en la App con bundle id `com.lookttery.app` y así acceder a las herramientas de desarrollo de Chrome.

En la siguiente imagen se puede ver las herramientas de debug de Chrome, que habitualmente se usan para desarrollos web, pero en este caso nos muestran ambos proyectos, el proyecto Shell aparece como `localhost` y el proyecto micro-frontal aparece con la IP desde la que está ejecutándose, en este caso la `192.168.1.40`, en este caso el proyecto Shell esta compilado en su versión para producción por lo que el código esta minificado y no aparecen los recursos de código ya que no se han generado los ficheros `.map` para ello. En cambio, el proyecto micro-frontal si está sirviéndose en modo desarrollo y como tiene generados los ficheros con extensión `.map` podemos inspeccionar el código con mayor eficiencia.

Es importante no confundir la expresión generado para producción de un proyecto con el empaquetado para producción (reléase nativa) de la App, ya que en este caso directamente los inspectores no detectarían el dispositivo y no podríamos ni acceder siquiera a las herramientas de depuración.

Por lo general las herramientas de debug de Chrome son mejores en muchos aspectos y, de ser posible es mejor usarlas antes que las de Safari, no obstante, en caso de necesitarlo el sistema es el mismo, pero desde Safari.

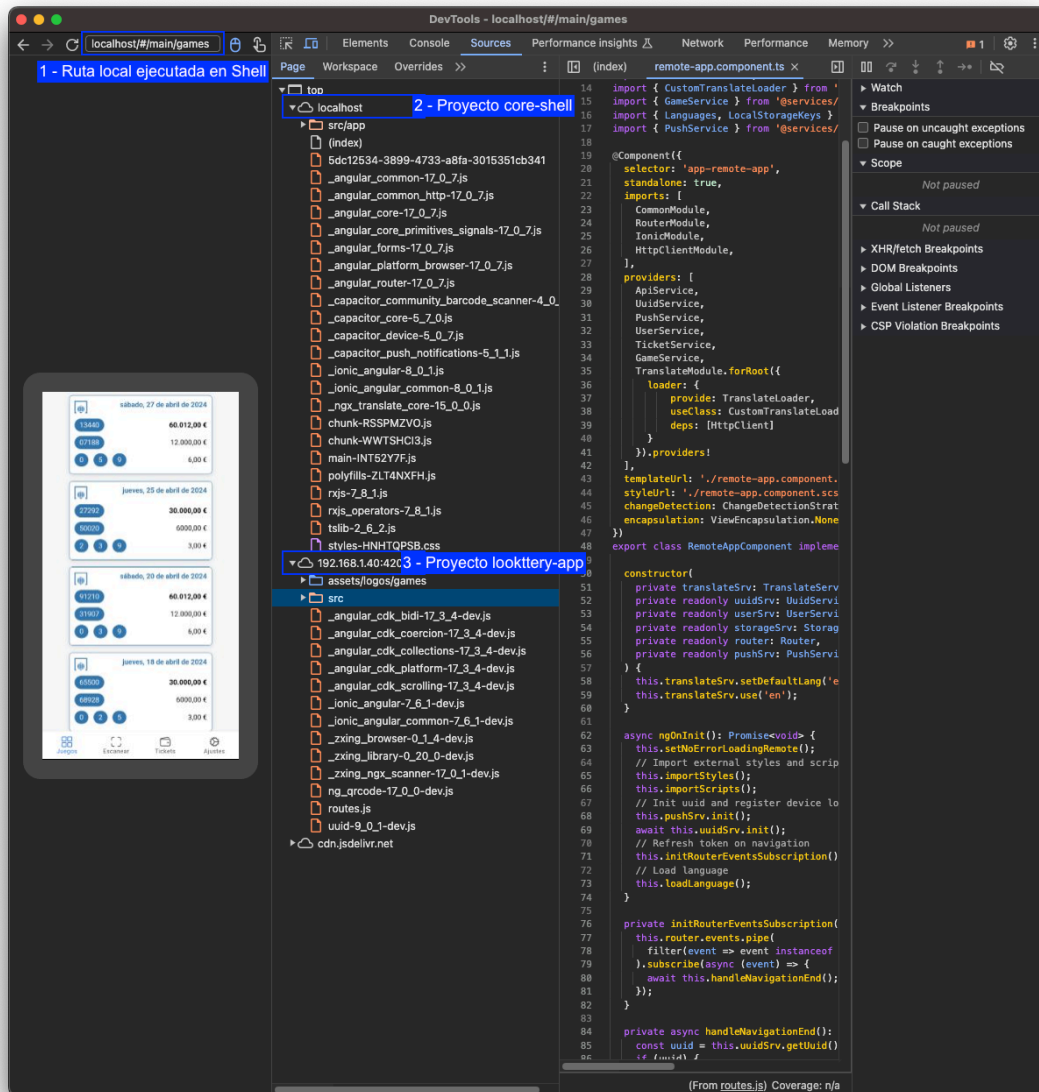


Ilustración 60 - Core shell Chrome Dev Tools

6.3.3 – Herramientas de debug nativas.

