UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Sitio web de encuentro para la comunidad Animal Crossing New Horizons

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Sánchez Melero, Alejandro

Tutor/a: Albert Albiol, Manuela

CURSO ACADÉMICO: 2021/2022

Resumen

Animal Crossing New Horizons es, actualmente, un videojuego para Nintendo Switch muy popular que cuenta con una comunidad muy extensa. Este juego ofrece multitud de opciones multijugador en linea. Sin embargo, sus jugadores carecen de herramientas para poder sacar el máximo potencial del juego, lo que supone que acaben segradados en diferentes puntos de encuentro.

El objetivo es crear una aplicación web donde poder reunir a toda la comunidad, ofreciéndoles todas las funcionalidades y recursos que se estiman oportunos, centrados principalmente en la experiencia de usuario. Para conseguir este propósito, se ha llevado a cabo un estudio con los usuarios finales para conocer sus intereses. A partir de ahí, se ha implementado la aplicación empleando tecnologías de última generación como React.js, TailwindCSS, CI/CD, TypeScript, el *framework* Next.js, Node.js, Prisma, PostgreSQL y Docker. La elección de estas tecnologías se debe a que juegan un papel fundamental en el desarrollo web moderno.

Finalmente, se ha obtenido un producto software donde, la parte del *front-end* cuenta con un diseño funcional y atractivo, así como el *back-end* que es escalable, robusto y más rápido mediante el uso de *Edge Functions*. Gracias a las características descritas, se ha logrado satisfacer la gran mayoría de necesidades de la comunidad.

Palabras clave: Animal Crossing New Horizons, React.js, Experiencia de usuario, Edge Functions, Aplicación Web

Abstract

Animal Crossing New Horizons is a very popular video game for Nintendo Switch with an extremely large community nowadays. This game offers a lot of online multiplayer options. However, their players do not have the tools to get the full potential of the game. Therefore, it means that the players are segregated into different meeting points.

The objective is to create a web application where all players can meet and offer them all functionalities and resources that are considered appropriate, mainly focused on the user experience. To achieve this purpose, a study has been carried out with end users to find out their interests. From there, the application has been implemented with state-of-the-art technologies like React.js, TailwindCSS, CI/CD, TypeScript, Next.js framework, Node.js, Prisma, PostgreSQL and Docker. The choice of these technologies is due to the fact that they play a fundamental role in modern web development.

Finally, a software product has been obtained where the front-end has a functional and attractive design, as long as the back-end which is scalable, robust and faster by using Edge Functions. Thanks to the characteristics described, it has been possible to satisfy the vast majority of the needs of the community.

Key words: Animal Crossing New Horizons, React.js, User experience, Edge Functions, Web Application

Índice general

Índice de figuras

CAPÍTULO 1
Introducción

Los videojuegos cada día están más presentes en nuestras vidas, habiendo superado ya a la industria cinematográfica. Por lo tanto, cada vez tienen una mayor relevancia a nivel social.

Es gracias a la llegada de Internet y a los avances tecnológicos que los videojuegos cada vez son más accesibles para todos los públicos, con títulos de lo más variados. A medida que avanzan los años tienen un mayor componente *online*, donde es importante interactuar con el resto de los jugadores para poder aprovechar todo el potencial que nos ofrece el juego, esto es lo que sucede en Animal Crossing New Horizons.

En ocasiones, no es posible disfrutar de todo lo que nos ofrecen, ya que sus desarrolladores no han creado una herramienta adecuada para que los jugadores puedan comunicarse de una manera eficiente fuera del propio juego. Esto da lugar a que se desaproveche un gran potencial de los videojuegos, debido a que los jugadores están limitados en la forma de comunicarse fuera del ámbito del juego. Dependen de auto-organizarse en comunidades segregadas, un jugador tiene que estar presente en varias comunidades para poder realizar todo lo que desea.

1.1 Motivación

Como entusiasta del mundo de los videojuegos, además de jugador de Animal Crossing New Horizons [1] (ACNH), soy consciente del problema que presenta no tener un punto de encuentro común y accesible.

Actualmente, la forma más popular entre los jugadores para interactuar entre sí se realiza mediante un proceso manual. Hay un grupo de personas que se encarga de organizar todas las publicaciones de la comunidad.

Teniendo esto en cuenta, veo conveniente el desarrollo de una herramienta que pueda agilizar el cómo interactúan los jugadores, de una forma sencilla y con unas tecnologías modernas.

1.2 Objetivos

El propósito principal del proyecto consiste en la creación de una aplicación web, dando lugar a un punto de encuentro donde los usuarios pueden cooperar entre sí en el videojuego ACNH y poder realizar todas las funciones que necesitan para disfrutar de todo el contenido que ofrece el juego.

Para lograr este objetivo se detallan los siguientes subobjetivos:

- La aplicación debe ser accesible y estar centrada en ofrecer la mejor experiencia de usuario, en adelante UX *(User Experience)*.
- Incorporar la presencia de un *chat* global para facilitar la comunicación entre todos los miembros de la comunidad.
- Respetar la privacidad, no incorporando publicidad ni rastreadores que hagan un seguimiento de la actividad del usuario dentro de la web. Logrando así que el usuario no sienta que está otorgando más datos de los necesarios.
- La web debe estar optimizada y ofrecer el mejor tiempo de respuesta posible, independientemente de la carga a la que se vea expuesta.
- En el portal web se podrán llevar a cabo las tareas que se realizan en otros puntos de encuentro, además de otras nuevas. De esta forma, se recogerán todas las funcionalidades necesarias para que los jugadores puedan disfrutar por completo de la experiencia *online* de ACNH.
- El empleo de la web debe suponer un ahorro temporal respecto a otras alternativas existentes, además de resultar más sencillo su uso por parte de los miembros.

1.3 Estructura de la Memoria

El presente documento va a contar con la estructura que explicaré a continuación:

En el capítulo dos haremos un repaso sobre el estado del arte, en él se van a revisar los principales puntos de encuentro donde se reúnen los jugadores de ANCH, analizando sus puntos fuertes y débiles, y reflexionando sobre ellos.

En el capítulo tres, veremos las dos metodologías que se emplean en el proyecto y se argumentará por qué han sido seleccionadas.

En el capítulo cuatro se explica cómo se realizan las diferentes fases por las que ha transcurrido el diseño de la interfaz de usuario. Otro aspecto importante que se va detallar aquí es la arquitectura, que va a servir de base para el proyecto, también el modelo de datos que soporta la aplicación. Por último, veremos todas las tecnologías que van a ayudar a construir la solución final.

En el capítulo cinco continuaremos con el desarrollo de la aplicación, revisaremos los aspectos con más relevancia para nuestra web. Veremos, su mapa de navegación, cómo se ha optimizado la aplicación web y la estructura de la *API*. Otro aspecto importante son los componentes que forman la aplicación, donde se revisará su implementación y el por qué se ha decidido implementarlos. Concluiremos con un análisis sobre la funcionalidad de lo que se ha implementado.

En el capítulo seis se mostrará el resultado obtenido, realizando un recorrido completo por la aplicación.

Un factor importante para este proyecto son las pruebas, en el capítulo siete las repasaremos, comenzando a nivel de *software*. Por otro lado, se llevarán a cabo unas pruebas a un grupo de usuarios para evaluar si el producto final cumple con las expectativas y objetivos planteados.

El capítulo ocho cuenta las conclusiones que se han extraído de la realización de este proyecto, valoraremos si hemos cumplido los objetivos, haremos una reflexión sobre el proyecto y atenderemos a la relación de este con los estudios cursados.

Para concluir, en el capítulo nueve se plantearán posibles adicciones y mejoras que poder llevar a cabo para perfeccionar el proyecto.

CAPÍTULO 2
Estado del arte

Actualmente, hay diferentes alternativas donde los jugadores de ACNH pueden interactuar entre sí. Estas opciones son páginas web en su mayoría, pero la más empleada es una aplicación móvil de mensajería. A continuación, se detallan las diferentes alternativas.

2.1 Telegram

Telegram [2] es una aplicación de mensajería lanzada en 2013 presente en Android e IOS, además cuenta con una versión web. Así mismo, ofrece funcionalidades muy superiores a otras aplicaciones de mensajería, entre estas funciones destacan los grupos con una gran cantidad de personas, los canales de difusión y una *API* para *bots*.

Todas las funcionalidades anteriormente mencionadas, junto a ser una aplicación móvil ampliamente conocida, hacen que muchos jugadores se decanten por esta opción para convertirla en su punto de encuentro. Actualmente, el mayor grupo de Telegram de la comunidad cuenta con 6678 miembros.

En esta aplicación los usuarios se dividen en varios grupos para poder comunicarse de forma eficaz. Hay un canal para la venta de nabos, un grupo de conversación general, otro para visitar islas de otros jugadores, un grupo para compartir diseños, etcétera. El problema es que muchos jugadores acaban creando su propia comunidad con sus reglas, lo que provoca segregaciones y enfrentamientos.

La difusión de los mensajes y la participación de los jugadores es sencilla pero, al estar tan segregada, provoca que el jugador deba pertenecer a varios grupos y que la búsqueda de lo que él realmente quiere sea compleja.

2.2 Comunidad AC

Esta web presenta un foro donde los usuarios pueden colaborar entre sí, con un diseño atractivo y limpio, se puede visualizar en la figura 2.1. Está desarrollado con XenForo, un *software* comercial cuyo propósito es facilitar la creación de foros, aportando una gran cantidad de soluciones que únicamente se deben modificar ligeramente para crear un foro.

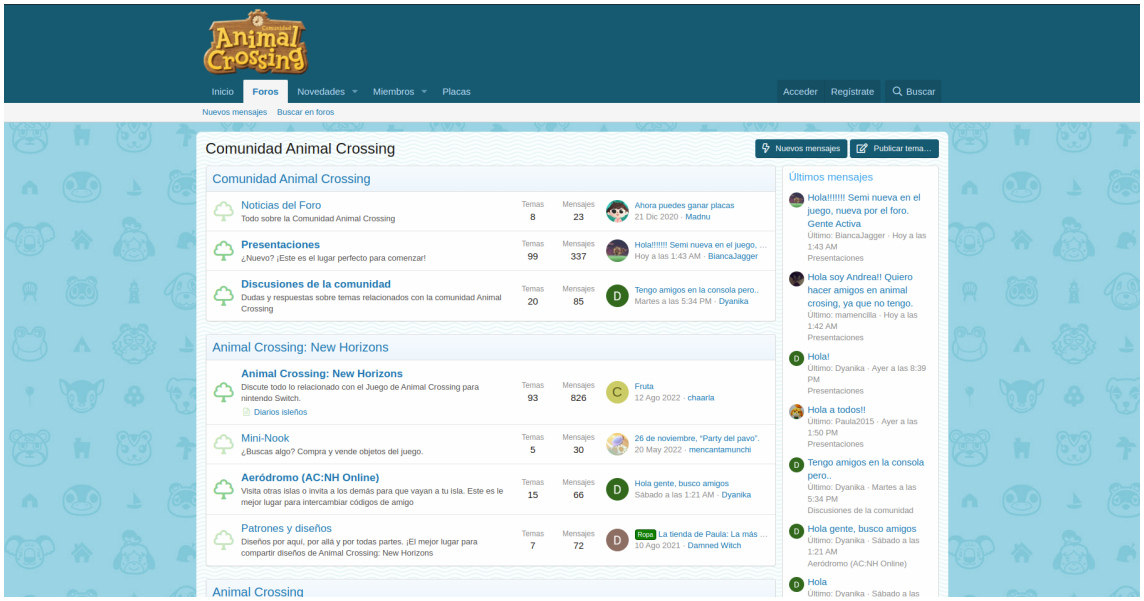


Figura 2.1: Página principal de Comunidad AC

Su principal ventaja es la sencillez de uso, ya que únicamente se puede interactuar mediante el uso de mensajes publicados en temas.

Ofrece a los jugadores una buena forma de comunicarse, pero no proporciona una forma de participar, crear eventos para la comunidad, poder comprar o vender nabos, siendo esta última una de las funcionalidades más demandadas.

Otro punto importante es la necesidad que tienen los usuarios en esta página de introducir datos personales, como su correo electrónico o el empleo de *cookies* por parte de páginas de terceros.

2.3 Mi Animal Crossing

Mi Animal Crossing es otro foro, pero en este caso, engloba a toda la saga de Animal Crossing. La funcionalidad que ofrece es muy similar a la que se ha mencionado en la sección anterior, como podemos contemplar en la figura 2.2.

Esta plataforma es antigua, por lo tanto, no está adaptada a las necesidades reales que buscan los jugadores. Esto provoca que no tenga un diseño móvil, lo cual causa que el uso de un *smartphone* sea algo incomodo y poco intuitivo, otorgando una mala UX.

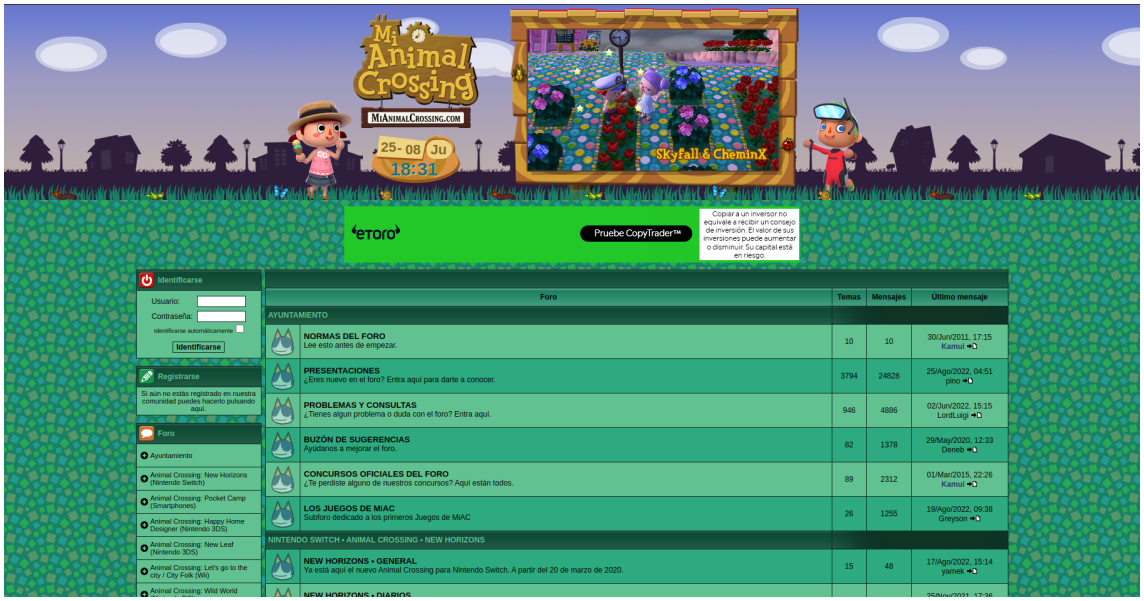


Figura 2.2: Página principal de Mi Animal Crossing

2.4 Nook Friends

La herramienta Nook Friends proporciona un instrumento más completo que los que se han mencionado con anterioridad, en la figura 2.3 se puede observar su página principal. Sin embargo, incluye publicidad intrusiva que impide visualizar el contenido con facilidad. También tiene un diseño poco intuitivo que no resulta atractivo.

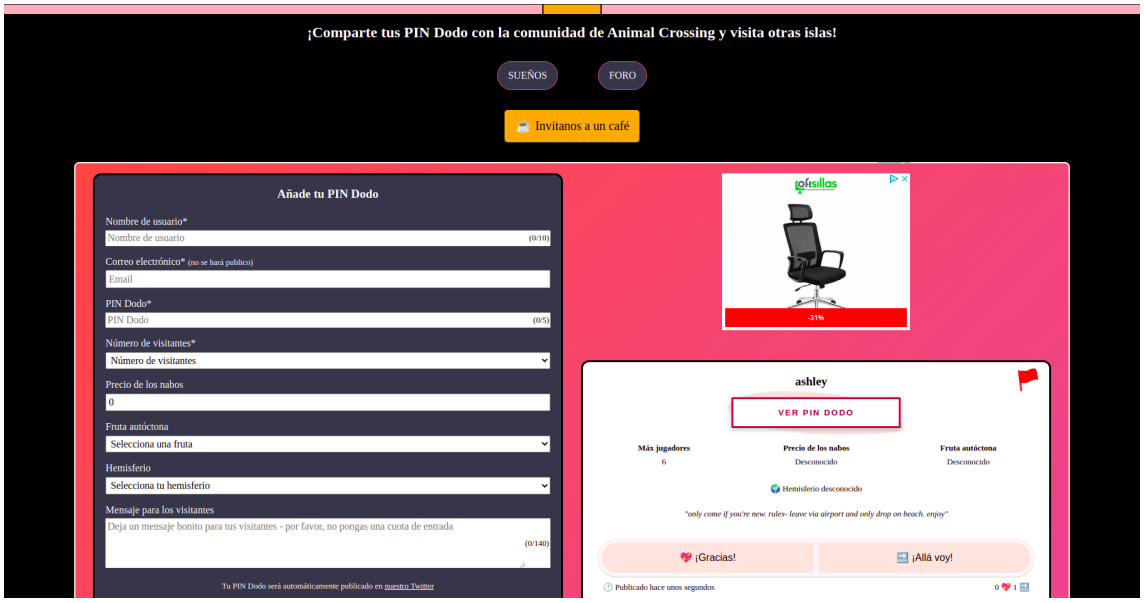


Figura 2.3: Página principal de Nook Friends

En la figura 2.4 se puede apreciar que el diseño no es consistente, además de que hay partes de la interfaz que carecen de estilo. Por lo tanto, pese a ofrecer un mayor número de funcionalidades que el resto, su uso es complejo, dando lugar a que la comunidad no lo use de manera frecuente.

Figura 2.4: Página Peces de Nook Friends

2.5 Conclusión

Al analizar estas herramientas se puede llegar a la conclusión de que cada una hace una serie de cosas bien, sin embargo, ninguna reúne todas las características que busca la comunidad. Algunas de ellas ofrecen un diseño más atractivo y mejor UX, como Comunidad AC, pero únicamente permite la comunicación entre jugadores. En el otro extremo, está Nook Friends, que ofrece más funcionalidad a costa de un diseño muy básico y limitado. En un punto intermedio, nos encontramos con Telegram que ofrece diversas herramientas donde los usuarios pueden organizarse y obtener unas funcionalidades algo limitadas por el entorno donde se encuentran.

Debido a esto, viendo las debilidades y fortalezas, se va crear una solución que incorpore las ventajas de cada uno de ellos: un foro bien estructurado, venta de nabos, creación de eventos, búsqueda de vecinos, etcétera. En nuestra aplicación web se podrá acceder desde cualquier dispositivo, obteniendo una buena UX y toda la funcionalidad que desean los jugadores de forma más automatizada, relevando en el sistema todo el proceso de gestión que se llevaba a cabo para algunas tareas.

CAPÍTULO 3
Metodología

Para poder desarrollar una solución al problema presentado, se ha seguido un enfoque de Desarrollo Centrado en el Usuario [3], de ahora en adelante DCU, además de *Test Drive Development* [4] (desarrollo guiado por pruebas), en adelante TDD, junto a integración continua y entrega continua [5], a partir de ahora CI/CD. En las siguientes secciones se describe cómo se ha aplicado DCU, TDD junto con CI/CD.

3.1 Desarrollo Centrado en el Usuario

Hoy en día, es muy importante involucrar al usuario en el proceso de creación de un software, es aquí donde DCU nos indica unas pautas para que el usuario sea el centro del foco durante el diseño y desarrollo. Este término surgió en 1970 y gracias a Don Norman comenzó a tomar notoriedad. Cuando empleamos este tipo de enfoque, tenemos que tener muy en cuenta las necesidades que busca el usuario. Durante el proceso se debe tener en cuenta la UX completa. DCU se divide en las siguientes fases:

- **Fase de análisis:** durante esta fase nos ayudaremos de diversos métodos para lograr obtener la mayor cantidad de información posible del usuario, en este caso, de los aficionados a ACNH. Este juego está orientado a un gran público, ya que la edad mínima recomendada para jugar comienza en tres años, por lo tanto, es importante conocer a quién va dirigida nuestra solución. Para lograr esto, hemos empleado las siguientes técnicas:

    1. Una **Investigación cualitativa**. En ella se ha llevado a cabo un cuestionario destinado a los jugadores, donde se ha tratado de averiguar el rango de edad, las funcionalidades que esperaban encontrar y cómo suelen interactuar con el resto de jugadores.
    2. El uso de la **Técnica Personas**. Gracias a los resultados de la encuesta, con esta técnica hemos podido construir un perfil del usuario que va a hacer uso de la aplicación. Para ello, se han analizado los resultados del cuestionario y se ha creado un perfil, definiendo sus principales objetivos, inquietudes y necesidades.
    3. Por último, al tener una persona ya definida, se ha procedido a crear diferentes escenarios a través de la **Técnica Escenarios**. Con este método podemos comprobar, definiendo un escenario ficticio, dónde está presente nuestra solución; además de comprobar si el usuario puede cumplir sus necesidades. Como consecuencia de la utilización de esta técnica se pueden averiguar situaciones imprevistas o casos de uso inesperados.

- **Fase de Diseño:** con toda la información recopilada durante la fase anterior se trata de definir un diseño de la interfaz. Para ello, nos podemos ayudar de diversas herramientas, como *mockups*, prototipos, bocetos, etcétera. En nuestro caso, nos hemos decantado por el uso de la herramienta **Uizard**, que ofrece un gran conjunto de soluciones para el diseño. Con ella, se han desarrollado *wireframes* [6] para poder dar una estructura y esquema a la aplicación.

