UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

# Portal inmobiliario dirigido a la comunidad universitaria

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Akhmedov, Marat

Tutor/a: Albert Albiol, Manuela

Cotutor/a: Torres Bosch, María Victoria

CURSO ACADÉMICO: 2023/2024

# Resumen ejecutivo

## Castellano

Este Trabajo Fin de Grado (TFG) propone el desarrollo de una plataforma web con las ofertas de alquiler dirigidas específicamente a estudiantes y personal universitario y que facilita la búsqueda de alojamiento adecuado y asequible.

Para abordarlo, se realiza un estudio de mercado para identificar las necesidades de la comunidad universitaria en términos de vivienda. Además, se implementó un sistema una plataforma web que permite publicar, buscar y gestionar anuncios de alquiler y venta de inmuebles, integrando funcionalidades de búsqueda con filtros específicos con una interfaz de usuario adaptable a dispositivos móviles y de escritorio

La creación del sitio web se lleva a cabo empleando herramientas como Symfony 7, basado en PHP 8.2, para la gestión de bases de datos se utiliza MySql. Para el frontend, se utiliza Twig, Bootstrap 5, y jQuery. La gestión del proyecto se realiza con Git utilizando Visual Studio Code como editor de código.

La plataforma permite a los usuarios ahorrar tiempo y recursos al ofrecer el alquiler para la comunidad universitaria en un solo lugar. Proporciona una mayor transparencia en el mercado inmobiliario y fomenta la competitividad entre las ofertas, contribuyendo a la moderación de los precios. Además, mejora la calidad de vida de los estudiantes y personal universitario al facilitar el acceso a información relevante y simplificar el proceso de búsqueda de vivienda.

En conclusión, la plataforma propuesta ofrece una solución práctica y eficiente a un problema real, mejorando significativamente la experiencia de búsqueda de vivienda para la comunidad universitaria. Se prevé que la plataforma contribuirá de manera favorable a la calidad de vida de quienes la utilicen.

## Valenciano

Este Treball fi de grau (*TFG) proposa el desenrotllament d'una plataforma web amb les ofertes de lloguer dirigides específicament a estudiants i personal universitari i que facilita la busca d'allotjament adequat i assequible.

Per a abordarho, es realitza un estudi de mercat per a identificar les necessitats de la comunitat universitària en termes de vivenda. A més, es va implementar un sistema una plataforma web que permet publicar, buscar i gestionar anuncis de lloguer i venda d'immobles, integrant funcionalitats de busca amb filtres específics amb una interfície d'usuari adaptable a dispositius mòbils i d'escriptori

El creació del lloc web es duu a terme emprant ferramentes com Symfony 7, basat en PHP 8.2, per a la gestió de bases de dades s'utilitza MySql. Per al frontend, s'utilitza Twig, Bootstrap 5, i jQuery. La gestió del projecte es realitza amb Git utilitzant Visual Studio Code com a editor de codi.

La plataforma permet als usuaris estalviar temps i recursos en oferir el lloguer per a la comunitat universitària en un sol lloc. Proporciona una major transparència en el mercat immobiliari i fomenta la competitivitat entre les ofertes, contribuint a la moderació dels preus. A més, millora la qualitat de vida dels estudiants i personal universitari en facilitar l'accés a informació rellevant i simplificar el procés de busca de vivenda.

En conclusió, la plataforma proposada oferix una solució pràctica i eficient a un problema real, millorant significativament l'experiència de busca de vivenda per a la comunitat universitària. Es preveu que la plataforma contribuirà de manera favorable a la qualitat de vida dels qui la utilitzen.

## Inglés

This Final Degree Project (TFG) proposes the development of a web platform with rental offers aimed specifically at students and university staff and that facilitates the search for suitable and affordable accommodation.

To address this, a market study is carried out to identify the needs of the university community in terms of housing. In addition, a web platform system

was implemented that allows publishing, searching and managing rental and sale property advertisements, integrating search functionalities with specific filters with a user interface adaptable to mobile and desktop devices.

The creation of the website is carried out using tools such as Symfony 7, based on PHP 8.2, for database management MySql is used. For the frontend, Twig, Bootstrap 5, and jQuery are used. Project management is done with Git using Visual Studio Code as a code editor.

The platform allows users to save time and resources by offering rentals for the university community in one place. It provides greater transparency in the real estate market and encourages competition between offers, contributing to price moderation. Furthermore, it improves the quality of life of students and university staff by facilitating access to relevant information and simplifying the housing search process.

In conclusion, the proposed platform offers a practical and efficient solution to a real problem, significantly improving the housing search experience for the university community. The platform is expected to make a positive contribution to the quality of life of those who use it.

# Índice de contenido

# Índice de figuras

# Índice de tablas

# Introducción

La búsqueda de vivienda para miembros de la comunidad universitaria es una tarea compleja debido a la dispersión de las ofertas en múltiples plataformas. Este Trabajo Fin de Grado (TFG) propone el desarrollo de una plataforma web diseñada para centralizar las ofertas de alquiler específicamente dirigidas a estudiantes y personal universitario. Esta plataforma pretende ser una solución efectiva a los problemas recurrentes que enfrenta este grupo al intentar encontrar alojamiento adecuado y asequible en un mercado inmobiliario cada vez más competitivo.

## 1.1 Motivación

En los últimos años, el precio de alquiler de viviendas ha subido considerablemente en las ciudades grandes que albergan universidades. Factores como el incremento en la demanda, la insuficiencia de oferta adecuada y la especulación en el mercado inmobiliario han influido en esta tendencia. Esta situación plantea un desafío significativo para los estudiantes y el personal universitario, quienes a menudo deben dedicar una parte considerable de su tiempo y recursos a encontrar una vivienda que cumpla con sus necesidades y presupuesto. Las plataformas actuales no están diseñadas específicamente para este grupo, lo que resulta en una experiencia de búsqueda fragmentada y, a menudo, ineficaz.

## 1.2 Objetivos

El objetivo principal es crear una plataforma en línea, totalmente gratuita tanto para los clientes como para los propietarios que facilite la búsqueda y oferta de propiedades inmobiliarias específicamente dirigidas a estudiantes universitarios, personal docente y administrativo de universidades.

Este objetivo se desglosa en varios objetivos específicos:

- Llevar a cabo un análisis de mercado para detectar las necesidades y preferencias de la comunidad universitaria en relación a la vivienda.
- Desarrollar una interfaz de usuario clara y fácil de usar que simplifique la navegación y el manejo de la plataforma.

- Crear un diseño adaptable (responsive) que funcione bien en dispositivos móviles y de escritorio.
- Desarrollar una plataforma de administración de propiedades que facilite a los usuarios la publicación, búsqueda y gestión de anuncios de alquiler y venta de inmuebles.
- Integrar funcionalidades de búsqueda avanzada con filtros específicos (ubicación, precio, tipo de propiedad, etc.).
- Implementar protocolos de seguridad que resguarden la información personal y los datos de los usuarios.

Estos objetivos proporcionan una guía estructurada para el desarrollo de TFG, garantizando que se cubren los aspectos necesarios para la creación de un portal inmobiliario exitoso y útil para la comunidad universitaria.

## 1.3 Impacto esperado

La implementación de esta plataforma se espera que tenga varios impactos positivos. En primer lugar, se prevé un ahorro considerable de tiempo y recursos para los usuarios, quienes ya no necesitarán consultar múltiples plataformas para encontrar una vivienda adecuada. En segundo lugar, proporcionar un punto de entrada único a una amplia gama de ofertas de alquiler, fomentará una mayor transparencia en el mercado inmobiliario y promoverá la competitividad entre las ofertas, lo cual podría contribuir a una moderación de los precios. Finalmente, al facilitar el acceso a la información y simplificar el proceso de búsqueda, según lo previsto la plataforma tenga un impacto positivo en la calidad de vida de los estudiantes y el personal universitario.