Es habitual que surja la necesidad de depurar código nativo en este tipo de proyectos. Por ejemplo, si usamos un plugin para Capacitor desarrollado por nosotros mismos o simplemente para configurar plugins que necesitemos usar o por otros problemas que nos surjan al generar nuestras Apps.

En mi caso he tenido que configurar de forma independiente ambos proyectos nativos para utilizar notificaciones push y, como no conseguí hacerlo funcionar a la primera, necesité depurar ambos proyectos para comprobar si estaban recibiendo correctamente el token en cada caso.

Para ello debemos abrir cada proyecto en su entorno de desarrollo nativo, para Android he utilizado Android Studio y para iOS he utilizado Xcode.

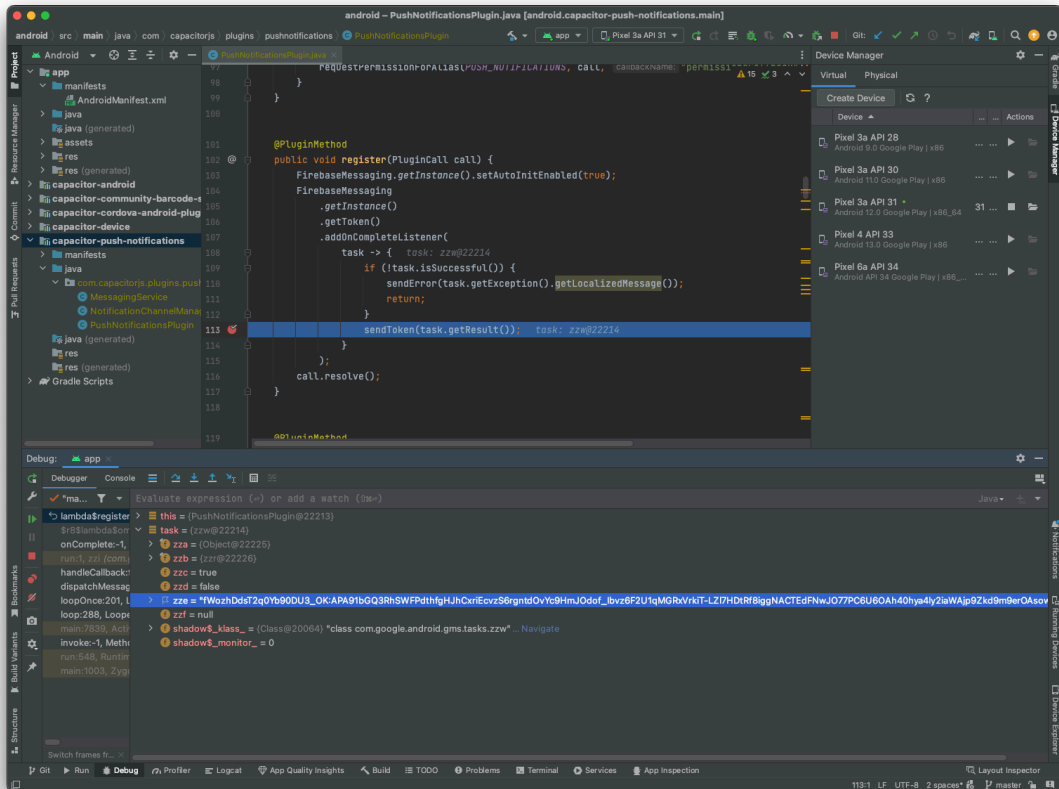


Ilustración 61 - Core shell depurando plugin de notificaciones para ver token

Como se aprecia en la imagen anterior he lanzado la ejecución contra el emulador Pixel 3a API 31 en modo debug, he puesto un punto de ruptura en la línea 113 del código java del plugin de notificaciones y en el inspector puedo ver el token recibido correctamente.

Por lo general todas las herramientas de debug comparten las mismas funcionalidades esenciales y se utilizan de forma muy similar, aunque la interfaz de usuario sea distinta en cada caso.

6.3.4 – Estructura del proyecto

El proyecto Shell es el que más ficheros de configuración importantes tiene, su estructura es similar a la de un proyecto Angular pero además está integrado con Capacitor y con Native Federation.

El fichero federation.manifest.json que se encuentra en los assets del proyecto es donde configuramos la URL del micro-frontal.

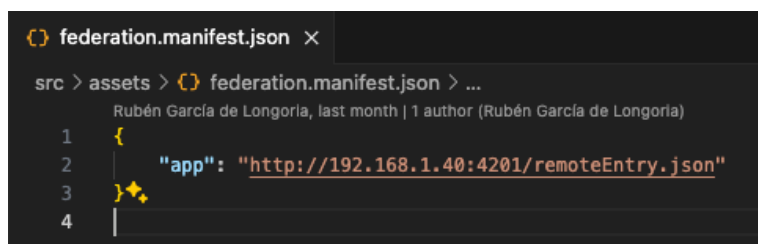


Ilustración 62 - Core shell configuración del micro-frontal

En el fichero capacitor.config.ts es donde se configuran las funcionalidades de Capacitor y alguna información esencial de la App como el identificador o BundleId, el directorio donde se

generarán los ficheros de distribución, los dominios de navegación permitidos y, en mi caso particular, la configuración de la presentación de las notificaciones push.

```
capacitor.config.ts x
capacitor.config.ts > ...
Rubén García de Longoria, last month | 1 author (Rubén García de Longoria)
1 import { CapacitorConfig } from '@capacitor/cli';
2
3 const config: CapacitorConfig = {
4   appId: 'com.lookttery.app', "lookttery": Unknown word.
5   appName: 'Lookttery', "Lookttery": Unknown word.
6   webDir: 'dist/core-shell',
7   server: {
8     // androidScheme: 'http',
9     allowNavigation: ['*'],
10  },
11  plugins: {
12    PushNotifications: {
13      presentationOptions: ["badge", "sound", "alert"],
14    },
15  },
16 };
17
18 export default config;
19
```