- **Fase de Implementación:** una vez está finalizado el diseño, se puede comenzar a implementar nuestra solución haciendo uso de diferentes tecnologías. Cabe destacar que esta fase también puede afectar de manera considerable a la UX, ya que podríamos desarrollar una solución ineficiente desde el punto de vista del rendimiento o que no sea accesible, impidiendo que la experiencia sea satisfactoria. En este caso principalmente destacan las siguientes tecnologías: React, Node.js, Next.js, Tailwind CSS, entre otras.

- **Fase de evaluación:** en esta fase repasaremos las pruebas de software que han sido creadas para la aplicación, además de aplicar un cuestionario a jugadores para valorar su satisfacción con el resultado obtenido.

3.2 Test Drive Development

Uno de los grandes problemas del desarrollo de software es la falta de pruebas. Esto puede conllevar a fallos críticos, donde hay que gastar recursos extra en solventar un problema que se podría haber evitado.

TDD nos ofrece una solución que permite minimizar riesgos, con esta metodología de desarrollo lo primero que se implementa son las pruebas, antes de comenzar a desarrollar la funcionalidad. Gracias a esto, nos aseguramos de que la funcionalidad que desarrollamos sea correcta, siempre y cuando las pruebas también lo sean. Pero, esto no es la única ventaja que nos ofrece esta metodología, también nos permite poder hacer modificaciones en el código en un futuro de una forma más sencilla y segura, así como comprobar que estas no han provocado errores nuevos o funcionalidad no deseada, logrando un código más limpio y escalable.

El sistema de control de versiones seleccionado es git y el proyecto está en un repositorio de GitHub. Debido a esto, para implementar CI/CD se han usado las GitHub Actions por su facilidad de uso.

CI/CD surge de la necesidad de automatizar tareas que se realizan frecuentemente, tales como compilación, pruebas y despliegue. En el *Martketplace* de Github existen acciones creadas por usuarios que se pueden reutilizar y permiten realizar entre otras, las tareas mencionadas anteriormente.

Una acción dentro del contexto de las GitHub Actions es una aplicación creada para la CI/CD que ofrece GitHub. Sirven para poder ejecutar una tarea compleja que se va a repetir a lo largo del tiempo. El esquema de funcionamiento, en general, comienza con un evento que genera un flujo de trabajo, este evento puede ser un *commit* que se realiza sobre una rama en concreto. El flujo generado se ejecuta en una máquina virtual que provee GitHub, estos flujos están formados por trabajos que, a su vez, están formados por acciones o *scripts*. Organizando estas acciones y estableciendo dependencias entre los trabajos, estos últimos se pueden ejecutar de forma concurrente o no dentro del flujo de trabajo, en la figura 3.1 se puede ver un esquema de su funcionamiento. Un ejemplo sería definir dos trabajos distintos, uno para las pruebas unitarias y otro para las pruebas de integración. Estos trabajos, al no tener una dependencia entre sí, se podrían ejecutar en

paralelo, en la figura 3.2 se puede contemplar un fragmento del fichero de configuración empleado, las dependencias y cómo se define un trabajo.

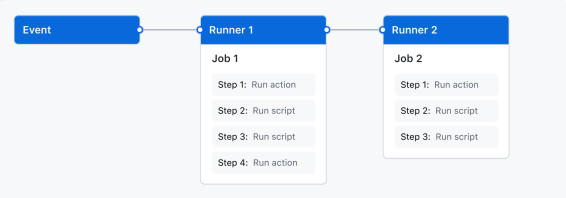


Figura 3.1: Funcionamiento de GitHub Actions

```yaml
build:
  needs: [avoid_reduncy]
  runs-on: ubuntu-latest
  steps:
    - uses: actions/checkout@v3
    - uses: actions/setup-node@v3
      with:
        node-version: '16'
        cache: 'npm'
    - name: Install dependencies
      run: npm ci
    - name: Build
      run: npm run build
    - uses: actions/upload-artifact@v3
      with:
        name: dist
        path: .next

test-unit:
  needs: [lint, build]
  runs-on: ubuntu-latest
  steps:
    - uses: actions/checkout@v3
    - uses: actions/setup-node@v3
      with:
        node-version: '16'
        cache: 'npm'
    - name: Install dependencies
      run: npm ci
    - run: npm run
    - uses: actions/download-artifact@v3
      with:
        name: dist
        path: .next
    - name: Unit tests
      run: npm run test
```

Figura 3.2: Ejemplo de dos Jobs empleados.

Gracias a emplear TDD tenemos pruebas para nuestra aplicación, además contamos con un analizador de código estático, como es Eslint, que permite mantener un estilo uniforme en el código de nuestro proyecto y puede encontrar problemas. Para este proyecto, CI/CD se ha integrado de la siguiente forma: se han automatizado los *commits* a la rama *main*, donde en cada *commit* se ejecutan todas las pruebas y se valida el estilo. En caso de que el flujo de trabajo tenga un resultado positivo, los cambios son incorporados en el repositorio remoto. Esta es la primera parte del proceso, la integración continua. Ahora falta poder llevar estos cambios al entorno de producción, que en nuestro caso, es gracias a Vercel [7], un proveedor de contenidos en la nube que ofrece muchas ventajas como una CDN y edge-functios. De este modo, tenemos nuestro proyecto sin ninguna configuración desplegado de una forma muy sencilla y que escala automáticamente.

CAPÍTULO 4
Diseño de la solución

Ahora que están todos los conceptos explicados para poder llevar a cabo nuestra solución, es hora de comenzar con su diseño.

Durante esta fase, se ha llevado a cabo el diseño de la interfaz gráfica de usuario (GUI), el modelo de datos, la arquitectura y se han seleccionado las tecnologías a emplear.

4.1 Diseño Interfaz de usuario

Para realizar el diseño de la interfaz se empleará la metodología DCU mencionada en el capítulo anterior, en nuestra solución esta fase toma especial relevancia. Es muy importante que este diseño sea correcto para proporcionarle al usuario la mejor experiencia posible. La UX no depende únicamente de la UI, pero en esta fase nos centraremos en la interfaz.

Para recabar información sobre los usuarios de la aplicación, se ha realizado una investigación cualitativa y posteriormente, con la información recopilada, se han aplicado las técnicas Personas y Escenarios.

4.1.1. Investigación Cualitativa

Para aplicar DCU necesitamos conocer a los usuarios que van a hacer uso de nuestra aplicación. Para ello, es necesario algún método para obtener información sobre sus necesidades. Se pueden emplear diferentes técnicas ya sean cuestionarios, entrevistas, estudio de documentación, entre otras. Hay que tener en cuenta que, para analizar las necesidades de los jugadores, lo más conveniente en nuestro caso es un cuestionario, ya que coordinar otro tipo de investigación resulta especialmente complejo. Los resultados detallados del cuestionario se encuentran en el Anexo B.

En este cuestionario se pregunta a los jugadores por sus datos demográficos, habilidades tecnológicas y cuestiones referentes al videojuego ACNH. Ha sido respondido por un total de trece jugadores, de donde se han obtenido los siguientes resultados:

- Tres cuartas partes de los encuestados son mujeres con una edad entre diecinueve y veinticuatro años.
- La gran mayoría usan su *smartphone* y portátil para acceder a internet, además suelen consumir contenido multimedia y redes sociales más de 3 horas al día. Otro dato relevante es que todos ellos emplean redes sociales, predominantemente Whatsapp y Instagram.

- Consideran que tienen una habilidad tecnológica alta.
- El setenta y cinco por ciento pertenece a una comunidad de Animal Crossing u otros juegos, valoran sobre todo poder ayudar, ser ayudados por los demás y que haya un buen ambiente.
- Los que pertenecen a una comunidad, emplean en su gran mayoría Telegram para comunicarse.

4.1.2. Técnica Persona

Esta técnica nos permite conocer mejor a la agrupación de usuarios con intereses similares, se construye una persona como si fuese una persona real. En ella se detallan sus objetivos, inquietudes, necesidades, etcétera. Para ello, con los datos recopilados en el cuestionario mencionado en la sección anterior, se logra tener un mayor conocimiento de los usuarios que van a hacer uso de nuestra solución.

Como se ha detallado en el párrafo anterior, en esta técnica los usuarios se agrupan por intereses comunes, en este caso, la mayoría de usuarios coinciden en sus intereses, por lo que se ha creado una única persona.

Esta persona que acabamos de crear juega habitualmente a ACNH, busca una web sencilla que ofrezca toda la funcionalidad que desea, que no tenga publicidad y presente una interfaz amigable, su definición completa está en la figura 4.1.

Laura Martínez Pérez

- **Genero:** Mujer.
- **Edad:** 22 años.
- **Profesión:** Estudiante en prácticas de Psicología.

OBJETIVOS

- Comerciar con sus nabos de forma sencilla y cómoda, sin tener que encargarse del proceso de gestión.
- Recibir y ofrecer ayuda a otros miembros de la comunidad.
- Participar en eventos creados por el resto de jugadores, además de poder crear sus propios eventos.
- Conocer y poder visitar islas del resto de jugadores.

HABILIDADES TECNOLÓGICAS

- Pasa en Internet más de 5 horas al día.
- Juega a videojuegos de manera habitual.
- Emplea mucho sus redes sociales, sobre todo Whatsapp e Instagram.

Figura 4.1: Persona

Gracias a haber definido a Laura, conocemos mejor el grupo de usuarios que va a emplear nuestra aplicación y se puede proceder con la siguiente técnica.

4.1.3. Técnica Escenarios

Los escenarios son otra técnica empleada para mejorar la UX, nos permiten conocer mejor situaciones reales de uso de nuestra solución, ya que estos muestran una secuencia de pasos que llevan a un objetivo, los objetivos se corresponden con los que busca el usuario. Un escenario cuenta con un contexto alrededor suyo que nos permite hacernos una idea de la situación en la que se encuentra el usuario.

Al haber definido a nuestra persona anteriormente, nos servirá para desarrollar los diferentes escenarios que pueden darse en Toom Forum, este va a ser el nombre que recibe nuestra aplicación web.

A continuación, se definen los diferentes escenarios planteados para Laura:

1. **Venta de nabos.**

   Es viernes por la tarde, Laura acaba de salir de trabajar, recuerda que el domingo compró nabos. Mientras está en el bus volviendo a casa, accede a Toom Forum y acude al apartado de nabos. Observa que hay un anuncio que publicita un precio de quinientas cincuenta y cuatro bayas, debido a este precio se apunta. Como lleva consigo su Nintento Switch, acude a la isla para vender sus nabos.

2. **Creación de un evento.**

   Se acercan unos días de vacaciones y Laura no tiene nada planeado. Por lo tanto, decide crear un evento que finalice el último día del periodo vacacional. Accede desde su portátil a Toom Forum y pregunta por el *chat* qué clase de evento quiere la comunidad. Posteriormente, Laura comienza a planificar y preparar el evento. Una vez lo tiene listo, entra de nuevo a Toom Forum y crea el evento. Tras haberlo publicado, decide compartirlo entre sus amigos por Whatsapp y Telegram.

3. **Participación en el foro.**

   Laura está jugando a Animal Crossing un martes después de comer, ha tratado de crear un nuevo diseño de ropa pero tiene dudas sobre cómo realizarlo correctamente. Por lo tanto, decide acceder a Toom Forum, entra en el foro en la sección de ayuda. Crea un nuevo tema donde indica sus dudas y añade una imagen del problema que tiene. Tras unas horas, vuelve a acceder para ver si tiene alguna respuesta.

4. **Búsqueda de vecinos.**

   Laura está en su casa un sábado por la mañana, recientemente ha visitado las islas de otros jugadores y ha visto a un vecino que le gustaría tener. Por lo que, accede a Toom Forum en busca de un jugador que tenga ese vecino en el foro. Al final, decide ir a visitarle para poder traer al nuevo vecino a su isla.

5. **Ayuda a un jugador.**

   Es un viernes por la mañana y Laura se encuentra en la universidad, son las 11:00 y está descansando, saca su teléfono móvil y decide acceder a Toom Forum para contemplar las novedades. Encuentra un mensaje de un usuario preguntando sobre un artículo que se vende dentro del juego. Laura que ya lo posee, responde al usuario añadiendo una imagen para aclarar mejor su explicación.

Gracias al uso de estas técnicas, disponemos de la información suficiente para comenzar con el prototipo de la interfaz.

4.1.4. Prototipo de la interfaz

Ya contamos con los suficientes datos para plantear una interfaz de usuario adecuada. Para ello, se han empleado *Wireframes*, un *wireframe* es un plano de nuestra aplicación que permite otorgarle una estructura de una forma muy sencilla y esquemática.

Cuando buscamos la mejor UX, los prototipos forman una parte fundamental del proceso, nos permiten comprobar y validar si nuestras hipótesis y funcionalidades son las que el usuario busca. También, permite generar una mayor compresión del producto a desarrollar, dando lugar a cambios tempranos que de posponerse supondrían un mayor esfuerzo.

Un prototipo puede realizarse de varias formas, desde la más simple con un bolígrafo y papel, hasta la más avanzada, que sería realizar un prototipo funcional empleando tecnologías web como HTML y CSS. En este caso, se ha optado por una opción intermedia, ya que es fácil de realizar y no se requiere de tal nivel de validación para tener que implementar los prototipos y hacerlos funcionales.

Para el desarrollo de los prototipos de las diferentes partes de la interfaz se ha empleado Uizard [8]. Se trata de una herramienta creada en 2018 que ofrece un gran número de funcionalidades, como crear prototipos simples, hasta diseños complejos, siempre ofreciendo al usuario herramientas potentes y sin necesidad de tener conocimientos de diseño para lograr un resultado óptimo. En la figuras 4.2 y 4.3 se puede observar el *wireframe* correspondiente a la página principal.

Un punto importante a destacar es el enfoque que va a seguir la aplicación. En la investigación cualitativa se ha averiguado que la mayoría de usuarios emplean su dispositivo móvil para navegar en los sitios web. Debido a esto, se ha seguido un enfoque Mobile First. Este enfoque, cada vez más empleado, consiste en desarrollar el sitio web pensando en su diseño para móvil y posteriormente adaptarlo a escritorio. Actualmente, la mayoría de la población emplea a lo largo del día su dispositivo móvil, en España más del veinticuatro por ciento de la población pasa más de cinco horas al día con su *smartphone* [9]. A parte de todo lo que se ha mencionado anteriormente, Google da preferencia en mostrar estos sitios web sobre otros que carezcan de este enfoque.

Volviendo a los *wireframes* de la página principal 4.2 y 4.3, se ha optado por un diseño claro y no sobrecargado de elementos, donde el usuario podrá acceder sin necesidad de tener una cuenta asociada. En él podrá encontrar una descripción del sitio y la información de mayor interés como son los eventos, las publicaciones de nabos y los últimos mensajes publicados.

Antes de proseguir, vamos a aclarar unas características generales que afectan a todo el diseño. Siempre va a haber una barra de navegación en la parte superior. Otro dato importante es que, en el caso de que el usuario haya iniciado sesión, podrá visualizar siempre un botón en la parte inferior derecha que también le acompañará al desplazarse dentro de la página, este le permitirá abrir el *chat* y poder comunicarse con el resto de usuarios.

Otra de las pantallas más destacables de la aplicación se corresponde con la compraventa de nabos, figura 4.4. Aquí los usuarios podrán visualizar todos los anuncios publicados por los demás jugadores, además de poder añadir, modificar o eliminar sus propios anuncios. Una vez el usuario haga *clic* en un anuncio publicado, procederá al *wireframe* de la figura 4.5. En este se podrá consultar el total de participantes y obtener un mayor detalle de la publicación.

Poder comunicarse con el resto de una manera simple y ordenada se puede conseguir gracias a un foro. Permite separar los hilos de conversación de los usuarios según una

temática, donde cada usuario podrá crear su propio hilo de conversación. Este proceso se divide en varias partes que se recogen en las figuras 4.6, 4.7 y 4.8.

Crear eventos, concursos u otro tipo de actividades también es uno de los requisitos para desarrollar esta solución. Por lo tanto, se ha definido una vista que reúna todas estas características, se puede observar en la figura 4.9. Al *clicar* en un evento se podrán observar todos los detalles de este, así como poder participar o comentar, esta parte se observa en la figura 4.10.

Que los usuarios puedan obtener información del resto es algo relevante, ya que gracias a esto podrán conocer el PIN Dodo de los demás jugadores, este PIN permite realizar una visita a su isla, también pueden conocer los vecinos que tiene ese jugador actualmente en su isla. Este *wireframe* busca mostrar toda la información relevante sobre un usuario, figura 4.11.

Figura 4.2: *Wireframe* de la página principal 1

Figura 4.3: *Wireframe* de la página principal 2

Figura 4.4: *Wireframe* de Anuncios de Nabos

Figura 4.5: *Wireframe* de detalle de anuncio

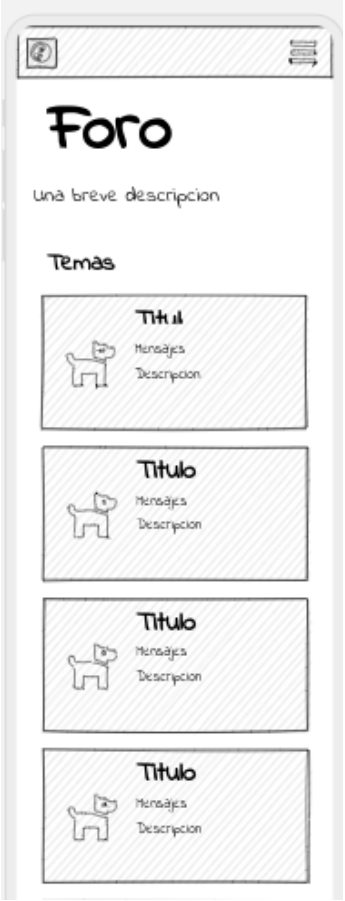


Figura 4.6: *Wireframe* Foro

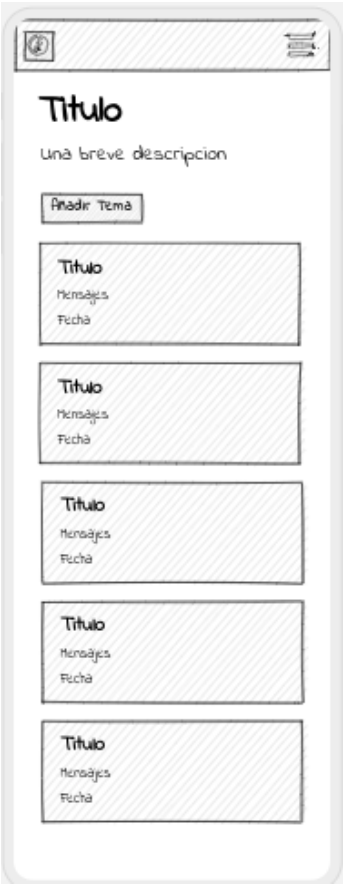


Figura 4.7: *Wireframe* Tema

Figura 4.8: *Wireframe* Detalles de un Tema

Figura 4.9: *Wireframe* Evento

Figura 4.10: *Wireframe* detalles de un Evento

Figura 4.11: *Wireframe* Perfil de Usuario

4.1.5. Validación

Una de las claves de DCU es valorar la **usabilidad** que ofrecen los prototipos, esta es definida como la facilidad que tiene un usuario para emplear una interfaz. En este caso, para evaluar la UX que ofrecen nuestros prototipos, vamos a emplear la **Escala de Usabilidad del Sistema** [10]. Se ha elegido este método porque resulta muy sencillo de aplicar y se obtienen unos resultados confiables.

Se va a emplear un cuestionario que consta de diez preguntas, siguiendo una escala Lickert, una escala empleada para evaluar la opinión y actitudes de una persona, en la que las puntuaciones oscilan entre uno y cinco. Una vez se han obtenido las puntuaciones para cada pregunta, se realiza una suma de todos los valores obtenidos, considerando lo siguiente: en las puntuaciones de las preguntas con número impar, se tomará la puntuación otorgada por el usuario y se le restara una unidad. En las preguntas pares, será de cinco menos la puntuación otorgada por el usuario. Las diez preguntas que forman nuestro cuestionario son las siguientes:

1. Creo que usaría esta aplicación frecuentemente.

2. Encuentro está aplicación innecesariamente complejo.
3. Creo que la aplicación fue fácil de usar.
4. Creo que necesitaría ayuda de una persona con conocimientos técnicos para usar está aplicación.
5. Las funciones de está aplicación están bien integradas.
6. Creo que la aplicación es muy inconsistente.
7. Imagino que la mayoría de la gente aprendería a usar está aplicación en forma muy rápida.
8. Encuentro que la aplicación es muy difícil de usar.
9. Me siento confiado al usar está aplicación.
10. Necesité aprender muchas cosas antes de ser capaz de usar está aplicación.

Los resultados obtenidos, contando con seis usuarios, son los siguientes: el primer usuario obtuvo una puntuación de setenta y tres puntos, mientras que el segundo una puntuación de ochenta y seis. El tercero una puntuación de setenta puntos, el cuarto de sesenta y siete puntos, el quinto ochenta y un puntos, el último una puntuación de setenta y cinco puntos. De esta manera, obtenemos una puntuación media de setenta y cinco con treinta y tres, que esta valorada entre buena y excelente. En la figura 4.12 se puede observar la métrica de evaluación.