## 1.4 Metodología

Para alcanzar los objetivos propuestos, se propone una metodología estructurada que incluye varias fases. En primer lugar, se llevaría a cabo una revisión del estado del arte, analizando plataformas similares existentes y evaluando sus características y limitaciones. Este análisis permitirá identificar las mejores prácticas y las áreas de mejora que se podrán incorporar en la plataforma propuesta. A continuación, se procederá al diseño y desarrollo de la

base de datos y la interfaz web, integrando las fuentes de datos y desarrollando funcionalidades avanzadas de búsqueda. Este proceso incluirá la selección de las tecnologías adecuadas y la implementación de un sistema robusto y escalable. Posteriormente, se llevarán a cabo pruebas de funcionalidad, rendimiento y usabilidad para asegurar que la plataforma cumple con las expectativas y necesidades de los usuarios.

## 1.5 Estructura de la memoria

La estructura del documento se organiza en varios capítulos que se detallan a continuación:

- **Estado del arte**: análisis de las plataformas actuales y crítica de sus características.
- **Análisis del problema**: especificación de requisitos y modelado conceptual del sistema propuesto.
- **Diseño de la solución**: descripción detallada de la arquitectura del sistema y el diseño detallado de sus componentes.
- **Desarrollo de la solución propuesta**: documentación del proceso de implementación, incluyendo los desafíos y decisiones clave.
- **Evaluación y pruebas**: métodos de verificación y validación de la plataforma desarrollada.
- **Conclusiones**: reflexiones finales sobre los resultados obtenidos, el cumplimiento de los objetivos y el aprendizaje derivado del proyecto.
- **Bibliografía**: fuentes bibliográficas consultadas durante la realización del TFG.

En resumen, este TFG propone una solución práctica para un problema real que afecta a la comunidad universitaria. Al centralizar las ofertas de alquiler y proporcionar herramientas avanzadas de búsqueda, se espera que la plataforma desarrollada facilite la búsqueda de vivienda, mejorando la calidad de vida de los estudiantes y el personal universitario.

# 2. Estado del arte

En este capítulo se documentan otras aplicaciones y soluciones que actualmente existen o han existido en el mercado y que ofrecen funcionalidades similares a las que se propone desarrollar en este TFG. Además, se analizarían las posibles alternativas y se justificará el camino elegido para llevar a cabo este proyecto.

## 2.1 Plataformas Existentes

Para desarrollar un portal inmobiliario dirigido a la comunidad universitaria, es importante entender cómo funcionan las plataformas actuales y qué ofrecen. En los últimos años, las plataformas inmobiliarias en línea han cambiado la manera en que los estudiantes buscan viviendas, facilitando el acceso a una gran variedad de opciones de manera rápida y sencilla.

Para los estudiantes universitarios, encontrar alojamiento puede ser un desafío, ya que suelen tener necesidades específicas como estancias cortas y presupuestos limitados. Algunas plataformas se han especializado en este público, ofreciendo herramientas que ayudan a encontrar no solo viviendas, sino también compañeros de piso, alojamientos temporales y opciones de alquiler flexibles. A continuación, se presentarán algunas de las plataformas más relevantes, que servirán como referencia para el desarrollo de un portal inmobiliario enfocado en estudiantes universitarios.

### 2.1.1 Idealista

Idealista es una de las plataformas inmobiliarias más populares en España, aunque también está presente en Portugal e Italia, conocida por su amplio catálogo de anuncios tanto de venta como de alquiler, publicados por particulares y agencias inmobiliarias. Su interfaz permite a los usuarios filtrar las búsquedas según diversos criterios, como precio, ubicación, número de habitaciones y tipo de vivienda. Además, Idealista cuenta con una de las mayores cantidades de oferta en comparación con otras plataformas similares, lo que aumenta las probabilidades de encontrar una vivienda adecuada.

Entre los filtros que Idealista permite aplicar se encuentran:

- **Precio:** los usuarios pueden establecer un rango de precios mínimo y máximo para acotar las opciones que se ajusten a su presupuesto.
- **Ubicación:** es posible seleccionar barrios específicos dentro de una ciudad o buscar en distintas localidades.
- **Número de habitaciones:** permite buscar viviendas con un número específico de habitaciones, adaptándose a las necesidades de espacio del usuario.
- **Tipo de vivienda:** se puede elegir entre diferentes tipos de inmuebles, como apartamentos, casas, dúplex, entre otros.
- **Superficie:** filtra según el tamaño en metros cuadrados de la vivienda.
- Planta del edificio: Ideal para quienes prefieren viviendas en plantas bajas o con acceso mediante ascensor.
- **Características adicionales:** como aire acondicionado, calefacción, conexión a internet, y presencia de electrodomésticos en la cocina (nevera, horno, microondas).

Sin embargo, Idealista no está específicamente orientado a la comunidad universitaria, lo que implica que los estudiantes deben filtrar manualmente las opciones más adecuadas para sus necesidades. Esta falta de enfoque específico puede hacer que la búsqueda de alojamiento sea menos eficiente para este grupo demográfico.

Las características de las viviendas ofrecidas suelen incluir detalles como el número de habitaciones, baños, la capacidad máxima de personas, conexión a internet, aire acondicionado, calefacción, y equipamiento de la cocina. Además, información sobre las características de las habitaciones, como si están amuebladas, tienen cama doble y ventana a la calle, también es común. Es posible que algunos anuncios incluyan planos de las habitaciones, lo cual ayuda a visualizar mejor el espacio disponible.

En cuanto a la convivencia, Idealista proporciona detalles sobre los compañeros de vivienda potenciales, indicando aspectos como el rango de edad, si son estudiantes o trabajadores, y si la vivienda es LGTB friendly. También se incluye información sobre si el propietario vive en la casa y el

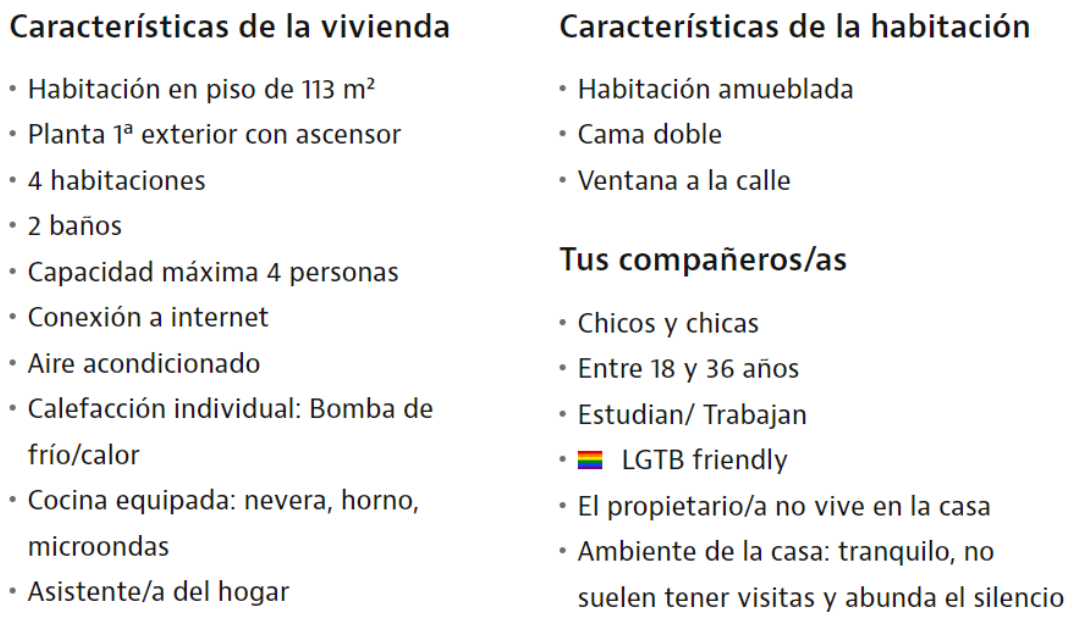
ambiente general del hogar, lo que puede permitir a los usuarios a tomar decisiones más informadas sobre su futura residencia.