Ilustración 63 - Core Shell configuración de Capacitor

En la raíz del proyecto existen dos directorios llamados Android y iOS que, respectivamente, contienen los proyectos nativos para cada sistema operativo.

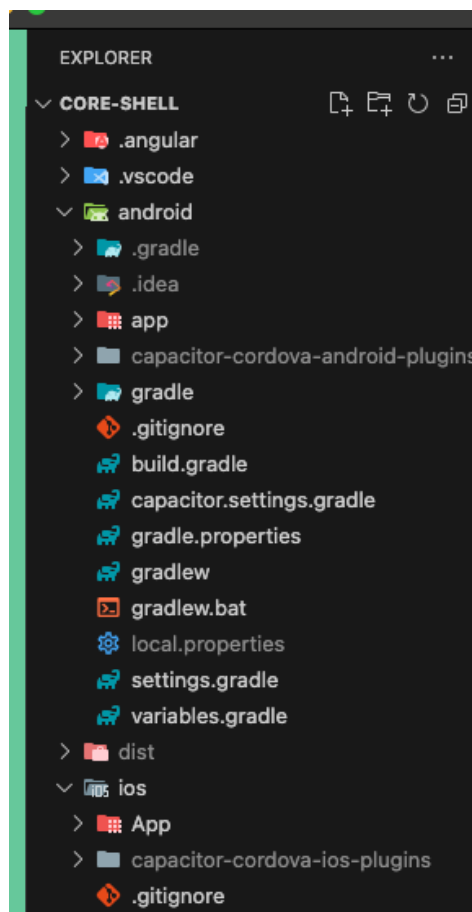
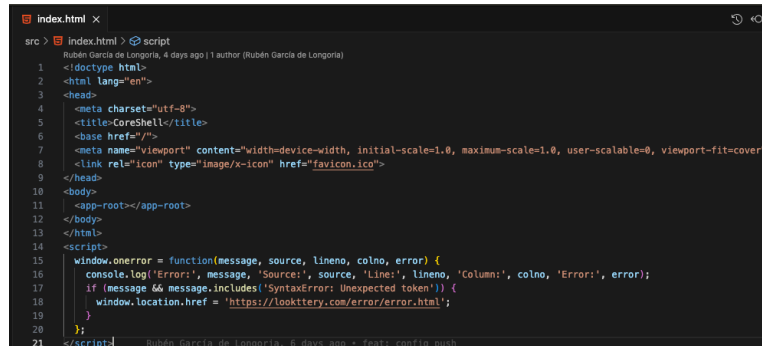


Ilustración 64 - Core shell directorios Android y iOS

En el fichero index.html del proyecto hemos incluido la lógica necesaria para, en caso de recibir un error de versiones de JavaScript, poder redirigir al usuario a la URL donde se le informa de cómo debe actualizar su dispositivo.