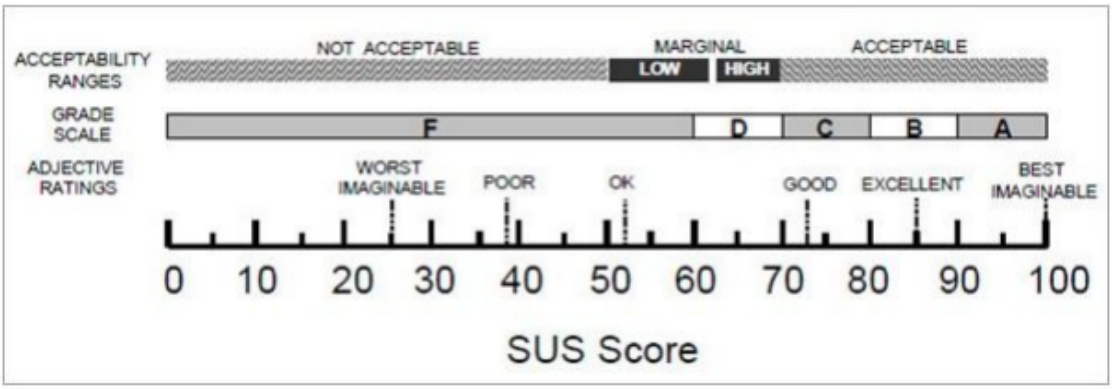


Figura 4.12: Puntuación SUS

4.2 Arquitectura

Ahora vamos a definir la arquitectura de nuestra aplicación web, un sitio web en la mayoría de los casos funciona mediante la arquitectura cliente-servidor. Esta arquitectura consta de dos partes, la primera de ellas es un cliente, en caso de un sitio web suele ser un navegador, en el otro extremo reside un servidor que está ejecutando un programa que permite que la web funcione. Su modo de funcionamiento consiste en que el cliente conoce la dirección del servidor, manda solicitudes de recursos al servidor, este las procesa y manda una respuesta para cada solicitud enviada por el cliente. Como se puede observar, la comunicación siempre la comienza el cliente.

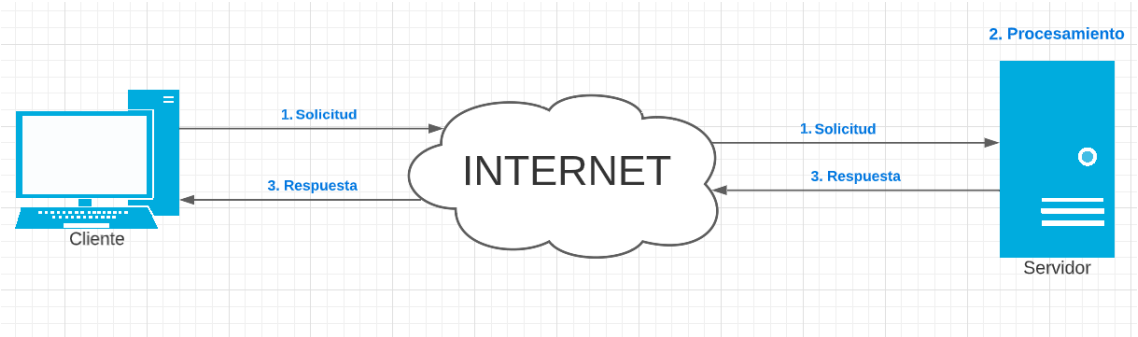


Figura 4.13: Arquitectura Cliente Servidor

Si nuestra aplicación web empleara únicamente un servidor para atender todas las peticiones, se podrían llegar a producir problemas de escalabilidad. Esto supondría que, por ejemplo, en el caso de una crecida repentina de clientes realizando solicitudes, nuestro servidor se pueda ver sobrecargado, en consecuencia, se produciría una decadencia en la calidad del servicio.

Este problema se puede solventar empleando varios servidores, que replican la aplicación que ejecutan y tienen delante de ellos un balanceador de carga. Este balanceador reparte la solicitudes de los clientes y selecciona a qué servidor es más indicado mandar esta solicitud. Otra forma también puede ser incrementando la capacidad de cómputo del servidor, pero no es la más adecuada, ya que tiene un límite.

La estrategia mencionada de colocar un balanceador de carga y replicas de los servidores nos es útil para poder satisfacer correctamente todas las solicitudes, pero aun queda un tema pendiente, la latencia. La latencia se puede definir como el tiempo que tarda en ir nuestra solicitud, ser procesada y recibir la respuesta. En este proceso juega un papel crucial la distancia que tenga que recorrer nuestra solicitud hasta el servidor. Si se pone nuestro servidor únicamente en España, provocaría que los clientes que accedan desde América tengan una calidad del servicio peor, con tiempos de carga superiores.

En los últimos años, las aplicaciones web son cada vez más empleadas, se puede apreciar cómo muchas versiones de escritorio han incorporado una versión web, como la *suite office* de Microsoft, esto es debido a la sencillez de uso, ya que no requieren de instalación por parte del usuario ni ocupan espacio físico en su ordenador. Debido a esto, ha sido necesario incorporar nuevas formas de distribuir el contenido para que un gran número de usuarios puedan acceder desde cualquier lugar del planeta, de una forma rápida y eficaz. Es aquí donde nacen las CDN y el *Edge Computing* [11].

Vamos a comenzar definiendo que es una CDN. Se trata de una red de distribución de contenido a nivel geográfico, son servidores interconectados en una red formada por una serie de *proxys* web y otros servidores, no relevantes en este caso. Un *proxy* es un intermediario entre el cliente y el servidor destino. En este caso, el *proxy* se encargara de entregar el recurso solicitado, siempre y cuando disponga de este recurso. Como la red está distribuida a nivel geográfico, normalmente a lo largo de múltiples países, permite que los usuarios tengan un acceso rápido a los recursos, además de un ahorro de ancho de banda y una disminución de la carga del servidor. Ahora bien, una CDN normalmente se encarga de distribuir contenido estático, el cual no cambia con relativa frecuencia, como por ejemplo, una web que muestra la carta de un restaurante.

Ya hemos visto que es una CDN, pero aun hay un problema presente, el contenido dinámico. Para poder solventar este problema vamos a emplear *Edge Computing*, en especial *Edge Functions*. Su funcionamiento es muy similar al de una CDN, siendo una red distribuida de servidores, donde se busca que la computación se produzca lo más cerca-

na posible al usuario que originó la solicitud. En nuestro caso, al emplear *Edge Functions*, lo que se va a ejecutar en el servidor van a ser funciones de nuestro código. Esto implica que no tengamos nuestras aplicaciones corriendo en un servidor o varios en un monolito, sino que las funciones de nuestra aplicación se van a repartir en varios servidores, pudiendo adaptar la carga de una forma muy sencilla. Por ejemplo, si tenemos una función que se encarga de recopilar todos los usuarios y comprobar los que tengan creado un evento y, simultáneamente muchos usuarios están realizando solicitudes requiriendo esta información, se puede crear una nueva instancia que atienda nuevas peticiones. Esta instancia únicamente ejecutará esta función.

Tanto *Edge Functions*, como las CDN, son empleadas en nuestra solución, esto nos lo proporciona **Vercel** de una forma sencilla con su *framework* Next.js, del cual hablaremos en el siguiente capítulo.

4.3 Modelado de datos

El modelo empleado para nuestra solución se puede observar en la figura 4.14. Está conformado por las siguientes tablas:

- **User:** representa a la cuenta de usuario, esta tabla juega un papel crucial en el modelo de datos, únicamente se recogen los datos indispensables para poder ofrecer todas las funcionalidades.
- **Image:** almacena toda la información referente a las imágenes. Inicialmente estaba planteado almacenar las imágenes directamente en la base de datos, pero debido a que es ineficiente cuando el número crece, se optó por emplear un servicio externo que las almacena. Por lo tanto, la base de datos guarda la información referente a las imágenes para poder acceder a ellas.
- **Island:** es la isla correspondiente a cada usuario, contiene toda la información, como la fruta autóctona que contiene, su código de sueño y los vecinos que tiene asignados.
- **Neighbour:** representa a un vecino, cada vecino contiene un gran número de atributos, actualmente existen trescientos noventa y un vecinos. Introducir esta cantidad de datos es inviable, debido a esto se ha empleado una *API* pública [12], de donde se han extraído e insertado en la base de datos para no depender de este servicio externo.
- **Advert:** almacena todos los anuncios que publican los usuarios referentes a la venta de nabos. En esta tabla se guarda qué usuario creó el anuncio, cuántos usuarios están participando, el precio, si el anuncio está activo, así como su duración, entre otros atributos.
- **Event:** esta tabla contiene los eventos creados por los usuarios, guarda el nombre del evento, fecha de inicio, comentarios, reglas, etcétera. Además de guardar el usuario que lo creó y los participantes como Advert.
- **Rule:** únicamente se encarga de registrar reglas para un determinado evento.
- **Comment:** registra los comentarios que crean los usuarios para un determinado evento, además de contener el usuario que lo ha creado.

- **Topic:** se encarga de mantener las diferentes secciones del foro, estas secciones son estáticas, los usuarios no van a poder añadir, eliminar o editarlas. Dentro de cada sección habrá una serie de temas.

- **Theme:** son los temas que se corresponden a una determinada sección del foro, guardan diferente información, como imágenes que pueden agregar los usuarios durante su creación, su fecha de creación o el usuario que lo creó.

- **ThemeMessage:** alberga el contenido de los mensajes creados por los usuarios dentro de un tema. Estos mensajes pueden ser respondidos y se muestra el mensaje padre e hijo, entonces, existe una relación reflexiva. También pueden contener una imagen, esta limitación se corresponde al almacenamiento limitado.

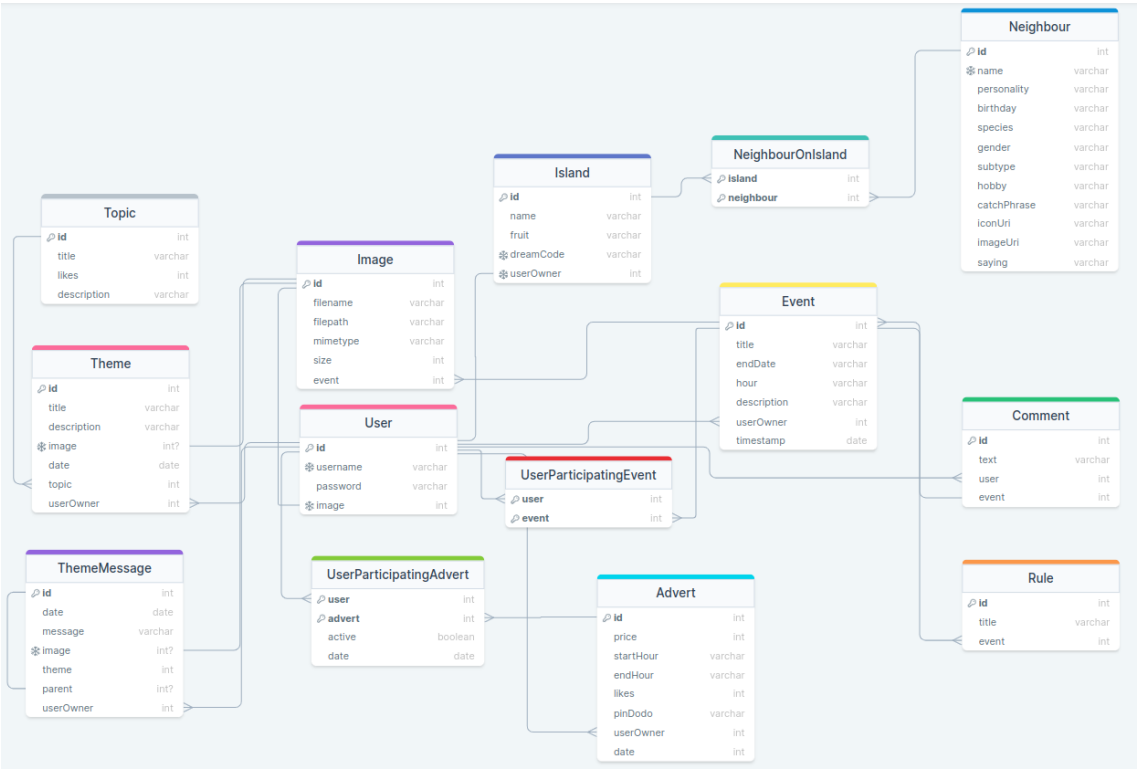


Figura 4.14: Modelo Base de Datos

4.4 Tecnologías Empleadas

En esta sección, se va a explicar en detalle cada tecnología empleada para desarrollar la aplicación y el por qué se ha elegido. Vamos a comenzar con las tecnologías base que son empleadas en la gran mayoría de los sitios web. Estas son: HTML5 [13], CSS3 [14] y Javascript [15].

4.4.1. HTML5

Es un lenguaje de marcas de hipertexto, es la base de la web que conocemos hoy en día, el cinco del final implica que es la quinta versión de este. Permite estructurar los elementos de una web empleando diversos elementos que nos proporciona, como pueden ser encabezados, títulos, párrafos, imágenes, etc. Es importante mencionar que HTML proporciona también un significado semántico, es decir, ayuda a organizar y distribuir el contenido del sitio web. Sobre esto volveremos más adelante para hablar sobre la accesibilidad.

HTML únicamente proporciona una estructura, por lo que nuestra web carecerá de estilo o funcionalidad, dando lugar a un sitio web similar a el primer sitio web de la historia Primera Web, muy lejos de los estándares actuales. Entonces, para hacerlo visualmente más agradable y otorgar una mejor UX, es necesaria la siguiente herramienta.

4.4.2. CSS3

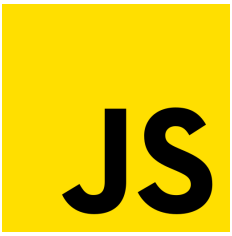
Son las siglas de Hojas de Estilo en Cascada, es un lenguaje de diseño que permite otorgar un estilo a contenido escrito en HTML, el tres del final indica la versión del lenguaje.

CSS permite otorgarle posición, color, ubicación, animaciones, tamaño, etc. a cualquier elemento, para ello, se emplean selectores que permiten lograr cualquier resultado que deseemos. Todo este conjunto crea una interfaz de usuario más amigable y proporciona una buena UX.

Dominar CSS puede llegar a ser complejo debido a que es muy fácil que se introduzca código repetido o inservible, ya que como su nombre indica los estilos se aplican en cascada, por lo tanto, un estilo que esté definido posterior a otro que se aplique al mismo elemento lo sobrescribirá. Más adelante, veremos una herramienta que nos ayudara a utilizar CSS.

Ahora tenemos una web con estructura y visualmente atractiva, pero carente de funcionalidad, que es otro pilar fundamental, es aquí donde entra **JavaScript**.

4.4.3. JavaScript



JavaScript es un lenguaje de programación, imperativo, débilmente tipado, dinámico, orientado a objetos, basado en prototipos e interpretado. En 1997, se declara como estándar ECMAScript, que a partir de entonces será considerada como la especificación de este lenguaje. También definen el DOM (Document Object Model) para evitar incompatibilidad en los navegadores, el DOM es una estructura en forma de árbol, que representa un documento HTML, ofrece una interfaz donde se puede modificar el estilo, contenido, funcionalidad, etc. Desde entonces, ha sido el lenguaje predilecto para la web.

Su versión más popular es ECMAScript 6 que llegó en 2015, introdujo un gran número de cambios, como la incorporación de clases, programación funcional o los módulos.

Hay que destacar que JavaScript no tiene nada que ver con Java, otro lenguaje de programación. Cuando JavaScript nació, Java era muy popular y decidieron añadirle *"Java"* al comienzo para darle notoriedad, realmente su nombre inicial fue Mocha.

Usualmente JavaScript ha sido empleado en el lado del cliente, es decir, normalmente en el navegador, pero su evolución ha sido tal que ya se ejecuta en el lado del servidor. Puede emplearse para sistemas en tiempo real, desarrollo de aplicaciones móviles, etcétera. Esto brinda una gran ventaja, esta es que con un único lenguaje puedes desarrollar una aplicación de escritorio, una aplicación móvil, un servidor web o una página web. Es importante destacar esto ya que, en este caso, el servidor también estará construido con JavaScript.

JavaScript ha avanzado mucho y se ha adaptado a las necesidades actuales, como pueden ser las clases o la introducción de *let* y *const*. Sin embargo, arrastra muchos problemas desde sus inicios, tales como el tipado dinámico y débil, que puede provocar errores inesperados en tiempo de ejecución. Es por esto que Microsoft decidió lanzar **TypeScript** [16].

4.4.4. TypeScript

Es un superconjunto de JavaScript de código abierto, su primera versión fue lanzada en 2012. Su principal característica es la introducción de tipos, es importante mencionar que esta comprobación únicamente se realiza en tiempo de compilación, en el cual se compila a código de Javascript.

En nuestro caso, ha sido empleado para garantizar que no hay problemas respecto a los tipos. Pero esta no es la única ventaja que nos ofrece, también agiliza el desarrollo,

ya que las variables y objetos, al estar tipados, ofrecen más información a nuestro editor de código, el cual nos podrá otorgar más datos en los autocompletados. Por ejemplo, en nuestro caso, se ha empleado Visual Studio Code [17] para el desarrollo de este proyecto.

4.4.5. Visual Studio Code

Es un editor de código creado por Microsoft, gratuito, de código abierto y disponible en todas las plataformas. Actualmente es el editor más empleado por los desarrolladores, esto es gracias a que otorga una interfaz de usuario agradable y sencilla, con mucha personalización y que incorpora un gran número de extensiones creadas por los usuarios que facilitan muchas tareas. La gran ventaja que tiene es que es muy modular y permite programar con la inmensa mayoría de lenguajes.

Para el desarrollo de nuestra aplicación se han empleado diversas extensiones que han resultado muy útiles, entre ellas destacan Prettier, Tailwind [18], Eslint [19], Docker [20] y Error lens.

4.4.6. Eslint

Es un analizar de código estático que permite detectar problemas típicos con JavaScript y mantener un estilo uniforme en todo el código. Por ejemplo, con él se puede configurar si se quiere emplear punto y coma, uso de comillas dobles o simples, entre un gran número de opciones. También otorga información sobre posibles mejoras. Permite la instalación de extensiones para dotarle de una mayor funcionalidad, en nuestro proyecto son las siguientes: TypeScript, React [21], React Hooks, Nextjs [22] y Prettier. Esta última permite además formatear el código y que se corrijan automáticamente los errores al guardar el fichero.

4.4.7. React

También llamado React.js, es una librería de JavaScript de código abierto desarrollada por Facebook en 2013 para la creación de interfaces de usuario, su principal atractivo es la creación de SPAs. Una SPA es una aplicación de una sola página, lo que implica que toda la página cargue una única vez. Esto otorga una mejor UX, con una navegación mucho más fluida. JavaScript permite hacer todo ello, pero de una forma mucho más compleja.

A continuación vamos a enumerar las principales características de React, ya que gracias a estas funcionalidades se ha seleccionado React para el desarrollo de la presente aplicación web.

- **Virtual DOM**. React proporciona un Virtual DOM, lo que le permite ser muy eficiente. Esto es porque se encarga de realizar los cambios en el Virtual DOM y únicamente manipula el fragmento modificado en el DOM real, en vez de actualizarlo completamente.
- **JSX**. Se trata de una extensión del lenguaje de JavaScript que emplea React. Permite mezclar HTML con código de JavaScript, lo cual lo hace muy cómodo debido a que no es necesario recurrir a plantillas, ni otras soluciones.
- **Componentes**. React permite crear componentes, los cuales pueden mantener un estado, únicamente cuando cambia dicho estado el componente volverá a renderizarse mostrando el estado actualizado.

Estas son algunas de las claves de React, aunque esta librería ofrece mucho más como son los *hooks*, que también han sido empleados en este proyecto.

React tiene un problema que impide que se ofrezca una buena UX cuando se quiere crear un proyecto serio. Este es que únicamente emplea CSR (Client Side Rendering), lo que significa que el cliente descarga una página web completamente vacía. Una vez el cliente ha obtenido el código JavaScript creará la página, lo cual supondrá varios problemas. El primero de ellos es que, si el cliente desactiva JavaScript, lo único que obtendrá será una web en blanco. Otro problema es el tiempo que tarda el cliente en generar todo el contenido y lo que esto implica para la experiencia de usuario. Aparte de la UX, provoca que nuestra página web no sea posicionada correctamente en Google. Cuando Google accede a nuestra web tiene que generar todo el contenido, al igual que un cliente normal, y esto demora tiempo, por lo tanto, no tiene tanto tiempo para poder recorrer toda nuestra web como si el contenido ya hubiera sido generado a falta de otorgarle funcionalidad.

Es aquí donde entra Server Side Rendering [23], en adelante SSR. Al contrario que CSR, la página la renderiza el servidor y la envía ya renderizada al cliente. Esto tiene muchas ventajas, ya que nuestra página estará mejor posicionada, otorgará una mejor UX y un menor tiempo de carga.

Como React no ofrece esta funcionalidad de forma nativa, se ha recurrido a un *framework* de React creado por Vercel llamado **Next.js**, que se detallará a continuación.

4.4.8. Next.js

Vamos a comenzar definiendo que es un *framework*, es una herramienta que nos otorga unas bases para acelerar el proceso de desarrollo, por ejemplo, proporciona funcionalidad ya implementada, como puede ser organizar el contenido y configuración predeterminada. En este caso, Next.js nos ofrece SSR, optimización de imágenes, creación de una API, optimización del SEO, entre otras.

La optimización de imágenes es muy útil, ya que dinámicamente y de forma automática adapta la resolución de la imagen al dispositivo empleado, logrando reducir el peso de la imagen y mejorando la UX al tener un menor tiempo de carga.

4.4.9. Tailwindcss

También es denominado Tailwind. Se trata de un *framework* de CSS. Aunque a día de hoy hay un gran número de *frameworks* para CSS, como por ejemplo **Bootstrap** (uno de los más usados), la gran diferencia que ofrece Tailwind respecto al resto es que no incorpora unas clases que ya tienen un estilo predefinido para algunos elementos como pueden ser botones, encabezados, etcétera. El principal problema de estos *frameworks* es que al final todas las páginas web creadas con ellos se ven de una forma muy similar, ya que no permiten grandes modificaciones. Sin embargo, Tailwind genera unas clases que permiten aplicar estilos individuales a un elemento, aunque hay que tener un mayor conocimiento de CSS para aplicarlo.