Adicionalmente, Idealista tiene la ventaja de estar disponible en más de 17 idiomas, facilitando el acceso a usuarios de diversas nacionalidades. Sin embargo, presenta algunas limitaciones importantes. Aunque la plataforma cuenta con una gran oferta, muchos de los anuncios permanecen en la web incluso después de que las propiedades hayan sido alquiladas, lo que puede resultar frustrante para los usuarios. Además, el diseño visual, aunque funcional, podría parecer un poco anticuado comparado con otras plataformas modernas. Finalmente, la plataforma no ofrece un servicio de protección al arrendatario, lo cual puede dejar a los inquilinos potenciales en una situación de vulnerabilidad ante posibles fraudes o disputas con los propietarios.

*Figura 1. Página web idealista.com*


*Fuente: idealista.com*

*Figura 2. Detalles de piso en Idealista*



**Características de la vivienda**

- Habitación en piso de 113 m²
- Planta 1ª exterior con ascensor
- 4 habitaciones
- 2 baños
- Capacidad máxima 4 personas
- Conexión a internet
- Aire acondicionado
- Calefacción individual: Bomba de frío/calor
- Cocina equipada: nevera, horno, microondas
- Asistente/a del hogar

**Características de la habitación**

- Habitación amueblada
- Cama doble
- Ventana a la calle

**Tus compañeros/as**

- Chicos y chicas
- Entre 18 y 36 años
- Estudian/ Trabajan
- LGTB friendly
- El propietario/a no vive en la casa
- Ambiente de la casa: tranquilo, no suelen tener visitas y abunda el silencio

*Fuente: idealista.com*

### 2.1.2 Uniplaces

Uniplaces es una plataforma internacional especializada en el alquiler de viviendas para estudiantes, con presencia en Europa, Oceanía y América. Ofrece un proceso de reserva en línea seguro y eficiente, con la página web disponible en nueve idiomas, lo que facilita el acceso a usuarios de diferentes nacionalidades.

#### Características principales de Uniplaces

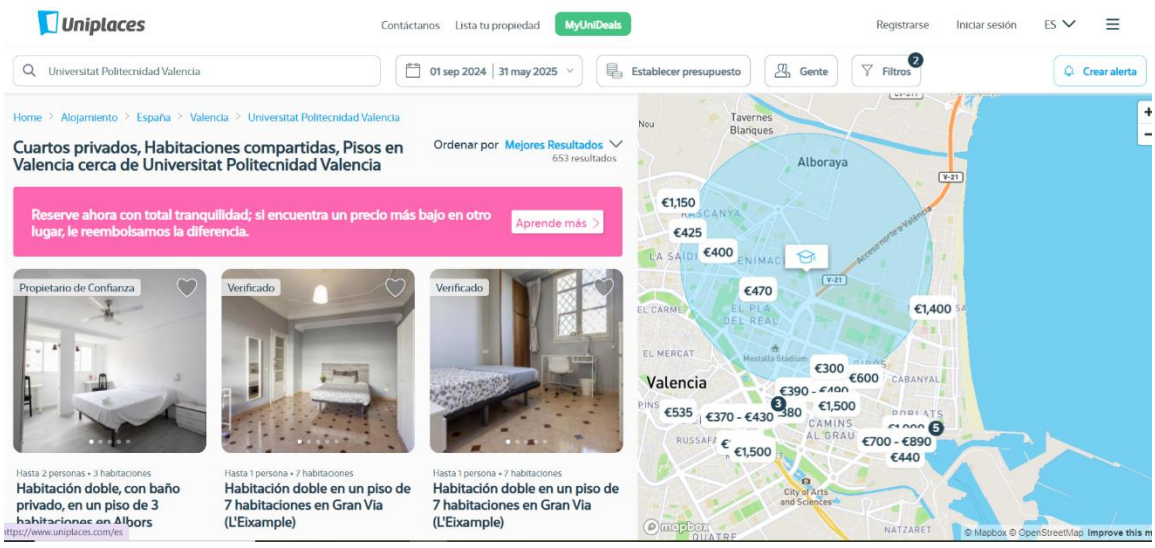
- **Alcance internacional:** Uniplaces está presente en varios países de la Unión Europea, Oceanía y América, enfocándose en ciudades universitarias importantes, permitiendo a los estudiantes encontrar alojamiento en diversas partes del mundo y adaptándose a sus necesidades de movilidad académica
- **Opciones verificadas**: la plataforma ofrece opciones de alojamiento verificadas, algunas de las cuales han sido confirmadas mediante visitas de agentes de Uniplaces, asegurando la autenticidad y calidad de los anuncios. Esta verificación proporciona una mayor tranquilidad a los estudiantes y sus familias.

- **Información detallada del propietario:** cada anuncio incluye una breve descripción del propietario, proporcionando transparencia adicional y facilitando la comunicación entre arrendador y arrendatario.
- **Reserva en línea:** Uniplaces permite realizar reservas directamente a través de su página web, simplificando el proceso de búsqueda y reserva. Los usuarios pueden consultar el calendario de disponibilidad de la habitación, lo que facilita la planificación y asegura que la habitación estará disponible en las fechas deseadas.
- **Políticas claras:** la plataforma proporciona detalles claros sobre las políticas de cancelación y las condiciones establecidas por los propietarios, asegurando que los estudiantes comprendan completamente las condiciones de su alquiler y reduciendo el riesgo de malentendidos o disputas.
- **Protección y certificación**: Uniplaces ofrece servicios de protección contra estafas, asegurando que las transacciones y acuerdos realizados a través de la plataforma sean seguros. Además, para los estudiantes internacionales, proporciona un certificado de alojamiento necesario para la solicitud de visados.
- **Gastos y servicios adicionales**: Uniplaces detalla un desglose de los gastos adicionales asociados al alquiler, permitiendo a los estudiantes planificar su presupuesto de manera más efectiva. Algunos alojamientos incluyen servicios adicionales como la limpieza, lo cual agrega comodidad y valor para los inquilinos.
- **Información completa y útil:** además de las características de las viviendas, Uniplaces proporciona información relevante sobre el entorno, incluyendo datos sobre la cercanía a universidades, transporte público y servicios básicos. Esta información ayuda a los estudiantes a encontrar el alojamiento que mejor se ajuste a sus requerimientos y estilo de vida.
- **Limitaciones:** aunque Uniplaces está orientado a estudiantes, la plataforma no siempre incluye ofertas de agencias inmobiliarias locales, lo que puede limitar la variedad de opciones disponibles. Esto se puede inferir de las experiencias de los usuarios y de la naturaleza de los anuncios, que en su mayoría son verificados directamente por la

plataforma o por propietarios individuales. Además, algunos usuarios han mencionado problemas con la atención al cliente y dificultades para resolver disputas, lo que puede ser un inconveniente adicional para quienes buscan un proceso de alquiler sin contratiempos.