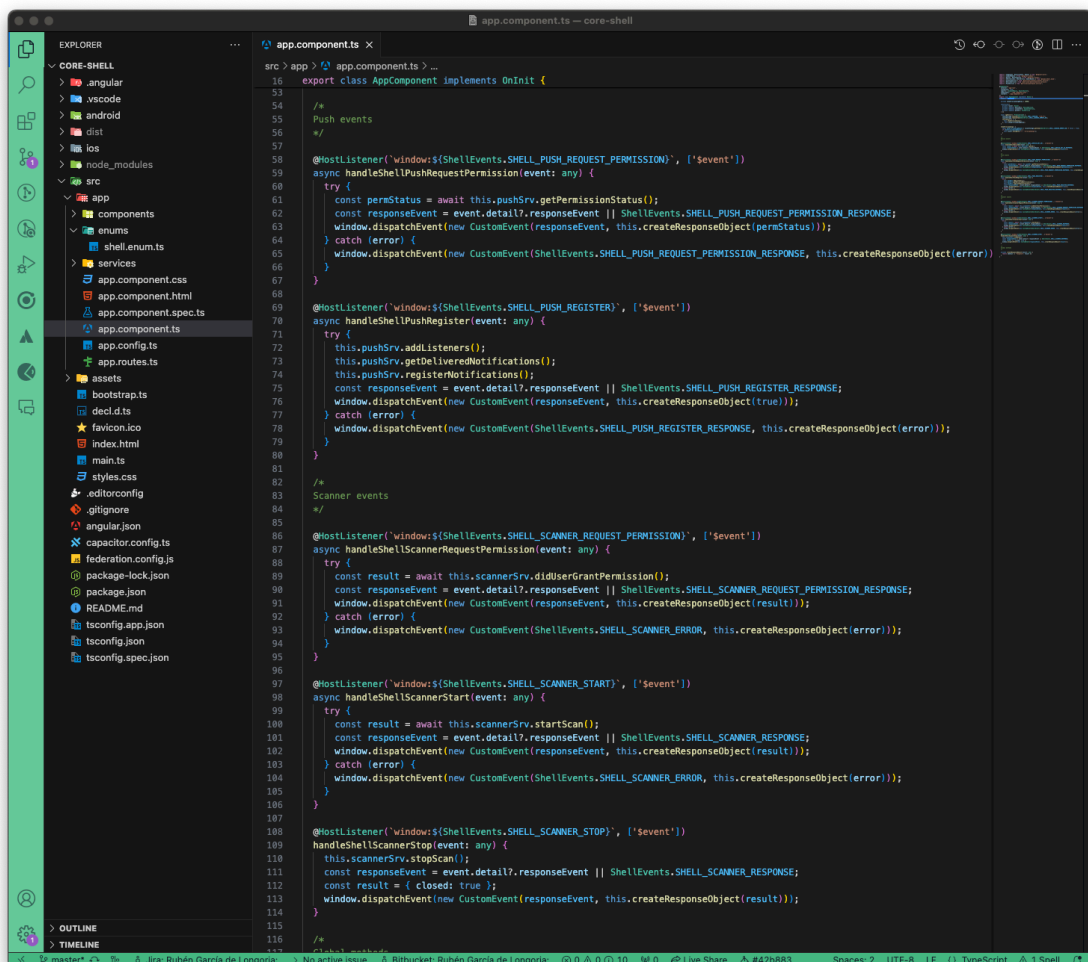
Además, hemos configurado unas cabeceras muy importantes para evitar el zoom y el cambio de tamaño del contenido y así mantener un aspecto sólido dentro de la App.



```
1 <!doctype html>
2 <html lang="en">
3 <head>
4 <meta charset="utf-8">
5 <title>CoreShell</title>
6 <base href="/" />
7 <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=0, viewport-fit=cover">
8 <link rel="icon" type="image/x-icon" href="favicon.ico">
9 </head>
10 <body>
11 <app-root></app-root>
12 </body>
13 </html>
14 <script>
15 window.onerror = function(message, source, línea, colno, error) {
16   console.log('Error:', message, 'Source:', source, 'Line:', línea, 'Column:', colno, 'Error:', error);
17   if (message && message.includes('SyntaxError: Unexpected token')) {
18     window.location.href = 'https://lookttery.com/error/error.html';
19   }
20 };
21 </script>
```

Ilustración 65 - Core shell fichero index.html

A diferencia del proyecto micro-frontal, en el proyecto Shell si tiene relevancia el componente de Angular app.component, y es aquí donde se define la lógica de eventos principal mediante la que se comunicará el proyecto Shell y el micro-frontal.



```
16 export class AppComponent implements OnInit {
17   //
18   // Push events
19   //
20   @HostListener('window:${ShellEvents.SHELL_PUSH_REQUEST_PERMISSION}', ['$event'])
21   async handleShellPushRequestPermission(event: any) {
22     try {
23       const permStatus = await this.pushSrv.getPermissionStatus();
24       const responseEvent = event.detail7.responseEvent || ShellEvents.SHELL_PUSH_REQUEST_PERMISSION_RESPONSE;
25       window.dispatchEvent(new CustomEvent(responseEvent, this.createResponseObject(permStatus)));
26     } catch (error) {
27       window.dispatchEvent(new CustomEvent(ShellEvents.SHELL_PUSH_REQUEST_PERMISSION_RESPONSE, this.createResponseObject(error)));
28     }
29   }
30
31   @HostListener('window:${ShellEvents.SHELL_PUSH_REGISTER}', ['$event'])
32   async handleShellPushRegister(event: any) {
33     try {
34       this.pushSrv.addListener();
35       this.pushSrv.getDeliveredNotifications();
36       this.pushSrv.registerNotifications();
37       const responseEvent = event.detail7.responseEvent || ShellEvents.SHELL_PUSH_REGISTER_RESPONSE;
38       window.dispatchEvent(new CustomEvent(responseEvent, this.createResponseObject(true)));
39     } catch (error) {
40       window.dispatchEvent(new CustomEvent(ShellEvents.SHELL_PUSH_REGISTER_RESPONSE, this.createResponseObject(error)));
41     }
42   }
43
44   //
45   // Scanner events
46   //
47   @HostListener('window:${ShellEvents.SHELL_SCANNER_REQUEST_PERMISSION}', ['$event'])
48   async handleShellScannerRequestPermission(event: any) {
49     try {
50       const result = await this.scannerSrv.didUserGrantPermission();
51       const responseEvent = event.detail7.responseEvent || ShellEvents.SHELL_SCANNER_REQUEST_PERMISSION_RESPONSE;
52       window.dispatchEvent(new CustomEvent(responseEvent, this.createResponseObject(result)));
53     } catch (error) {
54       window.dispatchEvent(new CustomEvent(ShellEvents.SHELL_SCANNER_ERROR, this.createResponseObject(error)));
55     }
56   }
57
58   @HostListener('window:${ShellEvents.SHELL_SCANNER_START}', ['$event'])
59   async handleShellScannerStart(event: any) {
60     try {
61       const result = await this.scannerSrv.startScan();
62       const responseEvent = event.detail7.responseEvent || ShellEvents.SHELL_SCANNER_RESPONSE;
63       window.dispatchEvent(new CustomEvent(responseEvent, this.createResponseObject(result)));
64     } catch (error) {
65       window.dispatchEvent(new CustomEvent(ShellEvents.SHELL_SCANNER_ERROR, this.createResponseObject(error)));
66     }
67   }
68
69   @HostListener('window:${ShellEvents.SHELL_SCANNER_STOP}', ['$event'])
70   handleShellScannerStop(event: any) {
71     this.scannerSrv.stopScan();
72     const responseEvent = event.detail7.responseEvent || ShellEvents.SHELL_SCANNER_RESPONSE;
73     const result = { closed: true };
74     window.dispatchEvent(new CustomEvent(responseEvent, this.createResponseObject(result)));
75   }
76
77   //
78 }
```