Otro beneficio de emplear Tailwind es la optimización que ofrece, ya que el código de CSS que no es necesario, lo elimina y optimiza al compilar la aplicación. Es por todo esto que cada vez se está volviendo más popular.

4.4.10. Node.js

Node.js es un entorno de ejecución que emplea el motor V8 de Chrome. Puede ejecutar código de JavaScript en el servidor. Como ya se mencionó anteriormente, nuestra parte de back-end va a estar implementada con JavaScript y gracias al empleo de Next.js podemos definir nuestra *API*, en el mismo proyecto.

4.4.11. PostgreSQL

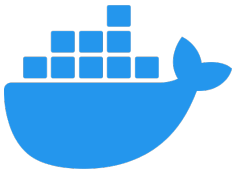
PostgreSQL [25], también llamado Postgre, es un gestor de bases de datos relacional orientado a objetos. Este gestor está ganando popularidad últimamente, ya que permite alta concurrencia y una gran variedad de tipos, entre otras características.

4.4.12. Prisma

Prisma [26] es un ORM. Este último término es, a grandes rasgos, un mapa objeto relacional donde están relacionados la programación orientada a objetos, el sistema de tipos y una base de datos relacional.

En este caso, Prisma se apoya en TypeScript y Node.js para crear de forma automática una asociación entre un esquema relacional definido con sus objetos y tipos correspondientes. Además de incorporar métodos que permiten crear, modificar, eliminar, entre muchos otros. Todo esto facilita mucho la conexión entre la capa de persistencia y la lógica.

4.4.13. Docker



Docker proporciona un mecanismo de gestión de contenedores, lo que permite separar la aplicación de la infraestructura. Un contenedor es un entorno aislado del sistema operativo anfitrión, y el resto de contenedores ocupan muy poco espacio debido a que no cargan un sistema operativo completo. Por lo tanto, usando Docker, podemos lograr ejecutar nuestra aplicación en cualquier entorno.

En este caso, Docker se ha empleado durante el desarrollo. En él se ha desplegado la aplicación, además de una instancia de Postgre, lo cual ha permitido que no sea necesario tener instalado nada en nuestro sistema anfitrión, ya que todo está en el contenedor.

CAPÍTULO 5
Desarrollo de la solución

En este punto se va a detallar el proceso de creación de la aplicación web. Para ello, haremos uso de las tecnologías que se han mencionado anteriormente, además de aplicar el diseño en base a los *wireframes* planteados.

El comienzo de esta fase juega un papel fundamental para que el proyecto al final del desarrollo esté bien estructurado y sea mantenible a largo plazo. Por lo tanto, hay que tener clara la estructura que vamos a seguir respecto a la organización de carpetas, las dependencias que se van a usar y cómo organizar nuestro código.

5.1 Mapa de Navegación

Gracias al enrrutamiento se puede dirigir cada petición a su ruta correspondiente para que se muestre cada página según corresponde. En Next.js este enrutamiento lo podemos definir mediante la estructura de directorios y archivos. En la figura 5.2 se puede observar el **mapa de navegación**. Este mapa de navegación ha sido posible gracias a las estructura de la figura 5.1

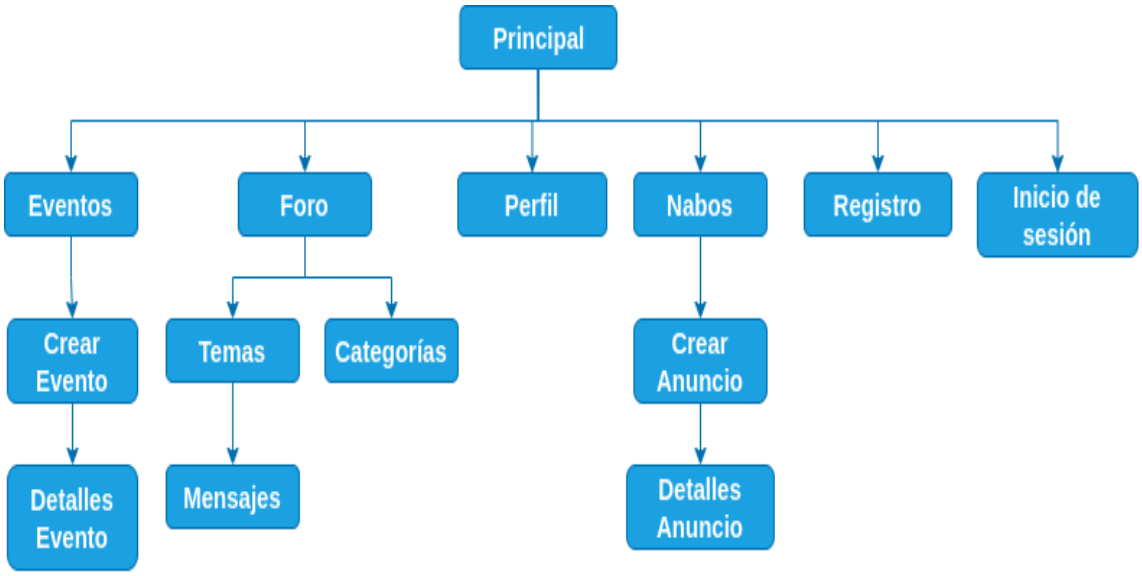


Figura 5.1: Mapa de navegación

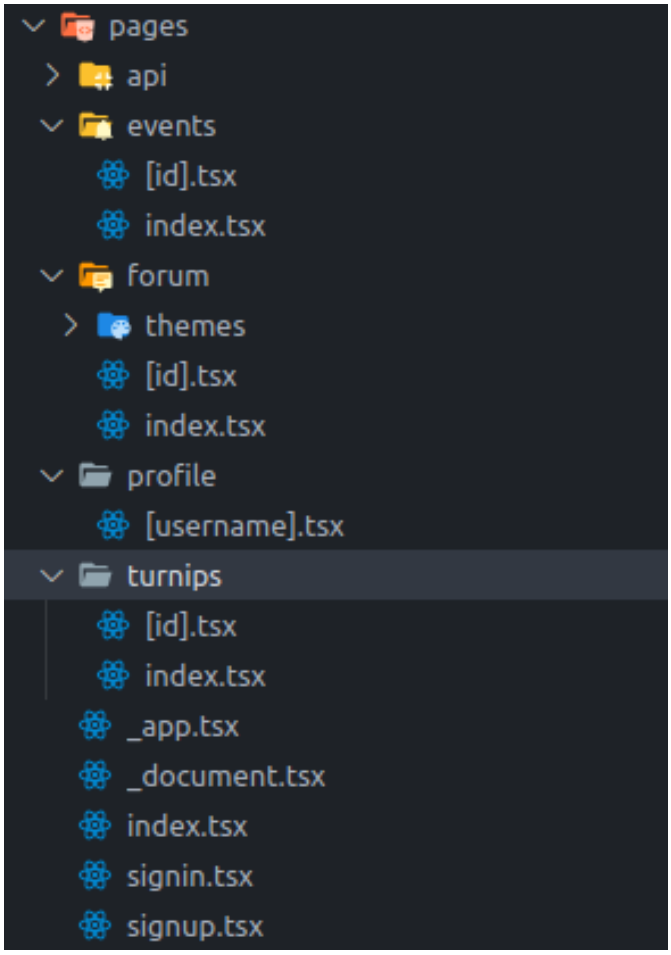


Figura 5.2: Estructura de las rutas

Los archivos que contienen [x].tsx sirven para indicar a Next.js el nombre del parámetro con el que nos referiremos en nuestra implementación. Por ejemplo, [id] permitirá extraer de la ruta el parámetro *id* con la información que contiene.

5.2 Pre-renderizado

Next.js ofrece tres formas de renderizar el contenido:

- **Client side rendering:** no requiere de configuración y su funcionamiento se basa en que el navegador solicita todos los recursos al servidor, este le devuelve todo el JavaScript necesario para poder renderizar la página. Una vez el navegador obtiene todos los datos, construye la página por completo.
- **Static file generation:** para ella hay que indicarle a Next.js que se trata de una página estática. Con motivo de esto, cuando la aplicación es generada, se renderiza el HTML y se empleará siempre el mismo para cada solicitud.
- **Server side rendering:** esta forma es la más empleada en nuestra solución, ya que la gran mayoría de páginas contienen contenido dinámico. Al igual que *Static File Generation*, hay que indicar que se va a emplear este tipo de renderizado y proporcionar una implementación para obtener los datos necesarios, en este caso, el HTML es generado para cada solicitud entrante.

```
export const getServerSideProps: GetServerSideProps = async () => {
  const adverts = await getAdvertsToday()

  const events = await getEventsWithOwner()

  const messages = await getMessageWithThemeAndUserOwner()

  const reducedAdverts = adverts.slice(0, 9)
  const reducedEvents = events.slice(0, 9)
  const messagesReduced = messages.slice(0, 9)

  const parsedAdverts = JSON.parse(JSON.stringify(reducedAdverts))
  const parsedEvents = JSON.parse(JSON.stringify(reducedEvents))
  const parsedMessages = JSON.parse(JSON.stringify(messagesReduced))
  return {
    props: {
      adverts: parsedAdverts,
      events: parsedEvents,
      messages: parsedMessages,
    },
  }
}
```

Figura 5.3: Método para indicar a Next.js usar SSR

En la figura 5.3 se puede observar la implementación de SSR en la ruta «/». Los tres primeros métodos obtienen los anuncios de ese día, los eventos y mensajes. Para ello, realizan una llamada directamente a la base de datos, sin necesidad de tener que llamar a una ruta de la *API*, ya que este código no se ejecuta en el cliente, se ejecuta en el lado del servidor. Estos datos que se han obtenido se pasan a el componente y permiten que el servidor pueda generar toda la página antes de enviarla al cliente.

5.3 API

En la mayoría de sitios web es necesario establecer una comunicación con el servidor, ya sea para obtener datos, publicarlos o editarlos. Esta comunicación se establece mediante una *API*, que permite a dos productos *software* distintos establecer una comunicación a través de protocolos y definiciones, sin necesidad de conocer la implementación de los servicios que ofrece. Actualmente, en el mundo web, predominan dos tipos de *APIs* para comunicarse con el servidor, Restful y GraphQL. En nuestro proyecto se ha empleado Restful.

En la mayoría de aplicaciones se define, por un lado, un proyecto con el *front-end* y, por otro, con el *back-end*. Si se emplea Next.js esto puede no ser así, este *framework* permite en el mismo proyecto incorporar ambos. Esta función es realmente útil, ya que se puede emplear de una forma más sencilla el sistema de tipos. Como se vio en la sección x, dentro del directorio routes existe otro llamado api, en la figura 5.4 está la estructura que se ha seguido.

El funcionamiento del enrrutamiento es el mismo que el que se ha usado para las rutas de la página web, la sintaxis es muy parecida a Express.js, un *framework* de node.js que permite la creación de servidores web de una forma sencilla. En la figura 5.5 se puede observar cómo funciona un *endpoint*, en este caso, el correspondiente a la creación de anuncios de nabos. Vamos a detallar los diferentes pasos que se realizan, el primero de ellos es extraer el método HTTP que se ha empleado.

El *endpoint* de la figura 5.5 únicamente responderá a solicitudes del tipo POST, entonces, ya nos podemos hacer a la idea de que se tratará la creación de anuncios. El método *parseTurnipAdvert* comprueba que el body de la solicitud contenga los campos correctos, para ello, se emplea TypeScript y permite lanzar un error en el caso de que algún campo falle. Inicialmente body tiene el tipo *any* de forma implícita, esto significa que podría tener cualquier valor. Después de llamar a la función, obtenemos el tipo correspondiente a la creación de un anuncio. Posteriormente, podemos ver un ejemplo de cómo funciona Prisma. En este caso, para la creación de un anuncio le indicamos los datos necesarios para la creación, cuando esté creado, nos devolverá el anuncio incluyendo todos los datos del usuario creado. Por último, se envía al cliente el anuncio creado, o un error si su solicitud es errónea.

Para poder comprender mejor cómo funciona la *API* se puede contemplar su documentación en las figuras 5.6 y 5.7.

Como se mencionó en la sección 5.2, desde el propio componente se pueden realizar llamadas directas a la base de datos sin necesidad de tener que llamar a la API. Es por esto que, en la API, faltan muchas solicitudes del tipo GET, ya que los datos se extraen directamente de la base de datos al cargar la página.

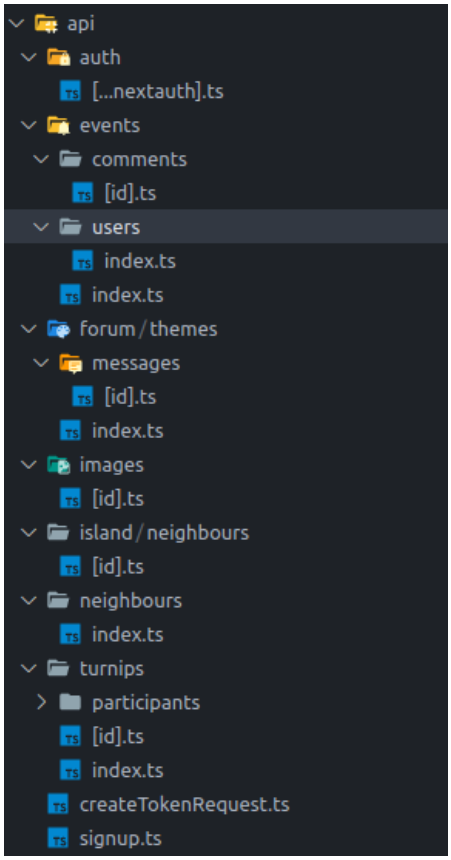


Figura 5.4: Estructura de la API

```
const handler = async (req: NextApiRequest, res: NextApiResponse) => {
  const { method, body } = req

  switch (method) {
    case 'POST':
      try {
        const turnipAdvert = parseTurnipAdvert(body)
        const { endHour, pinDodo, price, startHour, userOwnerId } = turnipAdvert
        const savedAdvert = await prisma.advert.create({
          data: {
            price,
            endHour,
            startHour,
            pinDodo,
            userOwnerId,
          },
          include: {
            userOwner: true,
          },
        })
        return res.status(200).json(savedAdvert)
      } catch (err) {
        if (err instanceof Error) {
          return res.status(400).json({ message: err.message })
        }
      }
  }
}

export default handler
```

Figura 5.5: Endpoint Anuncios

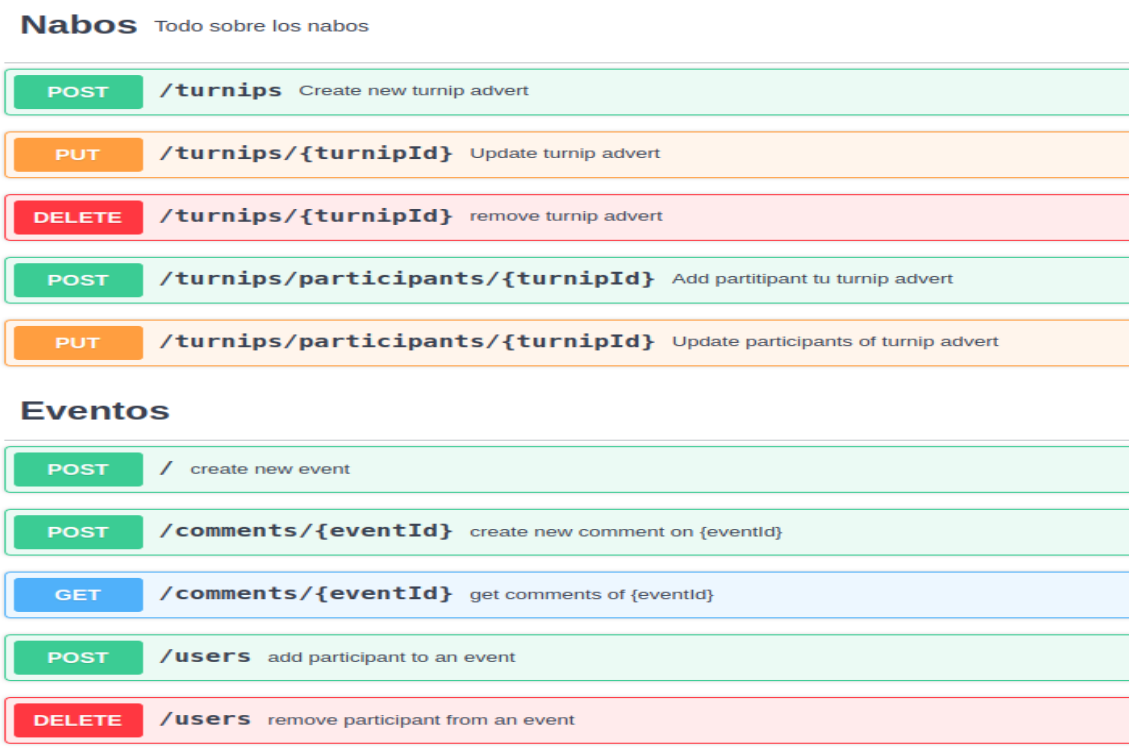


Figura 5.6: Documentación Swagger 1

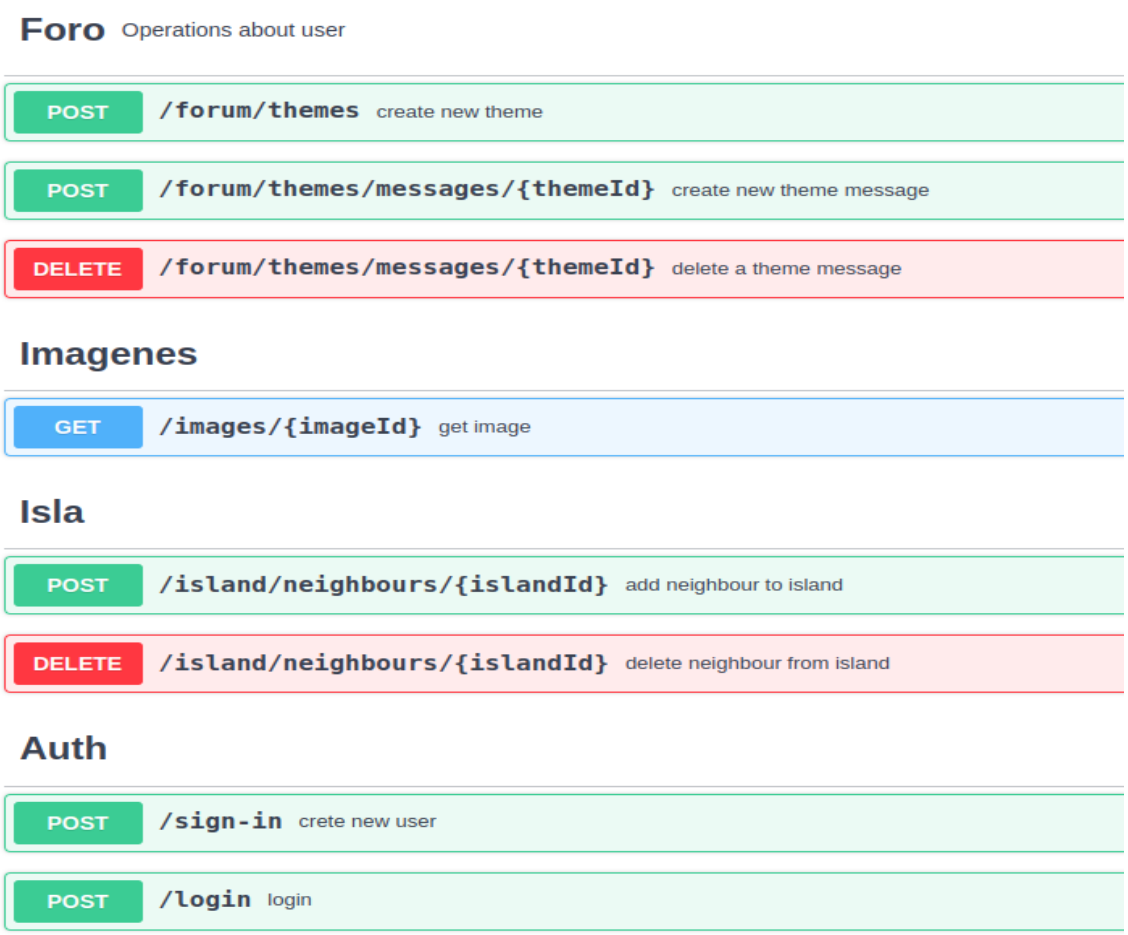


Figura 5.7: Documentación Swagger 2

5.4 Componentes

El proyecto cuenta con cuarenta y tres componentes, no todos ellos tienen la misma relevancia. Debido a esto, vamos a repasar algunos de los más relevantes que han supuesto un mayor reto a la hora de implementarlos y se ha buscado que sean reutilizables.

Antes de detallar su funcionamiento, se van a aclarar brevemente dos conceptos que permitirán comprender mejor cómo están implementados estos componentes. El primer concepto es el **estado**, en *React* el estado se encarga de mantener un valor y cuando este cambia, se vuelve a renderizar el componente, mostrando el estado actualizado. El segundo concepto es un ***hook***, a grandes rasgos, es una función que puede ser llamada desde otro componente y permite manipular el estado o el ciclo de vida de un componente.

5.4.1. Toggable

El primero de ellos es un componente llamado ***Toggable.tsx***. Su principal cometido es mostrar un botón, cuando este botón es presionado se mostrará un contenido, como podemos contemplar en la figura 5.8. El botón que permitía mostrar el contenido se ocultará. Entonces, se mostrará otro botón que permitirá ocultar el contenido, como se puede observar en la figura 5.9.