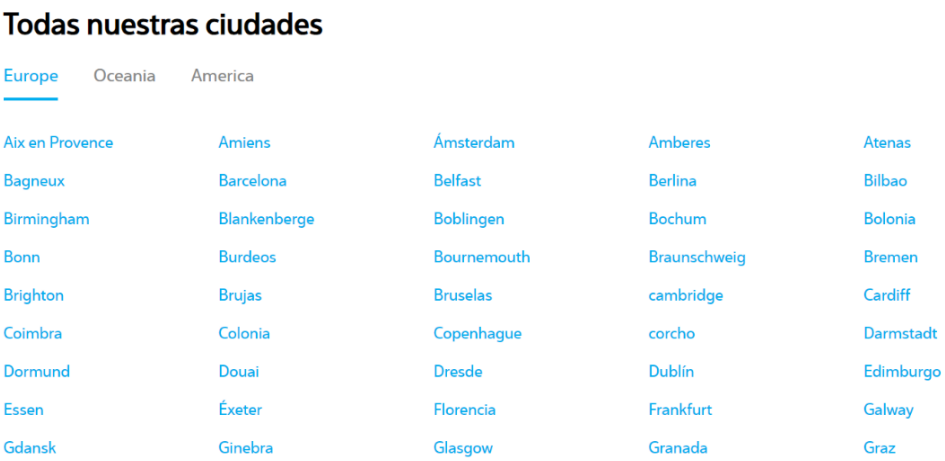
En resumen, Uniplaces es una plataforma del alquiler de viviendas estudiantiles internacional con una interfaz muy bonita e intuitiva. Es una herramienta valiosa enfocada especialmente en estudiantes que buscan un alojamiento en el extranjero. A pesar de esto, tiene ciertas limitaciones, como la falta de ofertas de agencias locales que encarece ligeramente el precio de las viviendas. Además, cobran 1 mes por el servicio del cliente final, lo que puede ser un problema en muchos casos para los estudiantes ya que tienen el presupuesto muy limitado.

*Figura 3. Página web uniplaces.com*



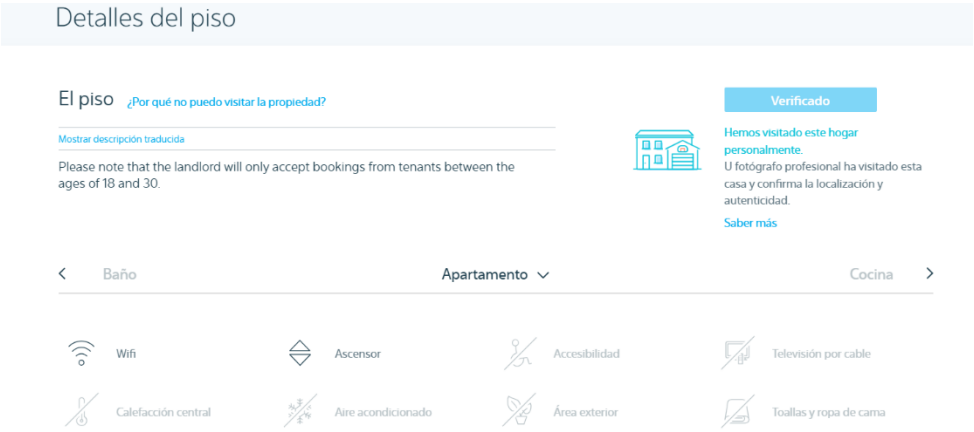

*Fuente: uniplaces.com*

*Figura 4. Ciudades en uniplaces.com*




*Fuente: uniplaces.com*

*Figura 5. Detalles de piso en uniplaces.com*




*Fuente: uniplaces.com*

### 2.1.3 Badi

Badi es una plataforma innovadora que se especializa en la búsqueda de compañeros de piso y el alquiler de habitaciones. Permite a los usuarios crear perfiles detallados y buscar compañeros compatibles basándose en intereses y estilo de vida. Esta plataforma es especialmente popular entre estudiantes y jóvenes profesionales debido a sus características avanzadas y facilidad de uso.

### Características Principales de Badi

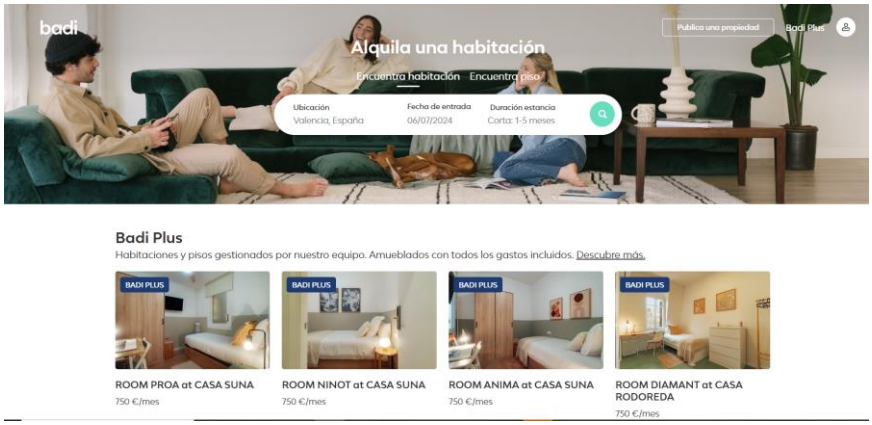
- **Visualización de candidatos:** una de las funciones destacadas de Badi es que permite a los usuarios ver la cantidad de candidatos que han enviado solicitudes al propietario de una habitación. Esto proporciona una idea de la demanda de la habitación y ayuda a evaluar la competencia, lo cual es útil para los usuarios que buscan aumentar sus probabilidades de éxito al aplicar para una habitación.
- **Servicio fast-track:** Badi ofrece un servicio adicional llamado fast-track, que posiciona a los inquilinos como candidatos prioritarios en la bandeja de mensajes de los anunciantes. Este servicio tiene un coste y no es una suscripción. Tiene una duración de 30 días, y si el usuario no ha encontrado habitación después de este periodo, el plan se extiende 30 días más de forma gratuita. Esta opción es especialmente útil para quienes necesitan encontrar alojamiento de manera rápida y eficaz.
- **Idiomas disponibles:** la plataforma está disponible en varias idiomas: español e inglés, lo que facilita el acceso a una audiencia más amplia y diversa, especialmente en ciudades internacionales y turísticas .
- **Interfaz de usuario:** los resultados de búsqueda en Badi se pueden visualizar en una lista y simultáneamente en un mapa, permitiendo a los usuarios ver la ubicación de las propiedades de un vistazo. Esto mejora la experiencia del usuario al proporcionar una visión clara de dónde se encuentran las opciones disponibles. Además, se pueden aplicar diversos filtros para especificar el tipo de estancia (corta, media y larga), el tipo de compañeros (mujer, hombre, no binario) y otras preferencias, mejorando así la personalización de la búsqueda y haciendo que sea más eficiente encontrar el lugar ideal .
- **Proceso de alquiler:** para alquilar una habitación en Badi, los usuarios deben primero solicitar una conversación por chat con el propietario. Este enfoque permite que ambas partes se conozcan mejor antes de concretar el alquiler, lo cual puede mejorar la compatibilidad y reducir conflictos futuros. Este sistema también proporciona una capa adicional de seguridad y confianza para ambas partes involucradas.

- **Información detallada:** Badi proporciona información detallada sobre el propietario y los compañeros de piso actuales, así como sobre el perfil del compañero preferido. Esta transparencia ayuda a los usuarios a tomar decisiones informadas y a encontrar una vivienda que se ajuste mejor a sus necesidades y preferencias. Saber quiénes serán los compañeros de piso potenciales y el ambiente del hogar puede ser crucial para asegurar una convivencia armoniosa.

Aunque Badi es una herramienta poderosa para encontrar compañeros de piso y habitaciones en pisos compartidos, no centraliza todas las ofertas de alquiler disponibles en el mercado. Esto significa que los usuarios podrían necesitar usar otras plataformas adicionales para llevar a cabo una búsqueda completa y exhaustiva del mercado inmobiliario. Esta limitación puede ser un inconveniente para estudiantes que buscan una gama más amplia de propiedades y opciones.