Ilustración 66 - Core shell definición de eventos y funcionalidades

6.3.5 – Despliegue en producción

Con la arquitectura de micro-frontales que he diseñado pretendo que este proceso se realice muchas menos veces de las habituales en el desarrollo de una App. Al separar la lógica de negocio en un micro-frontal, todos los cambios relativos al proyecto looktty-app se realizarán como se explica posteriormente en el apartado de despliegue en producción del micro-frontal.

En cambio, siempre que realicemos un cambio en este proyecto necesitaremos realizar los pasos que detallo a continuación. Por lo general serán cambios en la lógica nativa de la App, por ejemplo, cuando se necesite subir de versión Capacitor para que funcionen nuevas versiones de Android y iOS no soportadas o, en mi caso, la próxima versión que tengo pensada sacar será cuando incluya el plugin de detección de cambios en la red, para poder detectar si el usuario se queda sin Internet o si recupera la conexión de forma nativa.

Lo primero que hay que hacer es compilar el proyecto en modo producción mediante el comando ***npm run prepare --configuration=production***, el cual es un comando definido por mí que hace en una sola instrucción el build del proyecto y la sincronización de ficheros con los proyectos nativos de Android y iOS.

```
"prepare": "ng build --configuration ${npm_config_configuration} && npx cap sync",  
"prepare-android": "ng build --configuration ${npm_config_configuration} && npx cap sync android",  
"prepare-ios": "ng build --configuration ${npm_config_configuration} && npx cap sync ios"
```

Ilustración 67 - Core shell scripts prepare

Una vez terminada la ejecución del script debemos abrir mediante Android Studio el proyecto nativo del directorio Android del proyecto.

Una vez abierto es normal que el entorno ejecute sincronizaciones de gradle e indexaciones, hay que esperar a que termine de realizar todas estas acciones antes de continuar.

Desde el menú Build activamos la opción de generar un paquete o APK firmado.

Si vamos a subir la App a la tienda oficial de Google debemos generar un App Bundle, si por el contrario vamos a distribuir de forma manual la App elegiremos APK, recordando que para instalar este APK en un dispositivo necesitaremos activar las opciones para permitir orígenes desconocidos en el propio dispositivo.

A continuación, procedemos a incluir el fichero .keystore y nuestros datos de usuario y contraseña para firmar la App y elegiremos si es un producto de debug (incluirá ficheros necesarios para poder depurar) o de reléase.

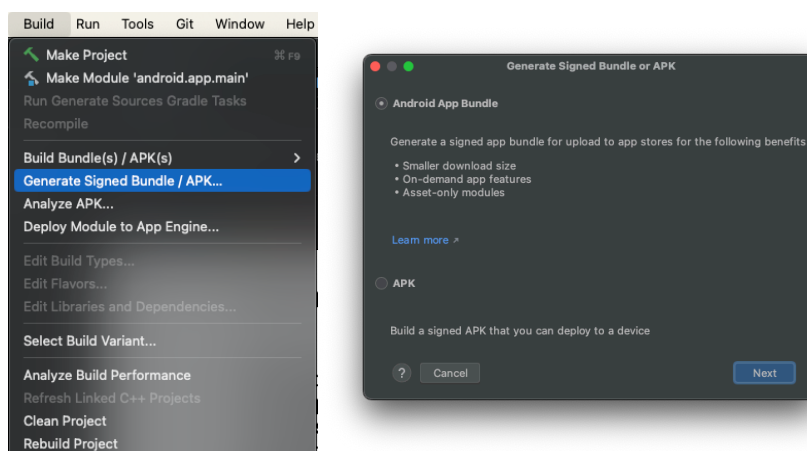


Ilustración 68 - Core shell Android menú generate APK

Finalmente se generará el producto final y un mensaje nos facilitará el acceso al directorio donde encontrarlo.