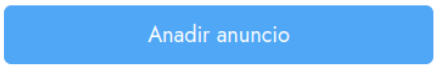


Figura 5.8: Botón Toggable

Figura 5.9: Contenido Toggable

Una vez detallado cómo funciona, vamos a ver su implementación en la figura 5.10. Como se puede contemplar, la primera parte llamada **Estado** se encarga de guardar el valor de si se muestra el contenido o no. El cuadrado verde contiene variables que servirán para indicar en el código JSX qué elementos se van a mostrar, por otro lado, la función que invierte el valor del estado. Por último, está el código JSX en el rectángulo azul, que muestra la vista del componente, se puede observar que los botones contienen un *event listener* en *onClick* y los **divs** tienen su estilo dependiente de las variables del rectángulo verde. El objeto *props* contiene los datos que le suministra el componente padre. Al lograr esta abstracción de ocultar cualquier contenido y permitir mostrar el texto que queramos en los botones, se ha podido emplear en multitud de páginas. Ahora vamos a detallar las *props* que recibe este componente:

- **labelShow:** Texto del botón que muestra el contenido.
- **children:** Contenido que va a mostrar cuando se haga clic en el botón. En el ejemplo mostrado en la figura 5.9 se corresponde con el formulario.
- **labelCancel:** Texto que se muestra para ocultar el contenido.

Figura 5.10: Implementacion Toggable

5.4.2. Custom Input

Para la validación de formularios se ha empleado una librería llamada *React Hook Form*. A grandes rasgos, su funcionamiento consiste en registrar cada *input* en el *hook* proporcionado por la librería, indicándole las validaciones que se deben cumplir. Esta librería también proporciona los errores que se han producido para poder mostrarlos al usuario, en la figura 5.11 se muestra un ejemplo de su implementación.

```
<input {...register("firstName", { required: true })} />
{errors.firstName?.type === 'required' && "First name is required"}

<input {...register("lastName", { required: true })} />
{errors.lastName && <p>Last name is required</p>}

<input {...register("mail", { required: "Email Address is required" })} />
<p>{errors.mail?.message}</p>
```

Figura 5.11: Ejemplo React Hook Form

En la aplicación web se emplean multitud de formularios, por lo tanto, hacer este procedimiento para cada *input* es tedioso. Por eso, se ha creado un componente reutilizable para que introducir *inputs* en los formularios sea un proceso sencillo y que no recaiga en el formulario la necesidad de controlar todos los *inputs*. En el rectángulo azul de la figura 5.12 está la interfaz que define el tipo de las *props*. En el rectángulo verde se realiza

el registro en el *hook* y el manejo de errores. Además de indicar las demás propiedades relevantes del *input*. En la figura 5.13 se puede contemplar un ejemplo de uso de este componente en el formulario de registro de usuario.

```tsx
You, 3 weeks ago | 1 author (You)
interface CustomInputProps<TFormValues extends FieldValues> {
  label: string
  name: Path<TFormValues>
  type: TypeInput
  rules: RegisterOptions
  register: UseFormRegister<TFormValues>
  errors: Partial<DeepMap<TFormValues, FieldError>>
  placeholder: string
}

type TypeInput = 'text' | 'number' | 'password'

const CustomInput = <TFormValues extends FieldValues>({
  label,
  name,
  type,
  register,
  errors,
  placeholder,
  rules,
}: CustomInputProps<TFormValues>) => {      You, 3 weeks ago • refactors …
  return (
    <div className="my-3">
      <label>{label}</label>
      <div className="py-2 w-full rounded-md border-blue-200 border-2 px-1 flex items-center hover:border-primary-lime-green">
        <input
          type={type}
          placeholder={placeholder}
          className="w-full mx-2 hover:outline-0 focus:outline-0"
          {...(register && register(name, rules))}
        />
      </div>

      <ErrorMessage
        errors={errors}
        // eslint-disable-next-line @typescript-eslint/no-explicit-any
        name={name as any}
        render={({ message }) => <p className="text-red-400">{message}</p>}
      />
    </div>
  )
}

export default CustomInput
```

Figura 5.12: Implementación *Custom Input*

```tsx
<div className="flex flex-col lg:grid grid-cols-2 gap-x-3">
  <CustomInput
    label="Usuario"
    errors={errors}
    name="username"
    placeholder="Nombre de usuario ... "
    type="text"
    rules={{
      required: 'El nombre de usuario es obligatorio',
      minLength: {
        message: 'El nombre de usuario debe tener mas de 4 caracteres',
        value: 5,
      },
    }}
    register={register}
  />

  <CustomInput
    label="Contrasena"
    errors={errors}
    name="password"
    placeholder="Contrasena ... "
    type="password"
    rules={{
      required: 'La contrasena es obligatoria',
      minLength: {
        value: 6,
        message: 'La contrasena debe tener mas de 5 caracteres',
      },
    }}
    register={register}
  />
```

Figura 5.13: Ejemplo de uso *Custom Input*

5.4.3. Autocomplete

Al incorporar la funcionalidad para que el usuario introduzca los vecinos que están presentes en su isla, surgió un problema. Actualmente ACNH cuenta con trescientos sesenta y siete vecinos, es inviable que el usuario tenga que introducir el nombre de cada vecino, con los errores que podría cometer durante este proceso. Debido a este inconveniente, se buscó una solución que tratara de completar automáticamente la información, a medida que el usuario introdujera el nombre del vecino. Tras una larga búsqueda, no se encontró ningún componente o librería creado por la comunidad que fuera útil para esta tarea, así que, se decidió implementar manualmente.

Realmente no es un componente único, si no dos. Uno de ellos se encarga de filtrar la búsqueda y de mantener el estado del *input* y, el otro, de renderizar la lista de resultados. Se ha diseñado de esta forma para que, si en un futuro se quiere cambiar como se muestran los resultados al usuario, únicamente haya que modificar un componente.

Vamos a explicar cómo funciona únicamente el componente *Autocomplete*, el encargado de la lógica de filtrado, ya que el otro únicamente se encarga de renderizar una lista de elementos que le pasa el componente *Autocomplete*. En la figura 5.14 está la implementación de este componente y en la figura 5.15 el *hook* que se encarga de la lógica.

```tsx
import { Neighbour } from '@prisma/client'
import { useAutocomplete } from '../../hooks/autocomplete'
import AutocompleteList from './AutocompleteList'

Linuxgunter, 2 months ago | 1 author (Linuxgunter)
interface AutocompleteProps {
  value: string
  handleChange: (event: string) => void
  handleSelected: (neighbour: Neighbour) => void
  fetchData: () => Promise<Array<Neighbour>>
}

const Autocomplete = ({
  value,
  handleChange,
  handleSelected,
  fetchData,
}: AutocompleteProps) => {
  const { handleChangeAutocomplete, onClick, showSuggestions, suggestions } =
    useAutocomplete({ fetchData, handleChange, handleSelected })
  return (
    <>
      <div className="py-2 w-full rounded-md border-blue-200 border-2 px-1 flex items-center
        <input      You, 4 weeks ago • interface and profile …
          value={value}
          onChange={handleChangeAutocomplete}
          placeholder="Nombre del vecino"
          className="w-full mx-2 hover:outline-0 focus:outline-0"
        />
      </div>

      {showSuggestions && value && (
        <AutocompleteList clickSuggestion={onClick} suggestions={suggestions} />
      )}
    </>
  )
}

export default Autocomplete
```

Figura 5.14: Componente *Autocomplete*

```
export const useAutocomplete = ({
  fetchData,
  handleChange,
  handleSelected,
}: autocompleteHookProps) => {
  const [data, setData] = useState<Array<Neighbour>>([])

  const [suggestions, setSuggestions] = useState<Array<Neighbour>>([])
  const [showSuggestions, setShowSuggestions] = useState<boolean>(false)

  useEffect(() => {
    fetchData().then((data) => setData(data))
  }, [fetchData])

  const handleChangeAutocomplete = (
    event: React.ChangeEvent<HTMLInputElement>
  ) => {
    handleChange(event.target.value)
    if (event.target.value.length >= 2) {
      const filteredSuggestions = data.filter((data) =>
        data.name.toLowerCase().startsWith(event.target.value.toLowerCase())
      )
      setSuggestions(filteredSuggestions)
      setShowSuggestions(true)
    }
  }

  const onClick = (event: React.MouseEvent<HTMLLIElement>) => {
    const neighbour = suggestions.filter(
      (suggestion) =>
        suggestion.name.toLowerCase() ===
        event.currentTarget.innerText.toLowerCase()
    )
    handleSelected(neighbour[0])
    setSuggestions([])
    handleChange(event.currentTarget.innerText)
    setShowSuggestions(false)
  }

  return {
    onClick,
    showSuggestions,
    suggestions,
    handleChangeAutocomplete,
  }
}
```

Figura 5.15: *Hook* para *Autocomplete*

Lo más destacable es la lógica que implementa, el primer método se encarga de traer los datos que sean necesarios antes de renderizar el componente. El segundo, se ejecuta cada vez que el usuario introduce un carácter en el *input*, donde se comprueba para cada vecino existente si comienza con la cadena de caracteres introducida por el usuario. Tras esto, actualiza el estado indicando las sugerencias que se deben mostrar. El último método selecciona al vecino introducido por el usuario, actualiza el estado correspon-

diente al vecino seleccionado para que se pueda mostrar con mayor detalle en la interfaz y actualiza la lista de sugerencias.

5.4.4. Pages

Al igual que el componente anterior, surge de una necesidad que no ha podido satisfacerse con componentes ya publicados.

Al crear las rutas de temas y mensajes, se presentó el problema mencionado en el párrafo anterior. En el caso de que hubiera un número elevado de resultados, podría provocar un mal rendimiento el hecho de tener tantos resultados, además de la dificultad para el usuario de navegar por la página. Por lo tanto, se creó un componente modular que puede ser empleado en cualquier ruta para emplear la paginación. Al igual que el componente anterior, está compuesto por un componente y un *hook*, pero en este caso, únicamente vamos a mostrar el *hook*, ya que el componente se encarga de renderizar las páginas para que el usuario pueda navegar.

Este componente ha resultado complejo de elaborar, debido a que los datos de las páginas no son estáticos. Un usuario puede añadir o eliminar mensajes y temas, lo que provoca que el contenido de estas páginas varíe a lo largo del tiempo, pudiendo seguir o eliminarse nuevas páginas.

En la figura 5.16 está detallada una parte de la implementación del *hook*. Este *hook* se encarga de mantener los datos, para poder gestionar las páginas mostradas al usuario de forma correcta.

El primer método que se puede contemplar en la figura 5.16 se encarga de manejar y actualizar la página actual donde se encuentra el usuario. La comprobación que realiza es debido a que el número de página se guarda en la *URL* para que el usuario pueda guardar en favoritos o compartir el enlace y que se muestren esos resultados, por lo tanto, el usuario podría manipular la *URL* y provocar un error.

Los dos siguientes métodos controlan la adición o eliminación de páginas. Estas son llamadas por otras dos funciones que faltan por ver. Estas últimas son las que se encargan de añadir o eliminar datos y, si es necesario, añaden o eliminan páginas.

Se emplean dos *hooks* proporcionados por *React*, no se va profundizar en ellos. Basta con saber que se encargan de mejorar el rendimiento aplicando técnicas de memorización y que no se ejecutan las funciones cada vez que cambie un estado.

```tsx
export const usePages = <T extends { id: number }>({
  pages,
  currentPage,
  startData,
}: usePageProps<T>) => {
  const [actualPage, setActualPage] = useState<number>(currentPage)
  const [actualPages, setActualPages] = useState<number>(pages)
  const [data, setData] = useState(startData)
      You, 4 weeks ago • pre …
  const handleChangePage = useCallback(
    (page: number) => {
      if (page > 0 && page ≤ actualPages) {
        setActualPage(page)
      }
    },
    [actualPages]
  )

  const addNewPage = useCallback(() => {
    setActualPages((prevPages) => prevPages + 1)
  }, [])

  const removePage = useCallback(() => {
    setActualPages((prevPages) => prevPages - 1)
  }, [])

  const addData = useCallback(
    (newData: T) => {
      setData((prevData) => {
        const copy = prevData.slice(0)
        if (copy[actualPages - 1]?.length === 10) {
          addNewPage()
        }
        const res = paginate(copy.flat().concat(newData))
        return res
      })
    },
    [addNewPage, actualPages]
  )
```

Figura 5.16: Componente *Autocomplete*

5.4.5. Notification

Notificar al usuario sobre los errores que comete o sobre resultados de sus acciones es crucial. El comportamiento de esta notificación es siempre el mismo, por lo tanto, lo podemos trasladar a un componente para facilitar su modularidad.

Además, en este componente (figura 5.17), se puede observar cómo se ha hecho que Toom Forum sea más accesible. En la etiqueta «p» se indica un atributo llamado *role*, este atributo permite asignar un *rol* a un elemento que semánticamente no define la acción que realiza. En este caso, es conveniente indicar que su *rol* es de tipo *alert*.

Este componente por sí mismo no es funcional, ya que no mantiene ningún estado referente a la notificación, es por eso que también existe un *hook* para otorgarle la funcionalidad. En este *hook* se define un tipo que indica el tipo de notificación se va a emplear (error o acierto), aquí se puede visualizar cómo *TypeScript* es de gran ayuda ya que permite definir claramente las posibilidades que se contemplan, además de no permitir que se indique un tipo distinto.

```tsx
import { memo } from 'react'
import { Notification } from '../hooks/useNotification'

You, 4 weeks ago | 1 author (You)
interface NotificationProps {
  notification: Notification
}

const Notification = ({ notification }: NotificationProps) => {
  if (notification.type === 'success') {
    return (
      <p
        role={'alert'}
        className="text-center text-primary-lime-green text-lg alert"
      >
        {notification.message}
      </p>
    )
  } else {
    return (
      <p role={'alert'} className="text-center text-red-500 text-lg alert">
        {notification.message}
      </p>
    )
  }
}

export default memo(Notification)
```

Figura 5.17: Componente Notification

5.5 Usabilidad

Vamos a comentar la usabilidad de nuestro sistema. Para ello, empleamos los diez principios de Jakob Nielsen, que es considerado uno de los padres de la experiencia de usuario y cuenta con un gran concomiendo de la usabilidad web.

1. **Visibilidad del estado del sistema:** este principio se refiere a que el usuario debe ser consciente de qué está ocurriendo en el sistema y dónde se encuentra actualmente. En nuestra aplicación, este principio se puede ver manifestado cuando el usuario tenga una sesión activa, en ese caso, visualizará en la parte superior su foto de perfil. Esto es muy útil debido a que el navegador almacena la sesión y esto permite al usuario no tener que iniciar sesión cada vez, por lo tanto, al acceder tras un tiempo podrá observar si ya está identificado o no. Otro indicador importante es que, al realizar una acción como añadir un anuncio, se muestra un indicador de carga. Respecto a la navegación, se muestra resaltado de un color diferente el lugar donde se encuentra el usuario actualmente.

2. **Adecuación entre el sistema y el mundo real:** se define como el parecido que tiene que existir entre lo que es familiar al usuario y lo que el usuario está visualizando. En la página se contempla cómo el icono de enviar un mensaje es similar al empleado en *Whatsapp*, que lo usan la gran mayoría de usuarios. Además se usa el color rojo para indicar errores o acciones que implican eliminar algo.

3. **Libertad y control por el usuario:** cuando un usuario comete un error debe tener la opción de volver atrás y poder solventarlo. En nuestro sitio web, este principio se ve reflejado, por ejemplo, con la creación de anuncios de nabos, donde el usuario podrá modificar su anuncio o eliminarlo de una forma sencilla e instantánea.

4. **Coherencia y estándares:** el contenido se debe mostrar de una forma uniforme, con un patrón en la distribución de contenidos y estilos. En nuestra web, el estilo es coherente en todo el contenido, empleando la misma paleta de colores y una distribución de los elementos bastante similar entre todas las páginas.

5. **Prevención de errores:** hay que evitar que el usuario cometa errores, para ello, es mejor guiarlo en el proceso. En este caso se notifica al usuario cuando trata de participar o apuntarse en un evento o foro, respectivamente, y no ha iniciado sesión. Posteriormente, se incorporarán más ayudas visuales en los *inputs*, indicando al final la longitud máxima y la cantidad de caracteres que ya se han escrito por el usuario.

6. **Reconocer frente a memorizar:** el usuario debe reconocer fácilmente los elementos y ofrecerle toda la información que este necesite, sin que deba hacer uso de su memoria. Puede apreciarse en la estructura de los elementos de la web que es siempre muy similar, al igual que en los formularios que tienen el mismo estilo y son muy fáciles de identificar.

7. **Flexibilidad y eficiencia de uso:** es conveniente que los usuarios puedan acceder de forma rápida y sencilla a las tareas que realizan de forma frecuente. En nuestra web, esto se aplica en la página principal, donde es muy fácil encontrar los elementos más relevantes recientemente publicados y permitiendo visualizar los datos con mayor importancia.

8. **Diseño estético y minimalista:** es conveniente que nuestra interfaz mantenga un diseño bonito, además de funcional, sin elementos que entorpezcan a los usuarios. Por ejemplo, toda nuestra web emplea un esquema de colores muy similar, la mayoría de componentes muestran la información mínima y sin estar sobrecargado de elementos que no aportan información relevante, como publicidad.

9. **Ayudar a salir de los errores:** cuando el usuario comete un error es necesario informarle de lo que ha ocurrido, para que lo pueda solventar de una forma sencilla. Para nuestra web, este principio se aplica en los formularios, donde cuando el usuario realiza una acción errónea, se le notifica con el error correspondiente para que pueda corregirlo.

10. **Ayuda y documentación:** es útil ofrecer al usuario documentación y ayuda para poder emplear fácilmente la interfaz. Ahora mismo, en nuestra web, no hay ninguna documentación hacia el usuario y tampoco se considera necesario.

CAPÍTULO 6
Producto Final

En este apartado se va detallar cómo funciona la aplicación web, para ello vamos a hacernos pasar por Laura y realizar los escenarios que planteamos anteriormente en el capítulo de diseño, sección de escenarios. A través de estos escenarios que realizan diferentes actividades, vamos a comprobar cómo luce el producto final.

Para poder publicar contenido nuevo, es necesario tener una cuenta de usuario, por lo que vamos a suponer que este usuario ya está creado y tiene la sesión iniciada.

En el primer escenario, Laura quiere poder vender sus nabos, entonces, accede a Toom Forum, va a la sección de nabos y visualiza todos los anuncios, selecciona uno de los anuncios que ofrece un buen precio, se apunta y posteriormente trata de acceder a la isla. Para lograrlo vamos a ver el procedimiento que realizará:

El primer paso será entrar a la página principal de nuestro proyecto (figura 6.1 y 6.2), observar los anuncios con un precio más alto (figura 6.3).

Figura 6.1: Pantalla principal 1

Figura 6.2: Pantalla principal 2

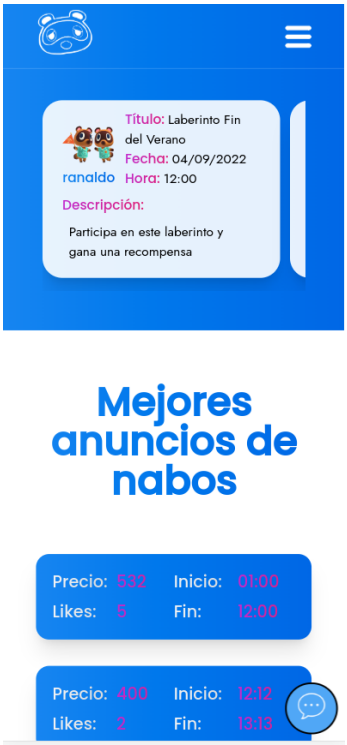


Figura 6.3: Pantalla principal 3

Al entrar al anuncio, visualiza todos los datos (figura 6.6)y se apunta para poder vender sus nabos. También es importante visualizar la pantalla correspondiente a los nabos en sí misma (figura 6.4 y 6.5)

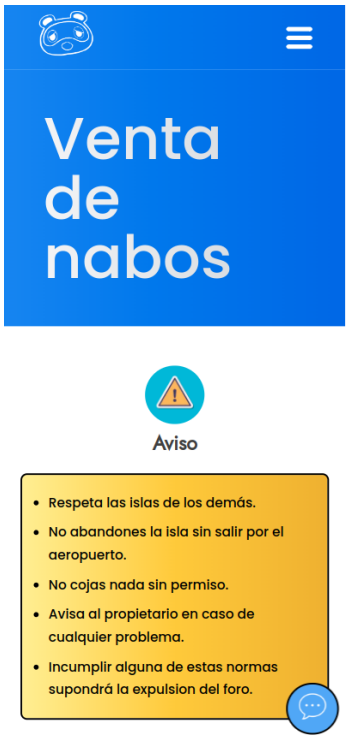


Figura 6.4: Pantalla nabos 1

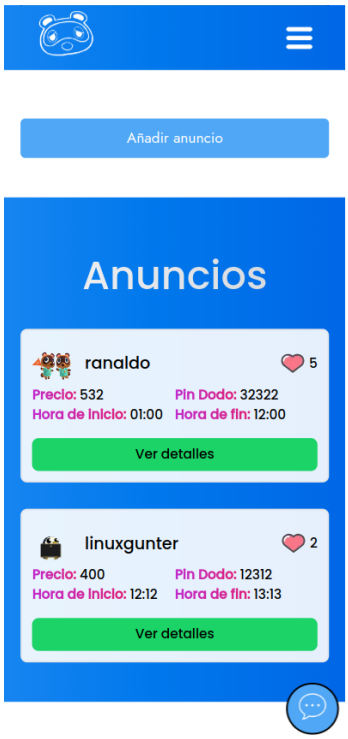


Figura 6.5: Pantalla nabos 2

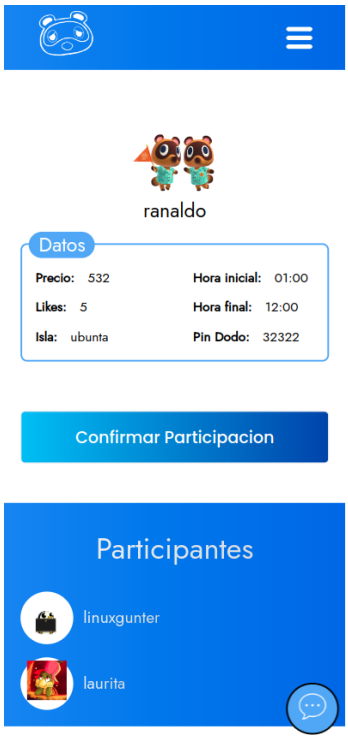


Figura 6.6: Pantalla detalle nabos

El segundo escenario describe la situación en la que Laura quiere crear un evento, para ello accede a Toom Forum, abre el *chat* y pregunta a los usuarios sobre qué clase de evento quieren. Una vez ha obtenido contestación, visita la sección de eventos y lo crea.