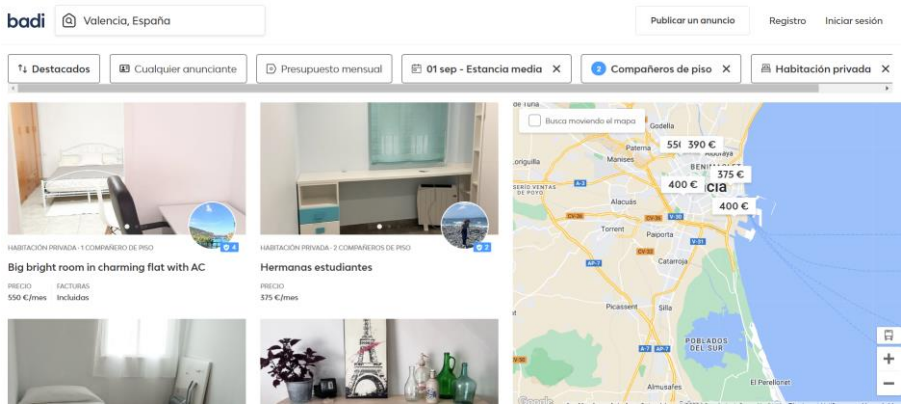
En resumen, Badi cuenta con varias características que permiten mejorar la experiencia del usuario, como la visualización de candidatos, el servicio fast-track, y una interfaz intuitiva que combina vistas de lista y mapa. A pesar de sus limitaciones en la centralización de ofertas, Badi ofrece un conjunto robusto de herramientas y servicios que permiten la búsqueda de vivienda compartida, especialmente para estudiantes y jóvenes profesionales. La información detallada sobre los compañeros de piso y la posibilidad de conversar con los propietarios antes de concretar el alquiler proporcionan una capa adicional de seguridad y confianza, haciendo que el proceso de búsqueda de alojamiento sea más transparente y accesible.

*Figura 6. Página principal de Badi*



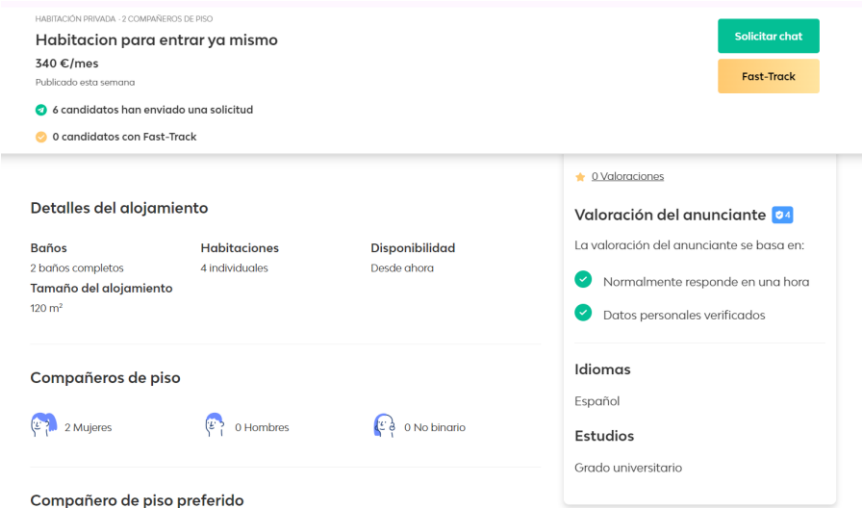
*Fuente: badi.com*

*Figura 7. Página de búsqueda de viviendas en Badi*




*Fuente: badi.com*

*Figura 8. Detalle de una vivienda en Badi*




*Fuente: badi.com*

### 2.1.4 Buscador de pisos de UPV

Portal de Alquiler de Pisos de la UPV es un servicio ofrecido por la Universidad Politécnica de Valencia para ayudar a sus estudiantes a encontrar alojamiento. Este portal proporciona información básica sobre las propiedades disponibles y los datos de contacto del propietario.

#### Características principales del portal de alquiler de pisos de la UPV:

**Información básica de las propiedades:** el portal ofrece una lista de propiedades disponibles para alquilar, incluyendo detalles como ubicación,

tamaño y precio. Sin embargo, no proporciona fotos de las propiedades, lo que limita la capacidad de los usuarios para visualizar el espacio antes de contactarse con el propietario.

**Contacto directo con el propietario:** cada anuncio incluye el número de teléfono del propietario, permitiendo a los usuarios contactar directamente para recibir más información o solicitar una visita. Este método de contacto directo puede agilizar el proceso de alquiler, pero también requiere que los usuarios tomen la iniciativa para recopilar más detalles y evaluar la propiedad en persona.

**Orientado a la comunidad universitaria:** el servicio está específicamente diseñado para la comunidad universitaria de la UPV, facilitando la búsqueda de alojamiento para estudiantes, profesores y personal universitario. Las propiedades listadas están en ubicaciones convenientes para acceder a las instalaciones universitarias, asegurando que los usuarios encuentren opciones de vivienda adecuadas cerca de la universidad.

### Limitaciones

**Falta de imágenes:** la ausencia de fotos en los anuncios puede ser un inconveniente significativo, ya que los potenciales inquilinos no pueden ver el estado y la apariencia del alojamiento antes de ponerse en contacto con el propietario. Esto puede hacer que el proceso de selección sea más tedioso y requiera más tiempo.

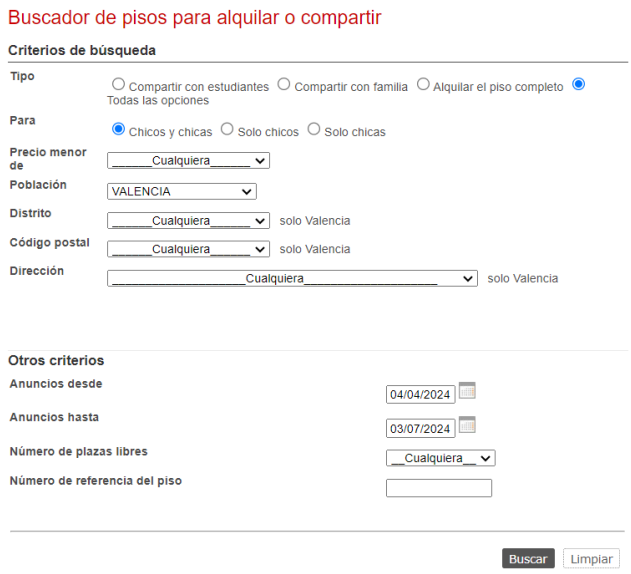
**Información limitada en línea:**

Al ofrecer solo información básica y un número de contacto, los usuarios deben hacer un esfuerzo adicional para obtener detalles completos sobre la propiedad. Esto incluye averiguar aspectos importantes como condiciones del contrato, servicios incluidos y estado real de la vivienda a través de visitas presenciales o comunicaciones directas con el propietario.

En resumen, el Portal de Alquiler de Pisos de la UPV es una herramienta útil para la comunidad universitaria en la búsqueda de alojamiento, proporcionando una plataforma específica y accesible para este propósito. A pesar de las limitaciones, como la falta de fotos y la información limitada en línea, el portal

sigue siendo una opción valiosa para encontrar alojamiento cerca de la universidad debido a su enfoque en satisfacer las necesidades de la comunidad universitaria de la UPV. La capacidad de contactar directamente a los propietarios y la orientación hacia ubicaciones convenientes hacen que este servicio sea una solución práctica para estudiantes y personal de la universidad.