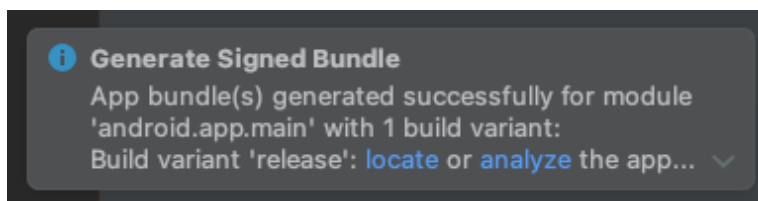


Ilustración 69 - Core shell Generate Signed Bundle

En el caso de iOS debemos de abrir el fichero App.xcworkspace del directorio iOS de la raíz de nuestro proyecto. Es importante no abrir el fichero .xcodproj ya que, aunque abrirá Xcode no tendrá todos los elementos necesarios.

Es importante dejar que Xcode realice todos los procesos de preparación e indexación necesarios antes de continuar.

Desde el menú de producto activaremos la opción de archivar y esperaremos a que se realice el proceso. Finalmente se abrirá la ventana del organizador donde veremos nuestro build realizado. Los siguientes pasos serán decidir si queremos subirlo a TestFlight y AppStore o si preferimos generar un IPA propio. Para instalar un IPA en un dispositivo de forma manual el proceso es algo complejo y previamente debemos de haber incluido el identificador único de dispositivo en el perfil de aprovisionamiento de la App, además, para distribuirla será necesario un software preparado como un MDM o una solución similar a AppCenter de Microsoft o un software Mac de confianza con el que instalar la App.

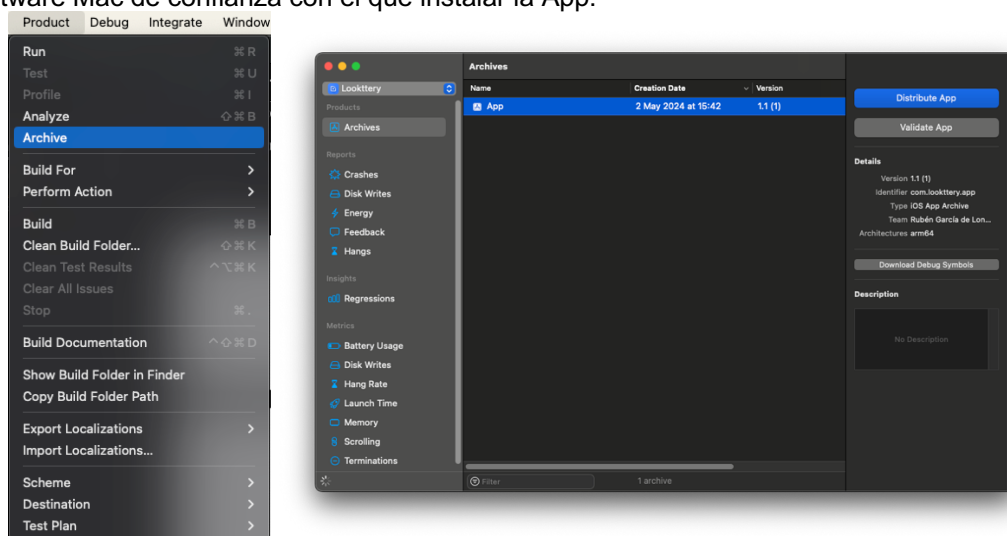


Ilustración 70 - Core shell archive a build

6.4 – Publicación en las tiendas oficiales

El objetivo final de la mayoría de las Apps es la de acabar publicadas en alguna tienda oficial o en algún portal de distribución de Apps interno como un MDM.

En el caso particular de Lookttery el objetivo final es tener la App publicada en la App Store de Apple y la Google Play Store.