Para comenzar, accede a la web y abre el *chat*, donde pregunta a los usuarios y recibe una contestación de estos (figura 6.7).

Ahora, accede al apartado de eventos y abre el formulario, completando toda la información necesaria (figura 6.8 y 6.9). Al crearlo, visualiza el evento que ha creado (figura 6.10).

Figura 6.7: Chat

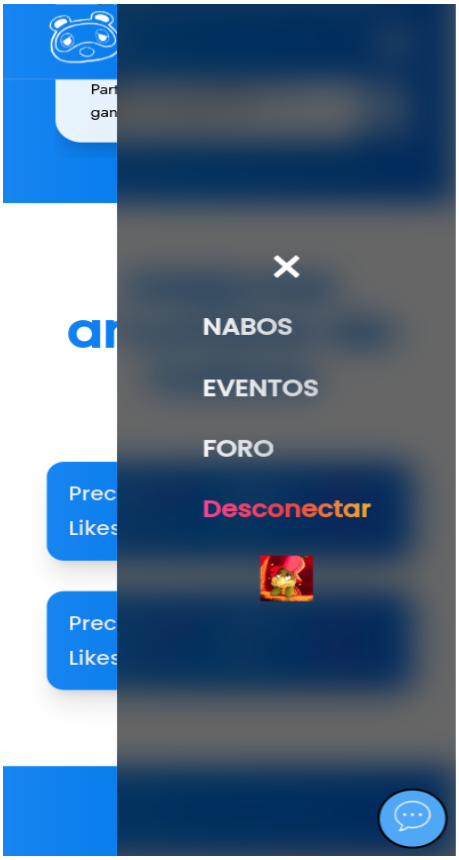


Figura 6.8: Creación Evento 1

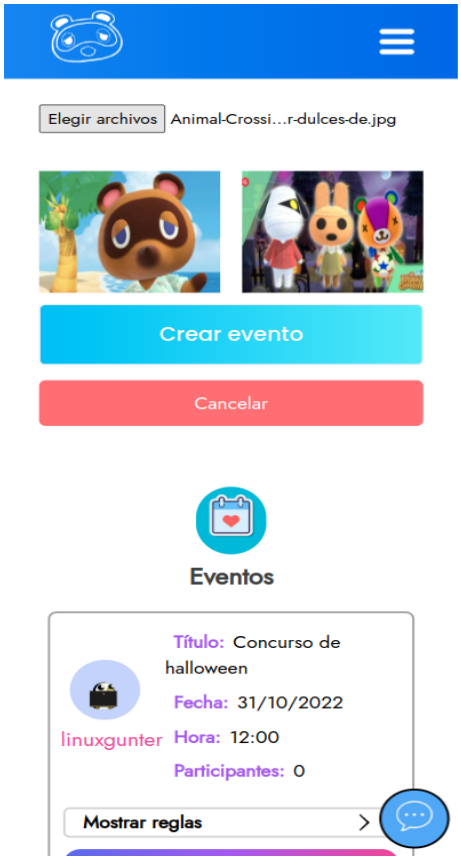


Figura 6.9: Creación Evento 2

Figura 6.10: Evento creado

El tercer escenario relata cómo Laura quiere recibir ayuda sobre cómo funciona el diseño de ropa, por lo tanto, entra al foro y crea un tema indicando su duda y añadiendo una imagen del problema que tiene, más tarde vuelve a observar las respuestas.

Para ello, accede a Toom Forum, accede a la barra de navegación (figura 6.11) y visita la sección del foro (figura 6.12).

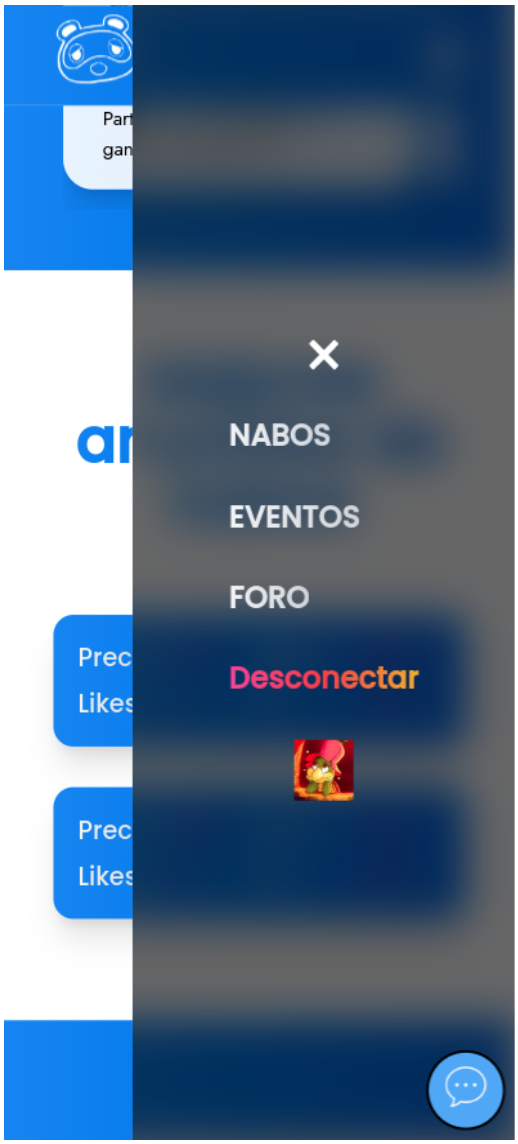


Figura 6.11: Barra de navegación

Figura 6.12: Página principal Foro

Selecciona la categoría de Ayuda (figura 6.13), crea un nuevo Tema explicando la duda que tiene (figura 6.14), más tarde vuelve a observar las respuestas (figura 6.15).

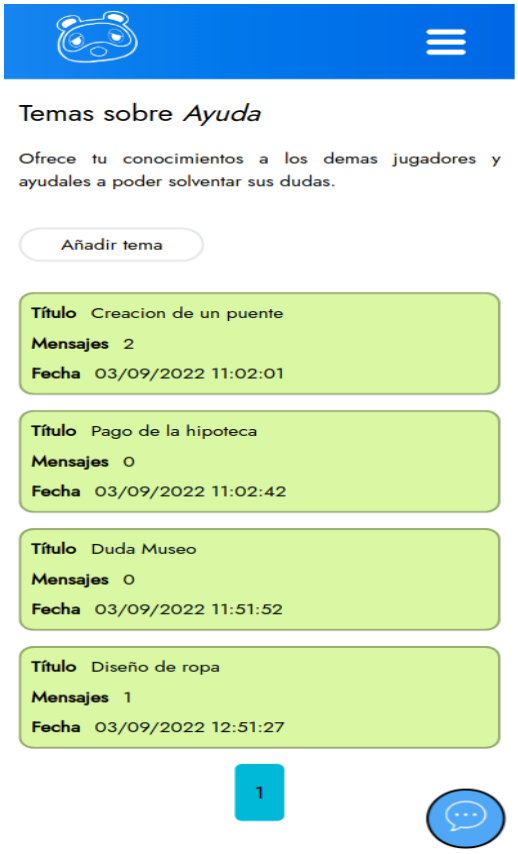


Figura 6.13: Página de Ayuda en el foro

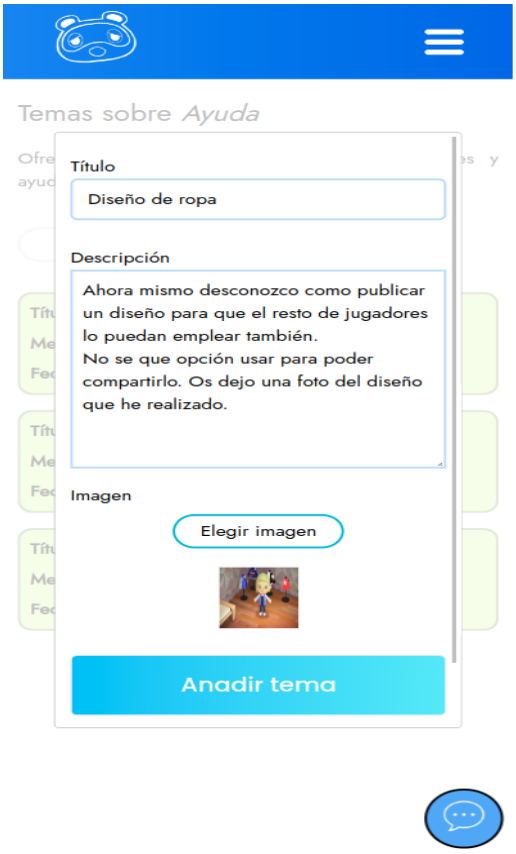


Figura 6.14: Modal Publicar Tema

Figura 6.15: Tema Publicado Diseño

Figura 6.16: Modal publicar tema Vecinos

El cuarto escenario narra cómo Laura quiere un nuevo vecino en su isla. Ella visita el foro, en la sección de vecinos, pregunta por un usuario que tenga ese vecino. Cuando ve las respuesta accede a el perfil, para comprobar los vecinos que tiene el usuario.

Lo primero que realiza es acceder al foro y crear un nuevo tema (figura 6.16). Más tarde vuelve al foro, revisa las respuestas (figura 6.17), accede al perfil de un usuario para comprobar si dispone dicho vecino y obtener sus detalles (figura 6.18).

Figura 6.17: Detalles Tema Vecinos

Figura 6.18: Página de Ayuda en el foro

El último escenario no se considera, ya que es muy similar al tercero.

Recorriendo todos los escenarios se ha podido contemplar en su totalidad la aplicación web de Toom Forum, se ha mostrado la aplicación desde un móvil, esto es debido a que esta orientada a su principalmente en estos dispositivos cómo se ha obtenido en la encuesta.

CAPÍTULO 7
Pruebas

Como se ha mencionado anteriormente en el capítulo que relataba la metodología, se ha empleado la metodología TDD, así que para los componentes más críticos se han desarrollado pruebas software unitarias y *end-to-end*. Otro apartado clave de nuestra aplicación es su usabilidad, entonces debemos emplear herramientas para evaluar si se ofrece un buen producto final.

7.1 Pruebas Software

Anteriormente se mencionó que, para esto, se ha empleado Vitest para las pruebas unitarias y Cypress para las pruebas *end-to-end*. Vamos a explicar rápidamente la implementación de algunas pruebas empleadas para validar nuestro software.

Echemos un vistazo a las pruebas para el componente *Autocomplete*. La implementación se detalla en la figura 7.1. Lo primero es definir una suite de pruebas mediante la palabra clave *describe* que, proporcionándole un nombre, nos permite identificar la *suite* de pruebas. Posteriormente, se definen un conjunto de pruebas, por un lado, estas comprueban que se genera una lista al proporcionarle sugerencias, por otro, comprueban que se llama a una función al *clicar* en un elemento y, por último, si no hay ninguna coincidencia, comprueban que se muestra un texto, indicándole al usuario la falta de sugerencias. La ejecución de las pruebas se detalla en la figura 7.2.

```
describe('Autocomplete list', () => {
  const AutocompleteListProps = {…
  }

  test('Should render a list', () => {
    render(
      <AutocompleteList
        clickSuggestion={AutocompleteListProps.handleClick}
        suggestions={AutocompleteListProps.neighbours}
      />
    )
    expect(screen.getByText('cyrano')).toBeDefined()
  })

  test('On click funcion is called', () => {
    const mock = vi.fn()
    render(
      <AutocompleteList
        clickSuggestion={mock}
        suggestions={AutocompleteListProps.neighbours}
      />
    )

    const li = screen.getAllByRole('listitem')
    fireEvent.click(li[1])

    expect(mock).toHaveBeenCalledTimes(1)
  })

  test('Empty list returns span text', () => {
    render(
      <AutocompleteList
        clickSuggestion={AutocompleteListProps.handleClick}
        suggestions={[]}
      />
    )

    expect(screen.getByText('No hay sugerencias disponibles')).toBeDefined()
  })

  afterEach(() => {
    vi.restoreAllMocks()
  })
})
```

Figura 7.1: Implementación Test Unitario en *Autocomplete*

```
 x alex  ~/dev/Nextjs-TFG  main ±  npm run test --file AutocompleteList.test.tsx

> tfg@0.1.0 test /home/alex/dev/Nextjs-TFG
> vitest "AutocompleteList.test.tsx"


 DEV  v0.18.1 /home/alex/dev/Nextjs-TFG

 ✓ __test__/AutocompleteList.test.tsx (3)

 Test Files  1 passed (1)
      Tests  3 passed (3)
       Time  1.10s (in thread 58ms, 1897.47%)


 PASS  Waiting for file changes...
       press h to show help, press q to quit
```

Figura 7.2: Ejecución de Pruebas *Autocomplete*

Para asegurarnos de que todos los componentes en conjunto funcionan de forma correcta, se han definido pruebas *end-to-end*. Vamos a ver la implementación de una prueba

sencilla donde se va a tratar de iniciar sesión, comprobando los casos en los que el usuario introduce unas credenciales correctas y otro que no. En la figura 7.3 se puede observar cómo la secuencia de pasos es sencilla, en este caso, se han indicado unas aserciones seleccionando elementos de la interfaz mediante los selectores empleados en CSS.

```
describe('Login', () => {
  beforeEach(() => {
    cy.visit('http://localhost:3000/signin')
  })

  it('user can login', () => {
    cy.get('h2').contains('Iniciar sesion')
    cy.get('input:first').type('ranaldo')
    cy.get('input:last').type('123123')
    cy.get('button').click()
    cy.url().should('eq', 'http://localhost:3000/')
  })

  it('user doesnt exists', () => {
    cy.get('input:first').type('ranaldononon')
    cy.get('input:last').type('123123')
    cy.get('button').click()
    cy.get('.alert').contains('Usuario o contrasena incorrectos')
  })
})
```

Figura 7.3: Implementación Prueba *end-to-end* Inicio de Sesión

La ejecución de estas pruebas se realiza a través de una interfaz gráfica, que genera un navegador y va ejecutando los pasos descritos de una forma visual. Este proceso no se puede mostrar en el documento, así que se va a mostrar el resultado final de la ejecución de esta prueba, como se observa en la figura 7.4.

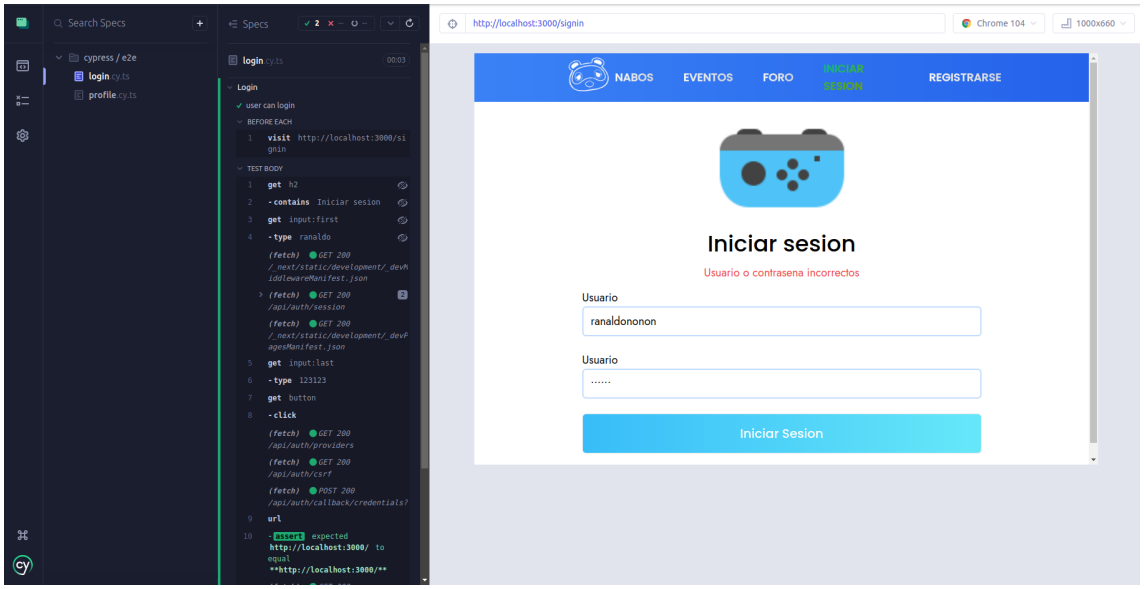


Figura 7.4: Ejecución de Prueba *end-to-end* Inicio de Sesión

7.2 Pruebas Usuario

Una vez finalizado el desarrollo, es importante comprobar que los jugadores pueden emplear la web de forma sencilla, ofreciendo una buena UX y que están conformes con la aplicación creada.

Para poder evaluar la satisfacción de los usuarios se ha empleado otro cuestionario, se puede obtener el detalle en el apéndice C, este se divide en dos partes. Cabe mencionar que, antes de responder al cuestionario, los jugadores han estado navegando durante un tiempo. La primera parte del cuestionario consta de preguntas relacionadas con la página web en general, la segunda parte describe una tarea que deben realizar los usuarios, esta tarea está formada por una serie de pasos: acceder al foro, seleccionar una categoría, crear un tema, un mensaje y una respuesta a este mensaje. En esta sección los usuarios deben evaluar la dificultad de esta tarea.

En este cuestionario han respondido un total de seis personas, donde el cincuenta por ciento ha accedido desde su *smartphone*, el treinta y cuatro por ciento desde su *portátil* y el resto desde su ordenador de sobremesa.

Se pregunta por diferentes aspectos sobre la web, como si consideran que la web está sobrecargada de elementos, el tiempo de respuesta, si se muestra todo el contenido necesario o el nivel tecnológico necesario para poder emplear la web. Todas estas preguntas estaban calificadas con una escala del uno al cinco. Referente a la pregunta de si consideran que la web esta sobrecargada, el sesenta y siete por ciento considera que está poco cargada; sobre el tiempo de respuesta, el sesenta y siete por ciento creen que la web es rápida; también consideran el sesenta y siete por ciento que la web ofrece un contenido relevante y que una persona con escasas habilidades tecnológicas puede emplear el sitio web. Estos resultados se pueden observar en las gráficas adjuntadas en el anexo C.

Respecto a la prueba guiada, la mayoría la pudo completar sin ningún problema. El sesenta y siete por ciento consideran que es sencillo realizar la tarea y que la estructura del foro es la correcta. Mientras que, el cincuenta por ciento, creen que es muy difícil cometer errores y, el treinta y tres por ciento, que es difícil.

CAPÍTULO 8
Conclusiones

Al inicio de este documento se enumeraron todos los objetivos que se pretendían cumplir. El principal era ofrecer a los usuarios las funcionalidades que se encontraban segregadas en diferentes puntos de encuentro en un único lugar, ofreciendo una experiencia de usuario superior al resto de alternativas. Llegados a este punto, podemos confirmar que este objetivo principal se ha cumplido satisfactoriamente.

Los puntos clave que nos han llevado al éxito, son los siguientes:

- Ofrecer toda la funcionalidad que ofrecen otros puntos de encuentro de manera segregada, de una forma más automatizada y simple, permitiendo a los usuarios ahorrar tiempo y no tener que manejar estas tareas de forma manual.
- Permitir que los usuarios puedan mantener interacciones mediante un *chat* y foro.
- Se ha creado una interfaz de usuario accesible, donde se ha empleado un HTML semántico adecuado para que usuarios con discapacidad no tengan problemas al navegar por nuestro sitio web.
- No se muestra publicidad al usuario, además de que su contraseña se encuentra cifrada en la base de datos para brindar una mayor seguridad.
- La web ofrece un tiempo de respuesta rápido, gracias al uso de técnicas como *Server side rendering*, el manejo de imágenes que proporciona *Next.js* y el empleo de las *Edge Functions*.

Realizar este proyecto ha sido un desafío, ya que el uso de tecnologías tan nuevas ha requerido de un gran esfuerzo y dedicación, donde ha destacado el proceso iterativo de mejora sobre elementos ya elaborados.

8.1 Relación del trabajo desarrollado con los estudios cursados

Todo el proyecto se basa en el desarrollo de una aplicación web y en ofrecer una buena UX, que son temáticas muy afines al desarrollo web. Por esta razón, las asignaturas que mayor relación tienen están orientadas al desarrollo web y la creación de interfaces.

Debido a estos factores relacionados con el desarrollo, la asignatura que guarda una mayor relación es **Desarrollo Web**. En ella se introduce al mundo del desarrollo web explicando las bases de este, como lo son las tecnologías base que lo soportan (HTML, CSS y JavaScript), cómo funciona el DOM y una introducción al lado del servidor.

Otra asignatura con especial relevancia es **Integración de Aplicaciones**. Gracias a ella se ha comprendido mejor cómo funcionan los servicios web y cómo intercambiar datos entre diferentes productos *software*. La asignatura de **Tecnologías de Sistemas de Información en la Red** también ha resultado útil, no tanto por su temario, si no por el conocimiento que otorga sobre JavaScript.

En relación a la interfaz de usuario, la asignatura de **Interfaces Persona Computador** ofrece conceptos muy importantes de la creación de interfaces y hace que el alumnado tome una mayor conciencia de lo importante que resulta esta parte, que ha sido un punto muy relevante en este proyecto.