*Figura 9. Formulario de búsqueda de pisos en buscador de UPV*




*Fuente: Buscador de pisos de UPV*

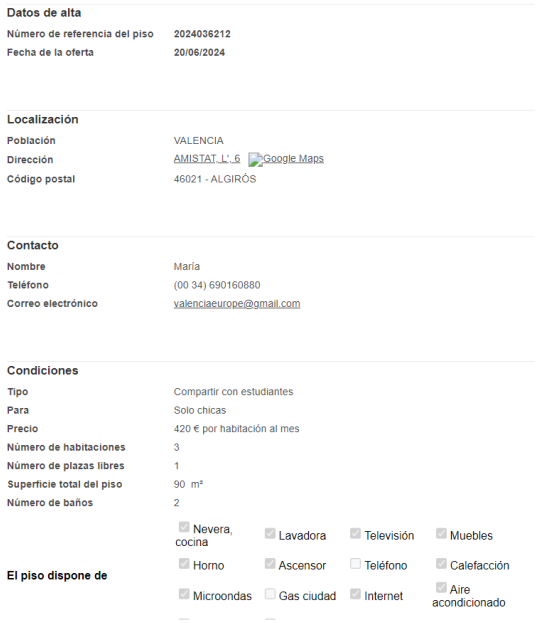
*Figura 10. Interfaz para listar pisos en buscador de pisos de UPV*

**Resultado de la búsqueda de pisos**

| Dirección | CP | Tipo | Para | Plazas | Precio | Fecha | Contacto | |
|---|---|---|---|---|---|---|---|---|
| ALBOCACER, 15 | 46020 | Piso completo | Chicos y chicas | 3 | 1140 € | 27/06/24 | Claudia | 605957501 |
| ALGUER, L', 30 | 46022 | Con estudiantes | Chicos y chicas | 3 | 390 € | 10/06/24 | Pilar | 677110558 |
| ALMANSA, 1 | 46001 | Con estudiantes | Chicos y chicas | 6 | 450 € | 19/06/24 | Paloma | 687596003 |
| AMADEO DE SABOYA, 3 | 46010 | Con familia | Chicos y chicas | 1 | 550 € | 25/04/24 | Francisco | 660412293 |
| AMISTAT, L', 6 | 46021 | Con estudiantes | Chicas | 1 | 420 € | 20/06/24 | María | 690160880 |
| BARCELONINA, 3 | 46002 | Con estudiantes | Chicos y chicas | 2 | 400 € | 05/06/24 | Pilar | 660484148 |
| BARRACA, 45 | 46011 | Con estudiantes | Chicos y chicas | 2 | 350 € | 17/06/24 | Vicente | 636497707 |
| BEATO JUAN GRANDE, 10 | 46011 | Piso completo | Chicos y chicas | 3 | 990 € | 05/06/24 | Francesc | 600819837 |
| BEATRIZ TORTOSA, 4 | 46021 | Piso completo | Chicos y chicas | 1 | 780 € | 26/06/24 | Enrique | 620659757 |
| BELGICA, 26 | 46021 | Con estudiantes | Chicos y chicas | 1 | 400 € | 21/06/24 | Roque | 623060360 |

*Fuente: Buscador de pisos de UPV*

*Figura 11. Interfaz con detalles de una vivienda en buscador de pisos de UPV*




*Fuente: Buscador de pisos de UPV*

### 2.1.5 Erasmusplay

Erasmus Play es una plataforma diseñada para ayudar a estudiantes internacionales a encontrar alojamiento en diversas ciudades europeas. Tiene una amplia gama de viviendas, desde habitaciones individuales hasta apartamentos completos y residencias estudiantiles.

#### Características Principales de Erasmus Play

**Diversidad de alojamientos:** la plataforma incluye habitaciones individuales, apartamentos completos, residencias y pisos compartidos para estudiantes. Los usuarios pueden comparar alojamientos de diversas plataformas en línea, asegurando una amplia gama de opciones verificadas.

**Filtros personalizados:** los usuarios pueden aplicar una variedad de filtros para encontrar el alojamiento ideal según sus necesidades. Los filtros incluyen tipo de alojamiento, precio, características del cuarto (como cama individual o doble), y reglas del alojamiento. También es posible filtrar por fecha de inicio y fin de estancia, lo cual es útil para ajustar la búsqueda a las fechas académicas

**Visualización y comparación de propiedades:** Erasmus Play permite a los usuarios ver detalles importantes de cada alojamiento, como la ubicación exacta, el precio, descripción del alojamiento, y las características más

importantes de la casa (por ejemplo, si tiene lavadora, internet, etc.). Además, los anuncios incluyen un calendario de disponibilidad, lo que facilita la planificación de la estancia.

**Idiomas disponibles:** la plataforma está disponible en inglés, español e italiano, lo que facilita su uso a una audiencia internacional. Erasmus Play está trabajando para adaptarse a otros idiomas en el futuro.

**Colaboración con universidades:** Erasmus Play colabora estrechamente con diversas universidades europeas, lo que le permite ofrecer un servicio optimizado y confiable para los estudiantes. Estas colaboraciones aseguran que la plataforma cumpla con las necesidades específicas de la comunidad estudiantil.

**Seguridad y verificación:** la plataforma garantiza la seguridad de los alojamientos mediante la verificación de sus socios. Todos los listados en Erasmus Play son verificados, proporcionando un nivel adicional de confianza para los usuarios. Además, Erasmus Play no requiere registro para la búsqueda inicial, lo que facilita el acceso a la información.

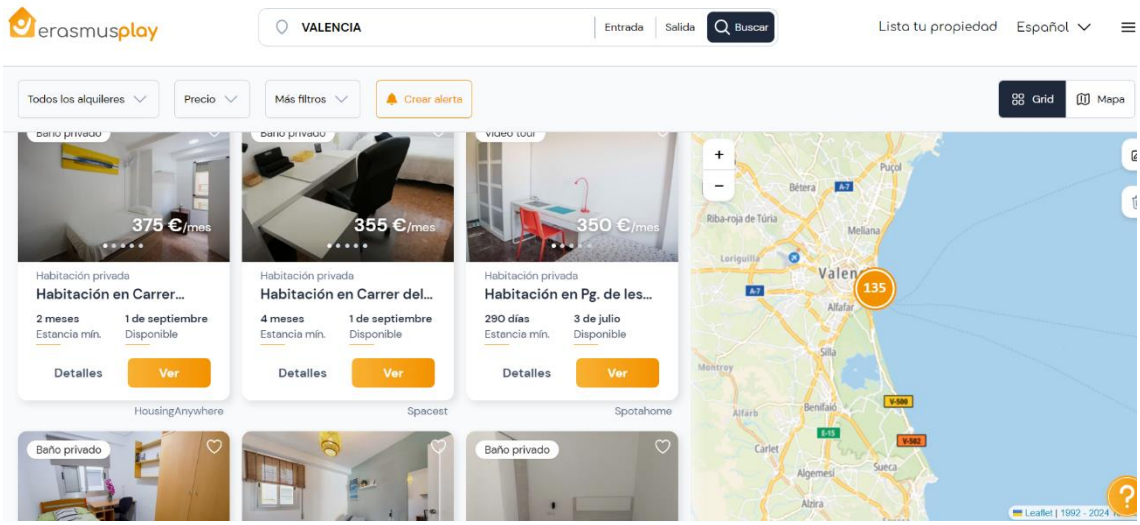
**Gratuita para los usuarios:** Erasmus Play es completamente gratuita para los usuarios, gracias a sus colaboraciones con universidades y plataformas de alojamiento. Los estudiantes pueden buscar y comparar opciones de alojamiento sin incurrir en costos adicionales.

**Redirección a plataformas colaborativas:** al seleccionar un anuncio, la página web redirige a los usuarios a las plataformas colaborativas donde se publicó el anuncio originalmente. Esto permite acceder a la información completa y diferentes posibilidades de reserva en el sitio original

En resumen, Erasmus Play es una plataforma eficiente y confiable para estudiantes internacionales que buscan alojamiento en Europa. Tiene gran variedad de opciones de vivienda, filtros personalizados para facilitar la búsqueda, y colaboración con universidades que garantiza la calidad y seguridad de los alojamientos. Aunque actualmente está disponible en tres idiomas, Erasmus Play está en constante evolución para adaptarse a una

audiencia global, lo que la convierte en una herramienta valiosa para estudiantes de intercambio

*Figura 12. Interfaz de búsqueda de viviendas en erasmusplay*




*Fuente:* erasmusplay.com

### 2.1.6 Erasmusu

Erasmusu es una plataforma integral que se centra en facilitar la experiencia Erasmus y de intercambio para estudiantes internacionales. Tiene gran cantidad de servicios que van más allá del simple alquiler de viviendas, convirtiéndose en un recurso clave para los estudiantes.