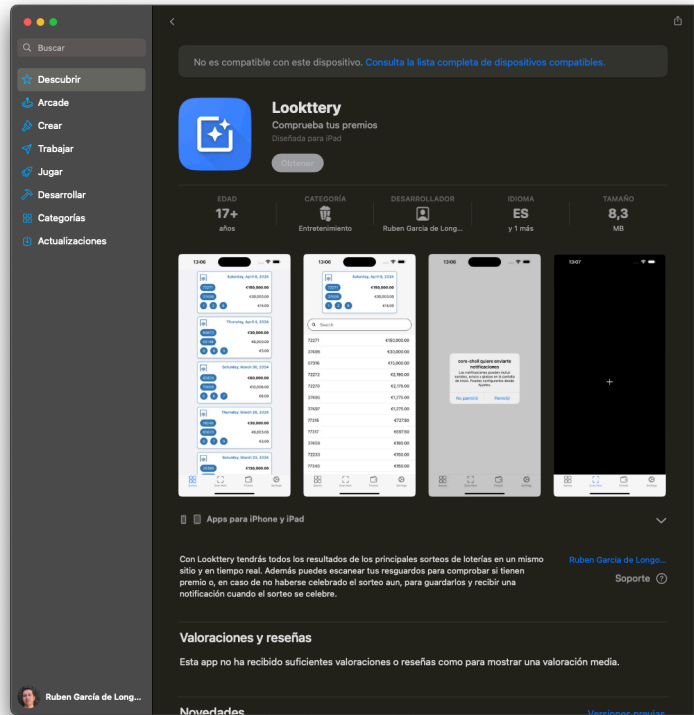


Ilustración 71 - Lookttery en la App Store

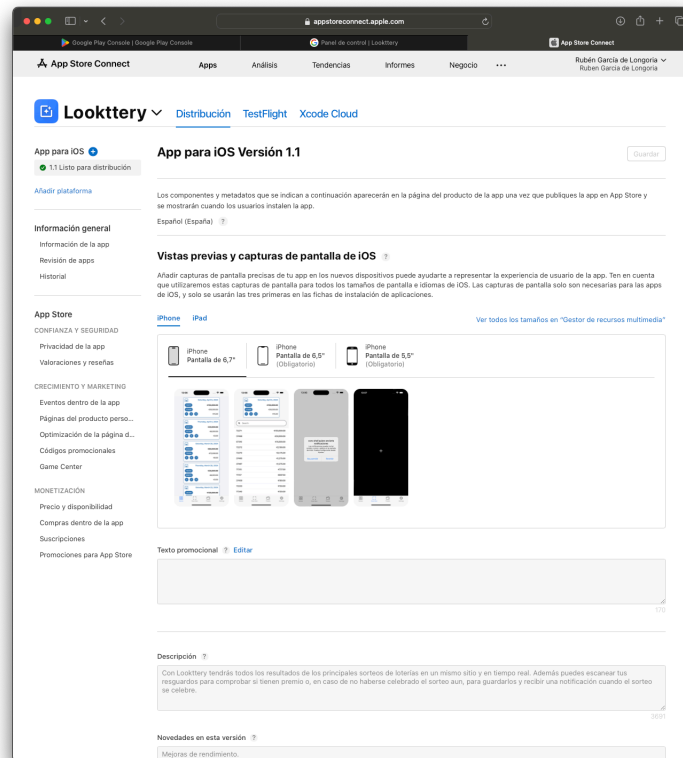


Ilustración 72 - Lookttery portal de desarrollador de Apple

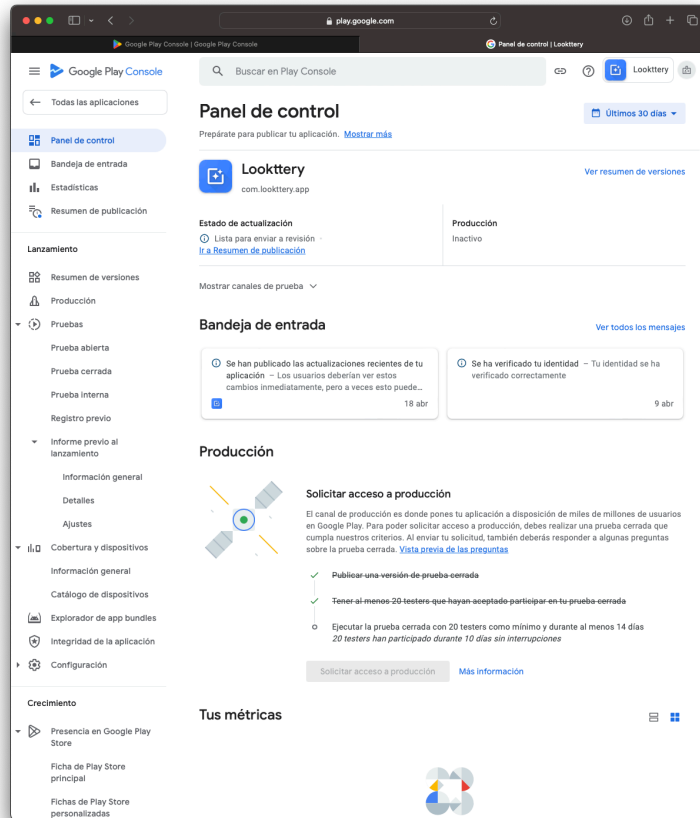


Ilustración 73 - Lookttery en la Play Store

Como se ve en la imagen anterior, después de crear una versión de prueba cerrada con 20 testers, queda mantener la prueba durante 14 días.

Para publicar ambas versiones los cambios son similares, aunque con diferencias en el proceso.

3. Conclusiones

Una vez realizado el MVP del proyecto puedo afirmar que se han conseguido todos los objetivos e incluso han sido superados. En el momento de escribir estas líneas, Lookttery trabaja con el juego de La Primitiva e incluye funciones adicionales, como por ejemplo más idiomas de los esperados.

Lookttery es un proyecto vivo, personalmente tengo intención de mantenerlo y ampliarlo con el fin de que pueda ser autosostenible económicamente y que genere suficientes ingresos para mantener el coste de los servicios que se requieren para su funcionamiento.

En un futuro a medio plazo me gustaría que fuese una App capaz de trabajar con los principales sorteos de la administración de loterías del estado de igual forma que ya lo hace con Lotería Nacional y La Primitiva. Si el número de usuarios crece y el sistema de donaciones no fuese suficiente incluiría publicidad de alguna forma poco invasiva para que no afecte demasiado a la experiencia de usuario.