Referente a la creación de interfaces y ofrecer la mejor experiencia de uso, está **Desarrollo Centrado en el Usuario**. En esta asignatura se explica con mucho detalle lo importante que es la experiencia de usuario y la accesibilidad que debe presentar una interfaz. Además presenta una serie de métodos a emplear para obtener la mejor experiencia de usuario, que han sido empleados en este proyecto.

Por último, **Bases de Datos y Sistemas de Información** ha servido para comprender mejor cómo crear el modelo empleado en la aplicación para la persistencia, ya que en ella se detalla cómo funciona SQL y cómo crear esquemas y modelos.

Ahora bien, en relación a las competencias transversales adquiridas a lo largo de estos años en el grado, considero que, en este proyecto, las que mayor relevancia tienen son las siguientes:

- **Diseño y proyecto:** Me ha permitido poder gestionar mejor las diferentes fases por las que ha transcurrido el proyecto, haciendo más sencillo el proceso y poder obtener un buen resultado final.
- **Aprendizaje permanente:** A lo largo de estos años he podido adquirir un gran número de conocimientos, pero todos ellos no bastaban para poder realizar este proyecto, por lo tanto, he tenido que realizar diferentes cursos y mucha práctica antes de poder comenzarlo.
- **Instrumental específica:** Gracias a esta competencia he podido escoger las herramientas necesarias para poder ofrecer una solución adecuada, donde he buscado emplear las tecnologías web más demandadas del mercado.
- **Análisis y resolución de problemas:** Además de ayudarme a poder analizar las necesidades de los usuarios y ofrecer una solución, han surgido otros problemas a lo largo del desarrollo, como puede ser la adopción de nuevas tecnologías para poder ofrecer la mayor experiencia.
- **Compresión e integración:** Me ha proporcionado las herramientas para sintetizar todas las fases del proyecto en este documento.

CAPÍTULO 9
Trabajo Futuro

El punto en el que se encuentra actualmente la aplicación es totalmente funcional y cumple con todos los objetivos planteados al inicio de este proyecto. Pero, aun así, se encuentra lejos de estar completamente terminada, ya que se le pueden añadir funcionalidades extra.

La principal funcionalidad que se contempla añadir en un futuro son las notificaciones. Actualmente, hay infinidad de sitios web que ofrecen notificaciones a los usuarios y para este proyecto se cree conveniente esta adición. Esto permitiría a los usuarios no tener que estar pendientes de cuándo pueden acceder a una isla para vender nabos o de publicaciones sobre una categoría en la que están interesados.

Otra parte que facilitaría la comunicación entre usuarios sería la incorporación de mensajes directos entre jugadores. En la actualidad, únicamente se ofrece un *chat* grupal que puede dificultar que un usuario establezca la comunicación privada con otro jugador.

Ahora mismo, no se ofrece información muy detallada sobre el juego que podría ayudar a los usuarios a encontrar datos relevantes sin necesidad de recurrir a diferentes páginas para obtenerlos. Entonces, la adición de una guía con los principales puntos del juego ayudaría a los jugadores a obtener una mejor experiencia dentro del juego y únicamente tener que recurrir a una web para obtener la información que requieran.

En relación con el párrafo anterior, tenemos guardada mucha información relevante de los vecinos que no se muestran al usuario. Sería conveniente mostrar esta información cuando un usuario está visitando perfiles de otros jugadores para buscar vecinos.

Un mecanismo que resultaría útil, consistiría en incorporar un botón para poder compartir anuncios, eventos o mensajes de una forma muy simple.

Todas estas mejoras funcionales están muy bien, pero no hay que dejar de lado el mantenimiento de nuestro código. Es una tarea fundamental, ya que pueden surgir problemas que se deban corregir. También es importante modificar el código existente y mejorarlo, permitiendo que las tareas de mantenimiento sean más sencillas y sea más legible por otros usuarios que quieran incorporar modificaciones.

Bibliografía

[1] La página oficial para Animal Crossing. En línea. *La página oficial para Animal Crossing - Inicio*. [s. f.]. Disponible en: https://animal-crossing.com/es/. [consultado el 18/08/2022].

[2] COLABORADORES DE LOS PROYECTOS WIKIMEDIA. *Telegram - Wikipedia, la enciclopedia libre.* En línea. Wikipedia, la enciclopedia libre. 27/01/2014. Disponible en: https://core.telegram.org/. [consultado el 19/08/2022]

[3] LOWDERMILK, Travis. *User-Centered Design: A Developer's Guide to Building User-Friendly Applications*. O'Reilly Media, 2013. ISBN 9781449359805.

[4] BECK, Kent. *Test Driven Development: By Example (The Addison-Wesley Signature Series)*. Addison-Wesley Professional, 2002. ISBN 9780321146533.

[5] *A beginner's guide to CI/CD and automation on GitHub | The GitHub Blog*. En línea. The GitHub Blog. [s. f.]. Disponible en: https://github.blog/2022-06-03-a-beginners-guide-to-ci-cd-and-automation-on-github/. [consultado el 05/06/2022].

[6] LIM, Winnie. *Guía para principiantes sobre cómo hacer un wireframe*. En línea. Web Design Envato Tuts+. 16/09/2020. Disponible en: https://webdesign.tutsplus.com/es/articles/a-beginners-guide-to-wireframing--webdesign-7399. [consultado el 12/07/2022].

[7] VERCEL. *Introduction to vercel*. En línea. Vercel Documentation. [s. f.]. Disponible en: https://vercel.com/docs. [consultado el 4/07/2022].

[8] *Uizard | Design Wireframes and Mockups In Minutes | Uizard*. En línea. Uizard | Design Wireframes and Mockups In Minutes | Uizard. [s. f.]. Disponible en: https://uizard.io/es/. [consultado el 16/06/2022].

[9] *Qué es Mobile First y por qué es importante | Quality Devs*. En línea. Quality Devs. [s. f.]. Disponible en: https://www.qualitydevs.com/2019/02/07/que-es-mobile-first-y-por-que-es-importante/. [consultado el 16/06/2022].

[10] *Sistema de Escalas de Usabilidad: ¿qué es y para qué sirve? | | UXpañol*. En línea. UXpañol. [s. f.]. Disponible en: https://uxpanol.com/teoria/sistema-de-escalas-de-usabilidad-que-es-y-para-que-sirve/. [consultado el 04/07/2022].

[11] *¿Qué es edge computing?* En línea. IBM - Deutschland | IBM. [s. f.]. Disponible en: https://www.ibm.com/es-es/cloud/what-is-edge-computing. [consultado el 03/07/2022].

[12] *Acnh api. En línea*. ACNH API. [s. f.]. Disponible en: http://acnhapi.com/. [consultado el 24/07/2022].

[13] HTML: *Lenguaje de etiquetas de hipertexto* | MDN. En línea. MDN Web Docs. [s. f.]. Disponible en: https://developer.mozilla.org/es/docs/Web/HTML. [consultado el 02/07/2022].

[14] *CSS* | MDN. En línea. MDN Web Docs. [s. f.]. Disponible en: https://developer.mozilla.org/es/docs/Web/CSS. [consultado el 02/07/2022].

[15] *JavaScript* | MDN. En línea. MDN Web Docs. [s. f.]. Disponible en: https://developer.mozilla.org/es/docs/Web/JavaScript. [consultado el 02/07/2022].

[16] *JavaScript with syntax for types*. En línea. TypeScript: JavaScript With Syntax For Types. [s. f.]. Disponible en: https://www.typescriptlang.org/. [consultado el 03/07/2022].

[17] MICROSOFT. *Visual studio code - code editing. redefined*. En línea. Visual Studio Code - Code Editing. Redefined. 03/11/2021. Disponible en: https://code.visualstudio.com/. [consultado el 01/07/2022].

[18] Tailwind CSS - *Rapidly build modern websites without ever leaving your HTML*. En línea. Tailwind CSS - Rapidly build modern websites without ever leaving your HTML. [s. f.]. Disponible en: https://tailwindcss.com/. [consultado el 08/07/2022].

[19] *Find and fix problems in your JavaScript code - ESLint - Pluggable JavaScript Linter*. En línea. Find and fix problems in your JavaScript code - ESLint - Pluggable JavaScript Linter. [s. f.]. Disponible en: https://eslint.org/. [consultado el 05/07/2022].

[20] *Home - docker*. En línea. Docker. [s. f.]. Disponible en: https://www.docker.com/. [consultado el 16/07/2022].

[21] *React – Una biblioteca de JavaScript para construir interfaces de usuario*. En línea. React – Una biblioteca de JavaScript para construir interfaces de usuario. [s. f.]. Disponible en: https://es.reactjs.org/. [consultado el 04/07/2022].

[22] *Getting started* | next.js. En línea. Next.js by Vercel - The React Framework. [s. f.]. Disponible en: https://nextjs.org/docs. [consultado el 05/07/2022].

[23] BRYANT, Alejandro. *Server side rendering y las pwas*. En línea. PWA Experts I/O | Progressive Web Applications | PWA Experts I/O. 16/05/2020. Disponible en: https://pwaexperts.io/blog/server-side-rendering-pwa. [consultado el 05/07/2022].

[24] *Node.js*. En línea. Node.js. [s. f.]. Disponible en: https://nodejs.org/es/. [consultado el 12/07/2022].

[25] *PostgreSQL*. En línea. PostgreSQL. [s. f.]. Disponible en: https://www.postgresql.org/. [consultado el 06/07/2022].

[26] *Prisma*. En línea. Prisma. [s. f.]. Disponible en: https://www.prisma.io/. [consultado el 06/07/2022].

APÉNDICE A
OBJETIVOS DE DESARROLLO SOSTENIBLE

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenible	**Alto**	**Medio**	**Bajo**	**No procede**
ODS 1. **Fin de la pobreza.**				X
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**				X
ODS 4. **Educación de calidad.**				X
ODS 5. **Igualdad de género.**			X	
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**				X
ODS 9. **Industria, innovación e infraestructuras.**			X	
ODS 10. **Reducción de las desigualdades.**		X		
ODS 11. **Ciudades y comunidades sostenibles.**			X	
ODS 12. **Producción y consumo responsables.**				X
ODS 13. **Acción por el clima.**				X
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**				X
ODS 17. **Alianzas para lograr objetivos.**				X

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

El 25 de septiembre de 2015, los líderes mundiales decidieron adoptar un conjunto de objetivos globales para erradicar la pobreza, proteger el planeta y asegurar la prosperidad para todos, como parte de una nueva agenda de desarrollo sostenible. Cada objetivo tiene metas específicas que deben alcanzarse en los próximos quince años.

En el verano correspondiente al año 2022, en el mes de julio, se alcanzaron las temperaturas más altas en España desde que hay registros. Es una situación preocupante, además cada año esta situación irá a peor. Por lo tanto, es importante reaccionar y tratar de frenar este rápido avance. A continuación, se detalla la relación de estos objetivos con el proyecto.

Como se puede observar en la tabla superior, nuestro proyecto no se relaciona directamente con la mayoría de los objetivos de desarrollo sostenible planteados en ella. Esto se debe a que se trata de una aplicación web destinada a una comunidad concreta de jugadores, el propósito de esta aplicación es brindar a la comunidad de un punto de encuentro donde puedan cooperar entre sí para fomentar la interacción entre los jugadores, así como ofrecer de diversas opciones para que puedan comerciar con sus nabos o crear eventos. Debido a esto, es difícil que el proyecto cumpla con todos los objetivos de desarrollo sostenible, a excepción de uno de ellos.

El objetivo con el que guarda una mayor relación el proyecto es el de **redución de las desigualdades**. Esto es debido a que el proyecto ha sido desarrollado teniendo en cuenta a los posibles usuarios con discapacidad. Además, nuestra apliación web se ofrece de manera gratuita a todo el mundo. Con todo ello, se intenta reducir al máximo las desigualdades que se puedan dar.

También podemos encontrar un nivel más bajo de relación con otros tres objetivos. Por un lado, con el de **igualdad de genero**, ya que en todo momento se busca la inclusión y la equidad entre mujeres y hombres de la comunidad. Además, como se puede observar en los resultados de la investigación la mayoría de personas que respondieron fueron mujeres, rompiendo así con el estigma de que los videojuegos son para hombres. Por otro lado, nuestro proyecto tiene cierta relación con el objetivo que hace referencia a la **industria, innovación e infraestructuras**, por el hecho de crear un punto de encuentro diferente a los que existen actualmente, tratando de otorgar un valor diferenciador, ofreciendo un producto fácil de usar y moderno. Y, por último, con el de **ciudades y comunidades sostenibles**, ya que tratamos de crear una comunidad de jugadores unida que busca un objetivo común, donde poder conocer a gente de distintos lugares y formar amistades.

Incluso se podría llegar a relacionar con los que estén relacionados con la naturaleza, debido a que el videojuego Animal Crossing New Horizons promueve unos valores sobre cómo debemos cuidar el planeta y el mundo animal. Es por esto, que podría guardar relación con los siguientes:

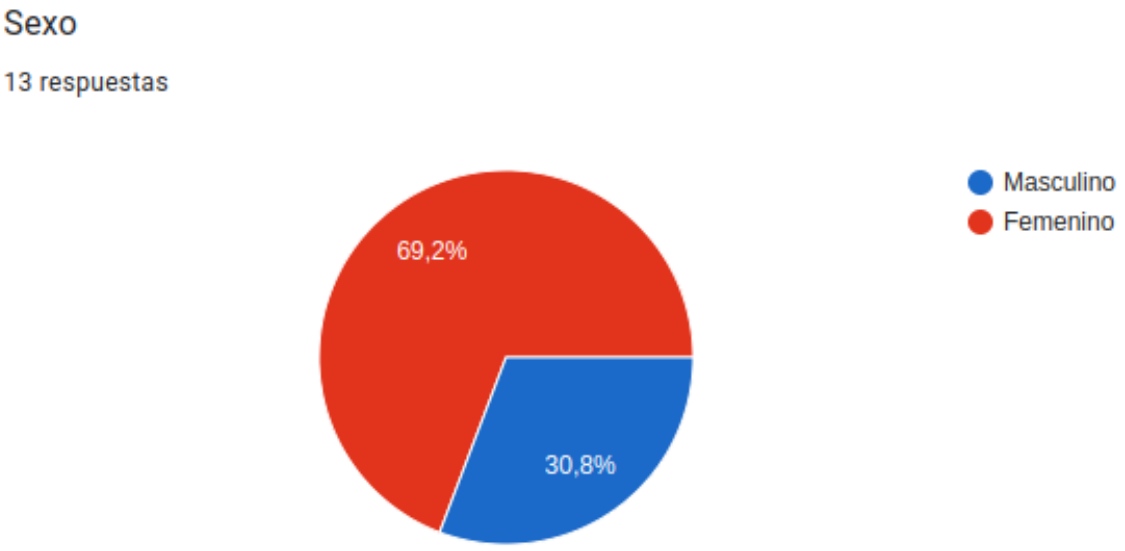
- **Acción por el clima:** Cada año que transcurre, la temperatura de nuestro planeta aumenta, debido a esto se ponen en riesgo nuestras vidas y a la naturaleza. En general, Animal Crossing promueve la lucha contra el cambio climático, al ser un juego que puede ser jugado por todos los públicos, a los niños les otorga valores como el reciclaje o el cuidado de la naturaleza. A parte de esto, Vercel, el proveedor que alberga nuestra aplicación web, emplea la red de AWS de Amazon, esta red de *cloud* es la más empleada y la que ofrece una mayor eficiencia energética, para el año 2025 tienen planeado que el cien por cien de su consumo energético provenga de fuentes renovables.

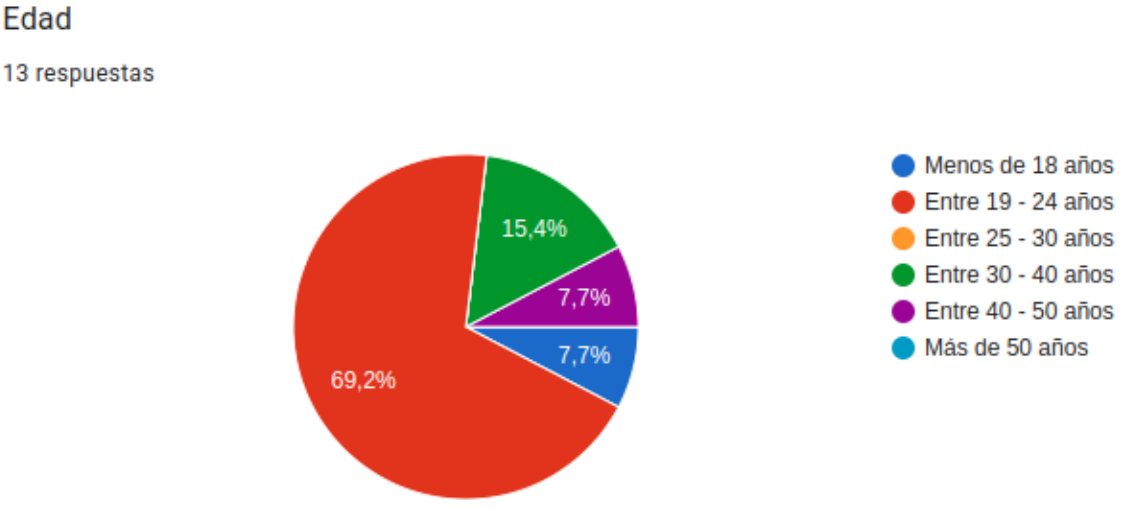
- **Vida submarina y Vida de ecosistemas terrestres:** En Animal Crossing todos los habitantes son animales, en él se puede descubrir la gran variedad de especies que habitan nuestro planeta.

Por otro lado, relacionados con la propia finalidad del foro, se podría llegar a relacionar con **Alianzas para lograr objetivos**, ya que al fin y al cabo los jugadores deben cooperar para cualquier actividad que se desarrolla en la página, ya sea para la compraventa de nabos o ayudar al resto de jugadores. Debido a esto, es crucial que los jugadores cooperen entre sí para lograr cumplir con sus objetivos.

APÉNDICE B
Cuestionario Investigación Cualitativa

En este Anexo se muestran todas las figuras referentes a los resultados obtenidos del cuestionario entregado a los jugadores presentes en un canal de Telegram de ANHC, donde la intención es obtener más información sobre ellos.

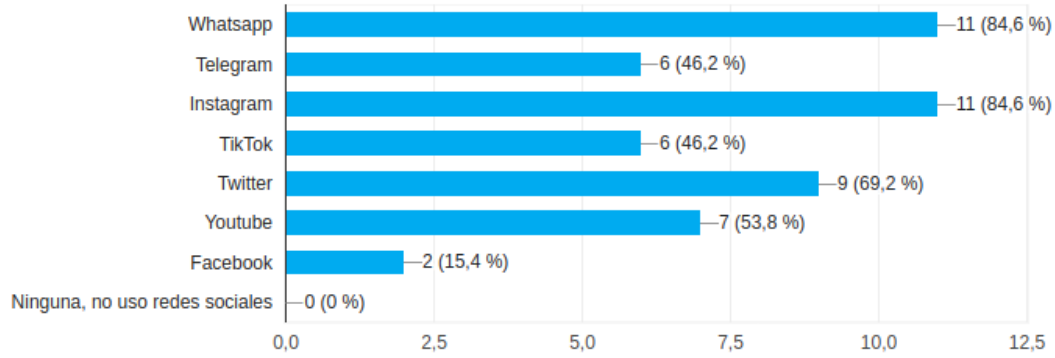




¿Qué redes sociales empleas con más frecuencia?

Copiar

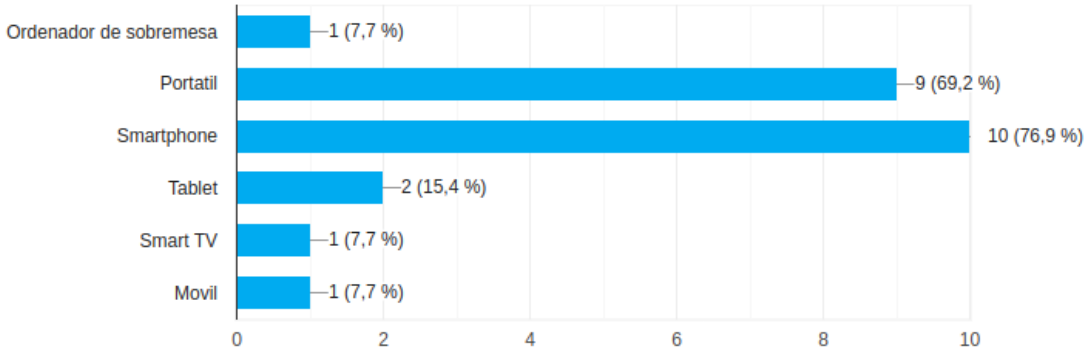
13 respuestas



¿Desde que dispositivos sueles acceder a los sitios web?

Copiar

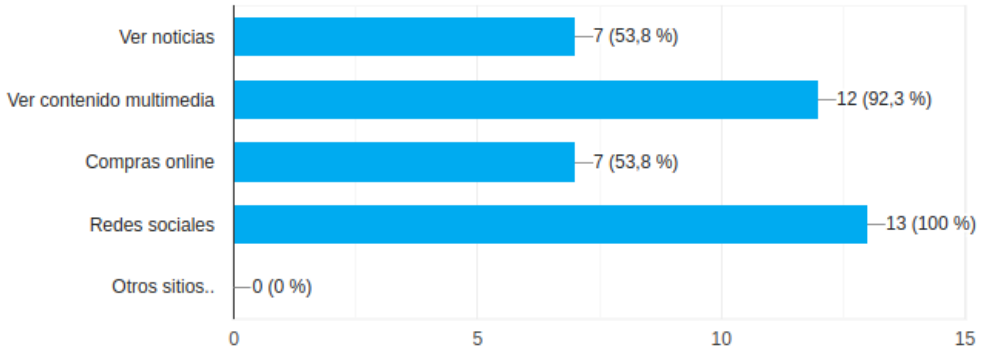
13 respuestas



¿Qué realizas cuando navegas por Internet?