#### Características Principales de Erasmusu

**Diversidad de alojamientos:** la plataforma incluye habitaciones individuales, apartamentos completos, residencias estudiantiles y pisos compartidos en numerosas ciudades alrededor del mundo. Esto permite a los estudiantes encontrar la opción de alojamiento que mejor se adapte a sus necesidades

**Filtros personalizados:** los usuarios pueden aplicar filtros avanzados para encontrar el alojamiento ideal según diversos criterios como el tipo de propiedad, el rango de precios y las características propias de cada vivienda. Esta personalización ayuda a ajustar la búsqueda a las necesidades individuales de los estudiantes

**Información detallada de los anuncios:** cada anuncio en Erasmusu incluye una descripción detallada del alojamiento, las características más importantes

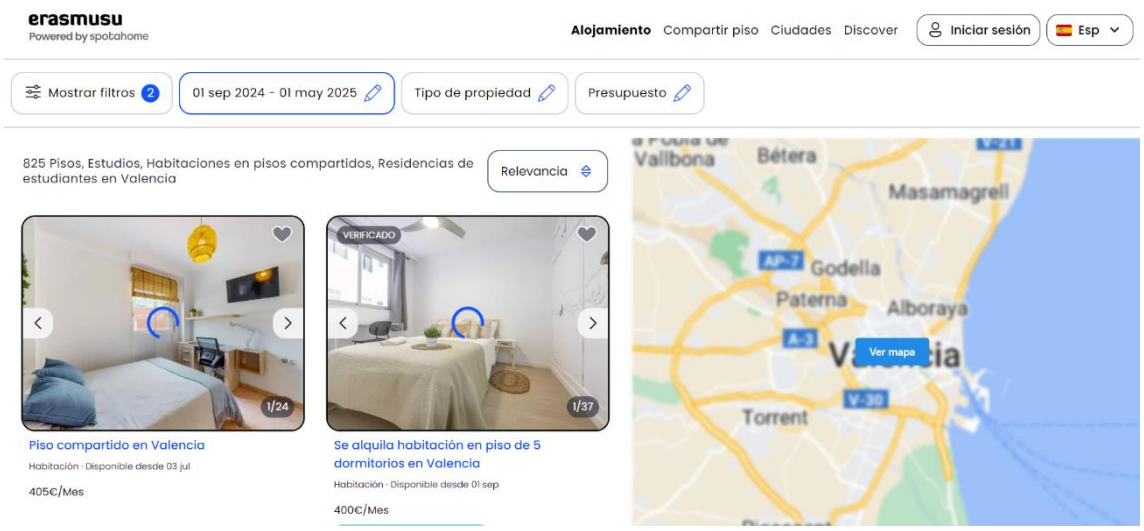
de la casa (como si tiene lavadora, internet, etc.), la ubicación exacta y un calendario de disponibilidad. Esto proporciona a los usuarios una visión completa del alojamiento antes de hacer una reserva

**Idiomas disponibles:** Erasmusu está disponible en múltiples idiomas, facilitando el uso a una audiencia internacional. Esto es esencial para estudiantes que se mudan a países donde se habla un idioma diferente al suyo

**Servicios adicionales:** además de la búsqueda de alojamiento, Erasmusu ofrece otros recursos útiles como foros de discusión, información sobre universidades, oportunidades de empleo, lecciones privadas, y blogs de experiencias Erasmus. Estos servicios adicionales hacen de Erasmusu una herramienta integral para estudiantes internacionales, ayudándoles a integrarse y mejorar al máximo su experiencia en el extranjero.

En conclusión, Erasmusu es una plataforma completa y multifuncional que facilita la vida de los estudiantes internacionales al proporcionar una gran gama de viviendas y servicios adicionales. La capacidad de aplicar filtros personalizados, ver detalles completos de los anuncios, y la disponibilidad en varios idiomas hacen de Erasmusu una herramienta esencial para estudiantes de intercambio. Además, los servicios adicionales y la redirección a plataformas colaborativas aseguran que los estudiantes tengan acceso a la información más completa y confiable para su estancia en el extranjero.

*Figura 13. Interfaz de búsqueda de viviendas en erasmusu*




*Fuente:* erasmusu.com

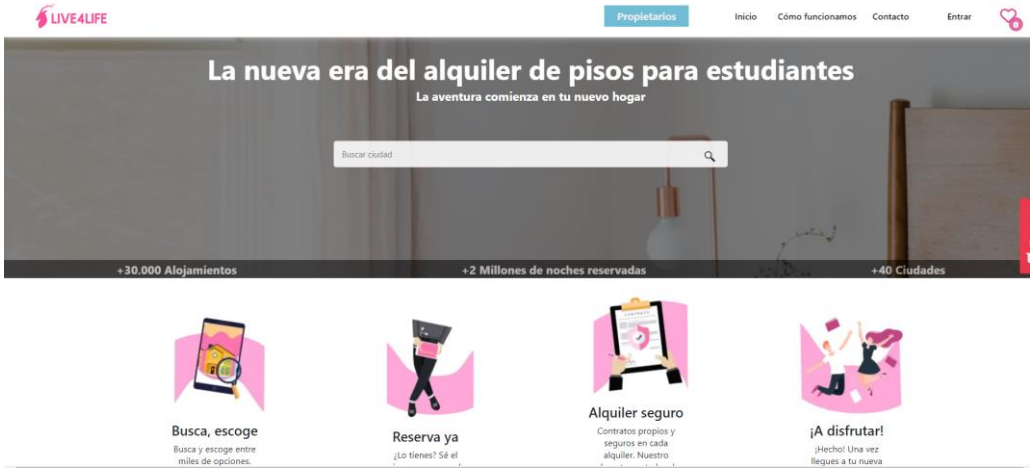
### 2.1.7 Live4Life

Live4Life es una plataforma especializada en el alquiler de viviendas para estudiantes y jóvenes profesionales en ciudades universitarias como Valencia. Ofrece una variedad de opciones de alojamiento, incluyendo pisos compartidos y estudios. Live4Life proporciona una interfaz fácil de usar y enfocada en las necesidades de los estudiantes, permitiendo la búsqueda de alojamiento por ubicación, precio, tipo de habitación y otras características relevantes.

Sin embargo, aunque Live4Life cuenta con filtros específicos, estos presentan ciertas limitaciones. Por ejemplo, los filtros de búsqueda actuales permiten a los usuarios seleccionar características básicas como el tipo de habitación y el precio, pero carecen de opciones más avanzadas que podrían ser extremadamente útiles para los estudiantes, tales como la proximidad a servicios específicos (bibliotecas, cafeterías, instalaciones deportivas), detalles sobre el ambiente del vecindario, o la disponibilidad de servicios adicionales como limpieza y lavandería. Además, la implementación de estos filtros no siempre es intuitiva, lo que puede dificultar la personalización efectiva de la búsqueda por parte de los usuarios. Estas carencias limitan la capacidad de los estudiantes para encontrar opciones de vivienda que se adapten perfectamente a sus necesidades y preferencias específicas.