A largo plazo me gustaría que funcionase con loterías extranjeras, sería una App personalizable con filtros suficientes para configurarse al gusto de cada usuario, a nivel de integración continua necesitaría mejorar los flujos para poder trabajar y publicar cambios sin mucho esfuerzo, en menos tiempo y con mayor seguridad ante bugs, por ejemplo, mejorando el sistema de rollback de versiones de la API y del micro-frontal.

Aunque ya tengo experiencia en el mundo del desarrollo de Apps móviles, tanto nativas como híbridas o multiplataforma, he de reconocer que este tipo de Arquitectura me ha encantado, el potencial de la actualización en tiempo real del proyecto es una mejora muy interesante. En todos los desarrollos que he trabajado podía pasar que un cambio visual o una funcionalidad subiese en una versión con algún fallo y, esto requería realizar el proceso de publicación de nuevo, mientras que con esta arquitectura es tan sencillo como publicar una aplicación web.

Todo el conocimiento adquirido como los problemas que he detallado en la memoria sobre los assets o el uso de rutas en el micro-frontal han sido solventados con soluciones optimas me va a resultar muy útil para futuros proyectos personales y para implantaciones en mi actual empresa, donde están muy interesados en esta arquitectura y donde siguen de cerca mis avances.

Más allá de la lógica de negocio de Lookttery, he tenido bastantes dificultades a la hora de encontrar ayuda y documentación de Native Federation y el uso de micro-frontales para proyectos móviles, ya que toda la documentación existente está orientada a soluciones web.

Aunque estoy un poco cansado en la etapa final del grado, es probable que realice algún artículo sobre este tema en alguna plataforma, como por ejemplo Medium, y realice algún video explicando este modelo y liberando el prototipo básico en repositorios públicos para intentar que se conozca como una alternativa realista a los proyectos habitualmente monolíticos.

4. Glosario

API Rest	Generalmente es como se conoce al software del lado del servidor que ofrece servicios o endpoints que realizan cierta lógica y generalmente trabajan con los datos de la base de datos. En este documento hace referencia al proyecto NodeJS llamado lookttery-api y que ofrece funcionalidades de CRUD y de otra índole sobre la base de datos.
AAB <i>Android App Bundle</i>	Extensión del paquete que contiene la información necesaria para instalar el APK adecuado para cada dispositivo.
APK <i>Android Package</i>	Extensión del paquete instalable de una App Android generado por Android Studio.
BundleId	Identificador único para Apps de Apple, suele tener una estructura similar a un dominio web invertido como por ejemplo com.lookttery.app y debe ser único. En Android es conocido como nombre de paquete.
IPA <i>iOS App Store Package</i>	Extensión del paquete instalable de una App Apple generado por Xcode.
JWT <i>json-web-token</i>	Método de comunicación estándar y abierto para la comunicación de información, en este caso se usa para enviar información de autenticación en las llamadas realizadas a la API Rest.
micro-frontal <i>micro-frontend</i>	Generalmente es como se conoce a un software independiente capaz de realizar una o varias funciones concretas de forma independiente y aislada. En este documento hace referencia al proyecto web que se carga de forma asíncrona con toda la lógica de negocio de la App, concretamente es el proyecto lookttery-app.
notificación push	Es una comunicación que nace del lado del servidor ante una lógica concreta para comunicarse con los dispositivos. En este caso hace referencia al aviso de nuevos juegos disponibles a los dispositivos de usuarios que tienen algún ticket del nuevo juego que acaba de ser almacenado en la base de datos.
secreto <i>secrets</i>	Es un dato secreto, privado crítico para el funcionamiento y seguridad de la App. En este proyecto son secretos la clave de encriptación de los tokens de autenticación JWT o la clave para el envío de notificaciones push desde el servidor.
shell	Es el proyecto base que carga los micro-frontales, en este caso es el proyecto core-shell y carga solamente un micro-frontal.
TestFlight	Es la plataforma de pruebas internas de Apple.

5. Bibliografía

Toda la información ha sido consultada ha sido digital y es la ofrecida en los siguientes enlaces:

- <https://capacitorjs.com>
- <https://github.com/ionic-team/capacitor>
- <https://angular.io>
- <https://github.com/angular/angular>
- <https://www.npmjs.com/package/@angular-architects/native-federation>
- <https://github.com/angular-architects/module-federation-plugin/blob/main/libs/native-federation/README.md>
- <https://ionicframework.com>
- <https://ionicframework.com/docs/components>
- <https://nestjs.com>
- <https://www.mongodb.com>
- <https://firebase.google.com>
- <https://railway.app>

6. AnexoS

Esta memoria se ve ampliada en un anexo, no muy extenso, donde se ha trasladado información sobre la parte de diseño orientado a usuarios y el prototipo.

El motivo es que se considera que esta información ocupa demasiado espacio teniendo en cuenta el límite de páginas establecido para la memoria, además se considera que esta información tiene un carácter más subjetivo, encuestas, análisis de resultados, sketches... y he pretendido que en la memoria quede la información más útil y objetiva posible.

El documento anexo se llama "PEC4_ANEXO_TF_Plantilla_Memoria_es_v5.pdf" y contiene los siguientes puntos:

- 1 – Diseño centrado en el usuario
- 2 – Prototipo