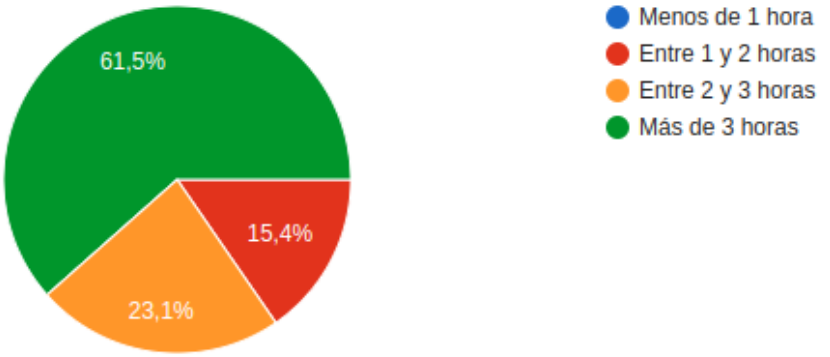
Copiar

13 respuestas



¿Cuánto tiempo sueles dedicar a navegar por Internet al día?

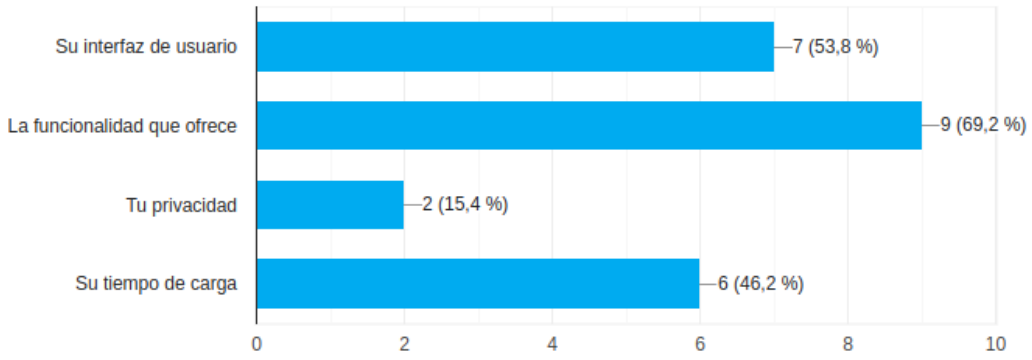
13 respuestas



Cuando visitas un sitio web ¿Qué es lo que más valoras?

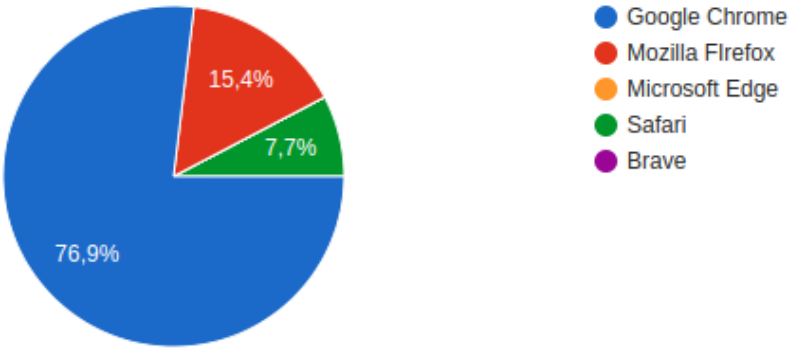
Copiar

13 respuestas



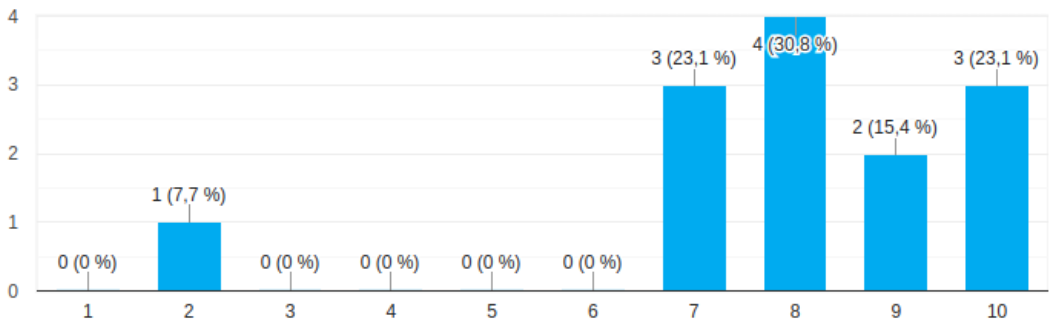
¿Qué navegador web usas con más frecuencia?

13 respuestas



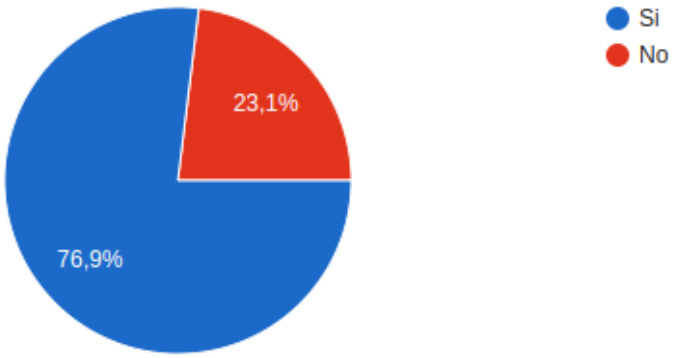
¿Cómo calificarías tus habilidades tecnológicas a la hora de navegar por internet? Copiar

13 respuestas



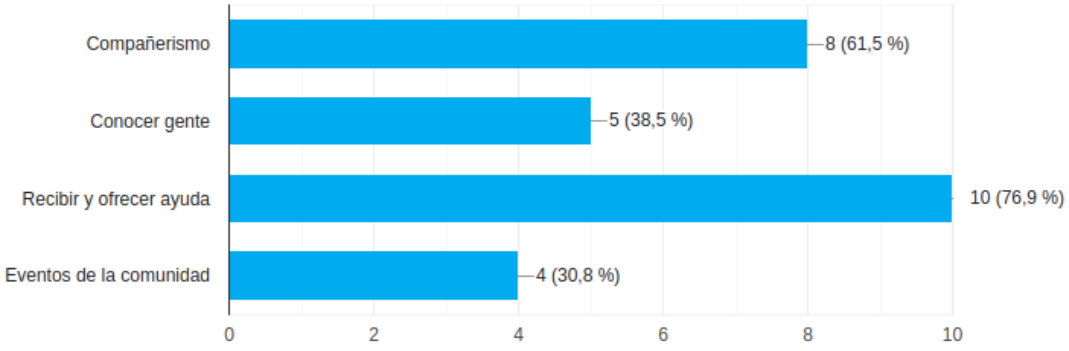
¿Perteneces a alguna comunidad de Animal Crossing o de otros juegos?

13 respuestas



¿Qué es lo que más valoras de una comunidad? Copiar

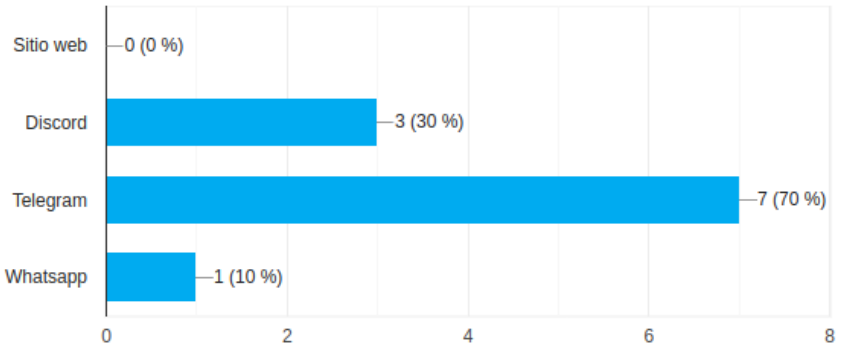
13 respuestas



Si perteneces a alguna comunidad ¿Dónde reside el punto de encuentro de la comunidad?

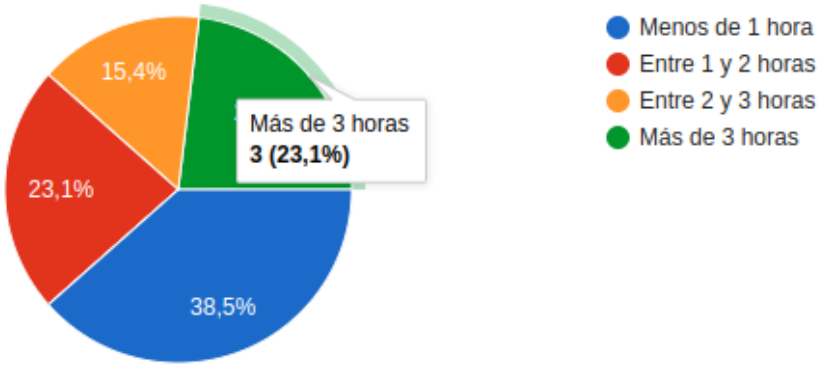
Copiar

10 respuestas



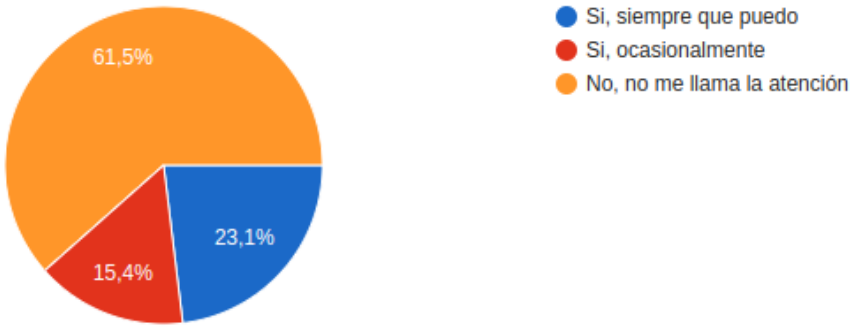
¿Cuántas horas sueles jugar a Animal Crossing New Horizons a lo largo de una semana?

13 respuestas



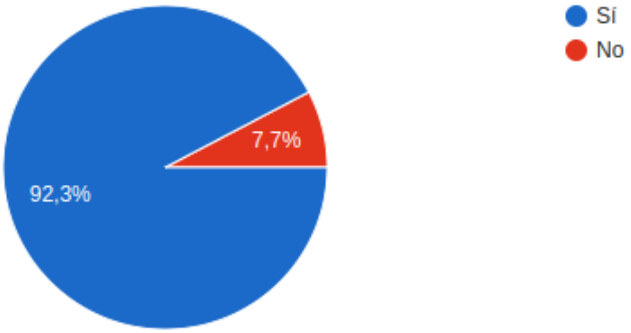
¿Sueles participar en concursos o eventos de Animal Crossing o otros juegos?

13 respuestas



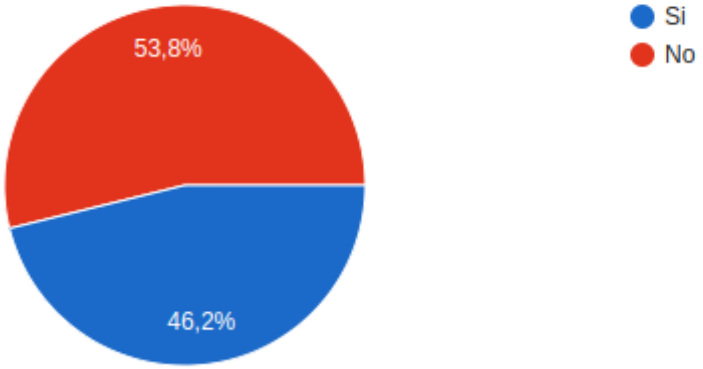
Cuando tienes alguna duda o problema, ¿Acudes a alguna comunidad para tratar de resolverlo?

13 respuestas



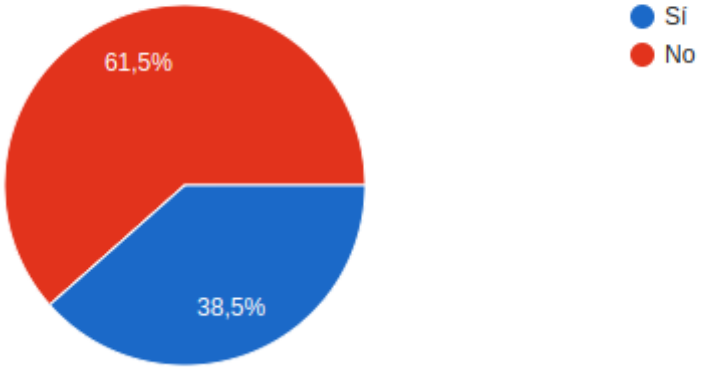
¿Sueles emplear algún punto de encuentro para comprar/vender nabos?

13 respuestas



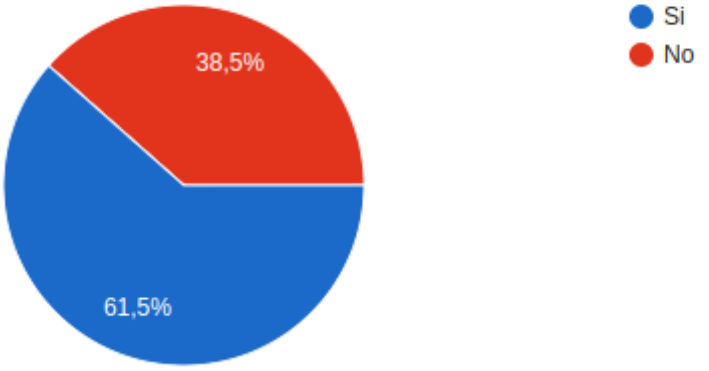
¿Acudes a islas de otros jugadores de manera frecuente?

13 respuestas



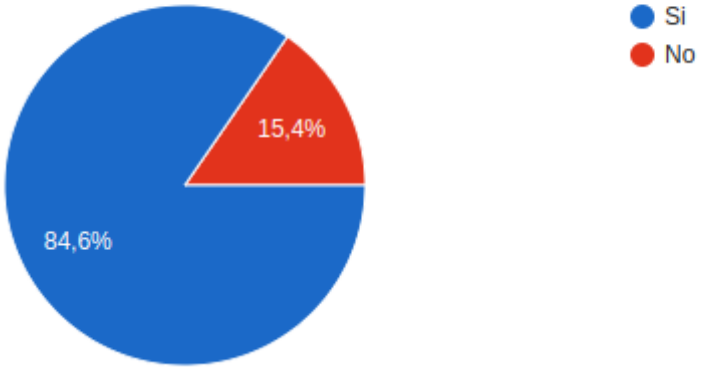
¿Visitas otras islas para catalogar objetos que no posees?

13 respuestas

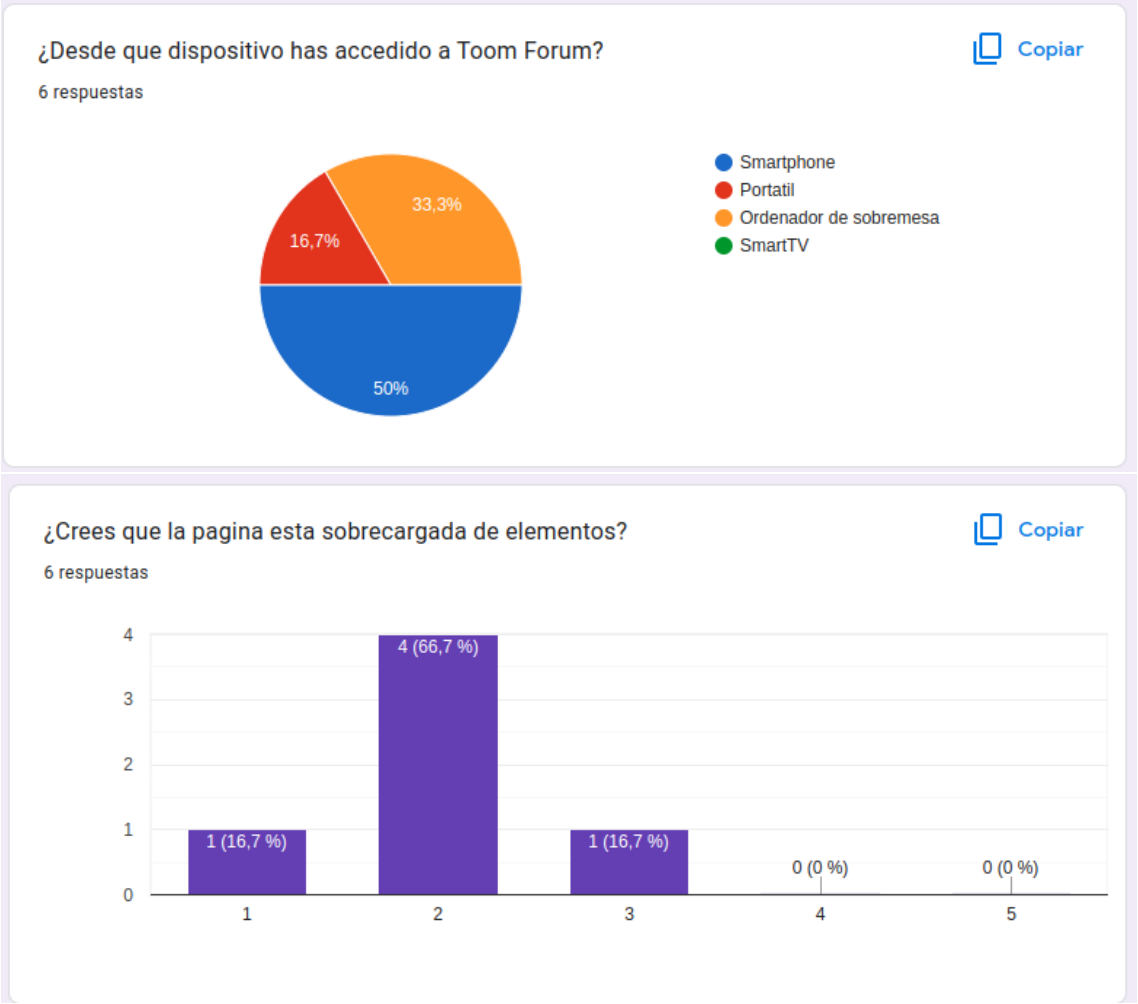


¿Acudes a otras islas para obtener frutas, flores, etc... que no poseas?

13 respuestas



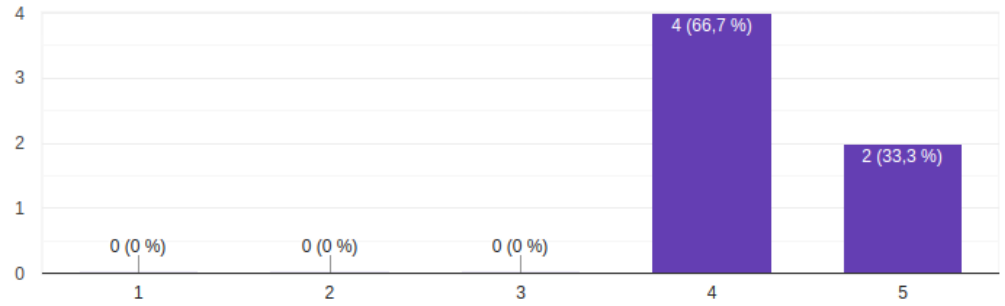
APÉNDICE C
Cuestionario Pruebas Usuarios



¿Consideras que la velocidad de la web es adecuada?

Copiar

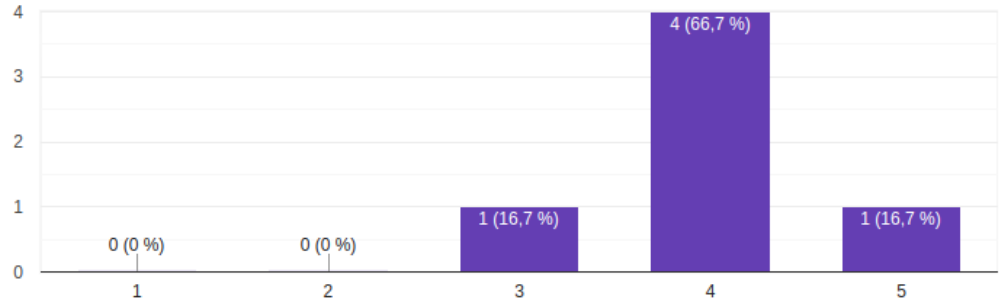
6 respuestas



¿Consideras que se muestra toda la información necesaria?

Copiar

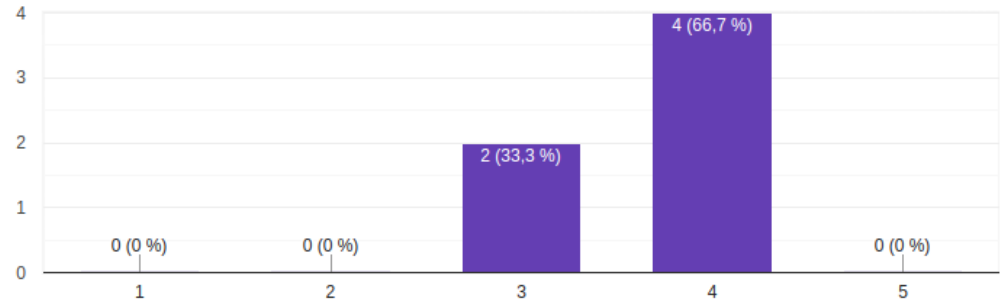
6 respuestas



¿Crees que cualquier persona puede emplear esta aplicación, independientemente de su nivel de conocimiento?

Copiar

6 respuestas



C.0.1. Prueba guiada