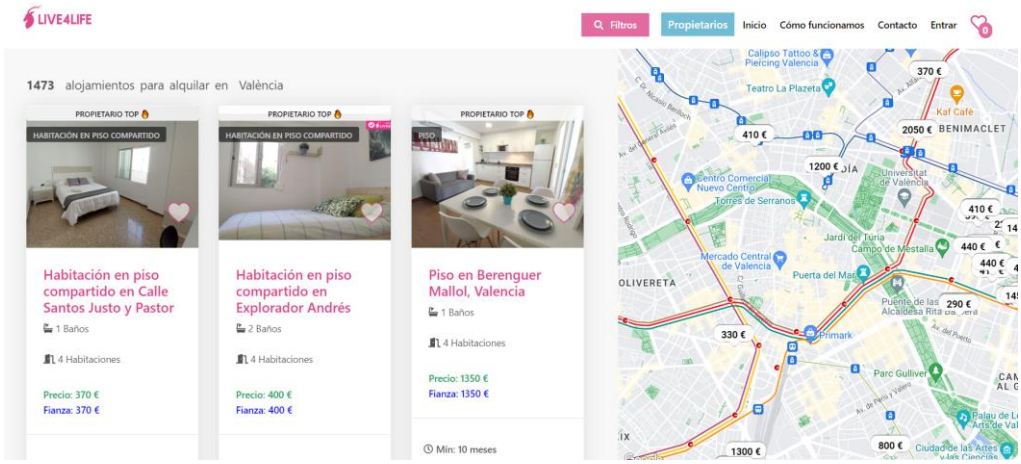
Las plataformas mencionadas anteriormente tienen diversas fortalezas y debilidades en relación con la búsqueda de vivienda para la comunidad universitaria. Idealista y Fotocasa ofrecen un gran volumen de anuncios, pero carecen de un enfoque especializado en estudiantes. Uniplaces está orientado a estudiantes, pero su cobertura no siempre es completa a nivel local. Badi ofrece una solución interesante para quienes buscan compartir piso, pero no centraliza todas las ofertas de alquiler disponibles. Live4Life se enfoca en estudiantes y jóvenes profesionales, ofreciendo una interfaz amigable y opciones que incluyen servicios adicionales, pero sus filtros de búsqueda son limitados y su alcance geográfico es reducido.

*Figura 14. Interfaz de la página principal de Live4Life*




*Fuente: live4life.es*

*Figura 15. Interfaz de la página de búsqueda de viviendas en Live4Life*




*Fuente: live4life.es*

## 2.2 Análisis del estado del Arte

Las plataformas inmobiliarias actuales dirigidas a estudiantes presentan varios problemas que justifican la creación de una nueva aplicación más adaptada a sus necesidades.

Muchas de estas plataformas no son completamente gratuitas, lo que representa un obstáculo para los estudiantes con presupuestos limitados. Además, suelen estar diseñadas para un público general, lo que significa que no abordan de manera específica las necesidades de los estudiantes, como la búsqueda de alojamiento temporal o de bajo costo. Las plataformas vinculadas a universidades a menudo ofrecen opciones limitadas y no están bien

integradas con otros servicios importantes, lo que complica la experiencia del usuario.

Asimismo, muchas de estas plataformas no están diseñadas pensando en las necesidades y habilidades técnicas de los estudiantes, lo que puede hacer que su uso sea difícil o frustrante. Además, aunque existen algunas plataformas enfocadas en la movilidad internacional, estas suelen tener una oferta limitada y no siempre están bien conectadas con los servicios universitarios, lo que fragmenta la experiencia del usuario.

Por todo esto, es evidente la necesidad de desarrollar una nueva plataforma que sea gratuita o de bajo costo, que ofrezca una experiencia de usuario sencilla y adaptada a las necesidades concretas de los estudiantes, que integre servicios útiles y que facilite la movilidad internacional mediante una oferta amplia y bien conectada con las universidades. Esta nueva plataforma busca superar las limitaciones actuales y proporcionar una solución más efectiva y accesible para los estudiantes universitarios en la búsqueda de alojamiento.

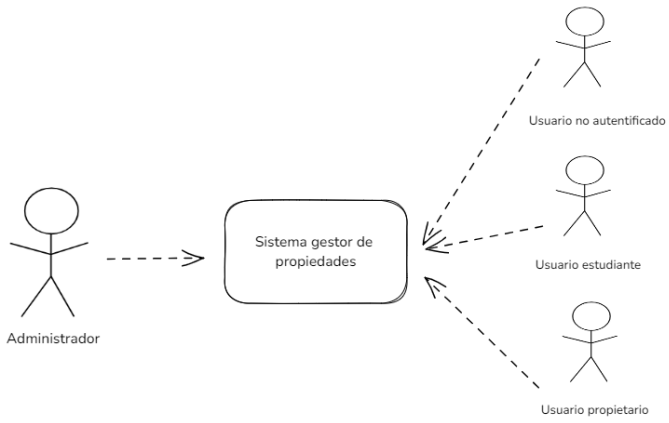
# 3. Análisis del problema

En este capítulo se identifican los requisitos necesarios para la creación de una plataforma que centralice las ofertas de alquiler dirigidas a estudiantes y personal universitario. Esto incluye la recopilación de datos, la modelación del sistema y la creación de diagramas que faciliten la comprensión y desarrollo del proyecto.

## 3.1 Diagrama de contexto

El diagrama de contexto es una representación visual que muestra el sistema propuesto y su interacción con los actores externos. Este diagrama identifica las principales entidades involucradas y las relaciones entre ellas. La figura 16 contiene el diagrama de contexto de la aplicación donde se muestra la estructura jerárquica de usuarios con diferentes roles:

*Figura 16. Diagrama de contexto*




*Fuente: Elaboración propia utilizando excalidraw.com*

## 3.2 Casos de uso

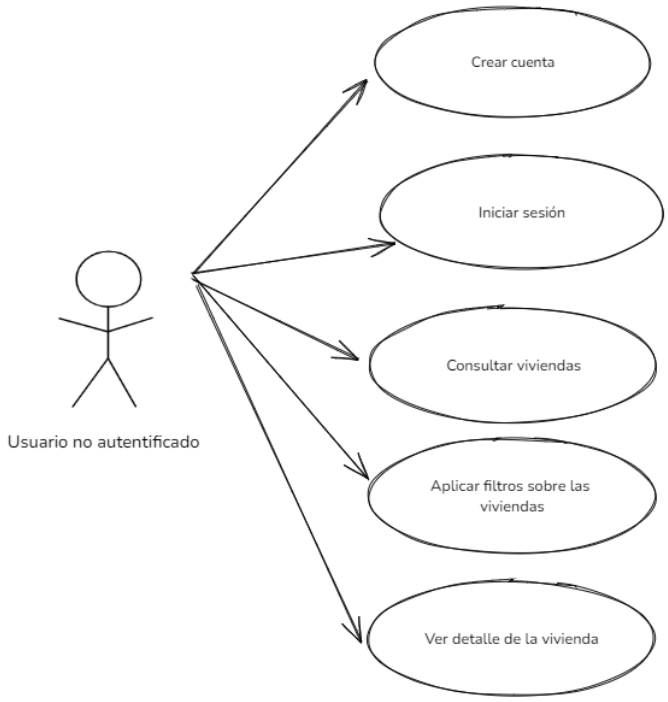
Los casos de uso son una herramienta esencial en la ingeniería de software utilizada para capturar los requisitos funcionales de un sistema. Describen las comunicación entre un actor y el sistema para lograr un determinado objetivo. Los casos de uso ayudan a identificar qué funcionalidades deben ser implementadas y cómo interactuarán los usuarios con el sistema. Cada caso de uso se centra en un único objetivo del usuario y detalla los pasos necesarios para completar una tarea, incluyendo posibles variantes y excepciones.

A continuación, se describen los diagramas de casos de uso de los actores que van a interactuar directamente con el sistema.

#### *3.2.1.1 Casos de uso de usuario no autentificado*

Los usuarios no autentificados son los visitantes de la plataforma que pueden buscar propiedades, pero no pueden guardar favoritos ni contactar con propietarios sin registrarse.

*Figura 17. Casos de uso de usuario no autentificado*

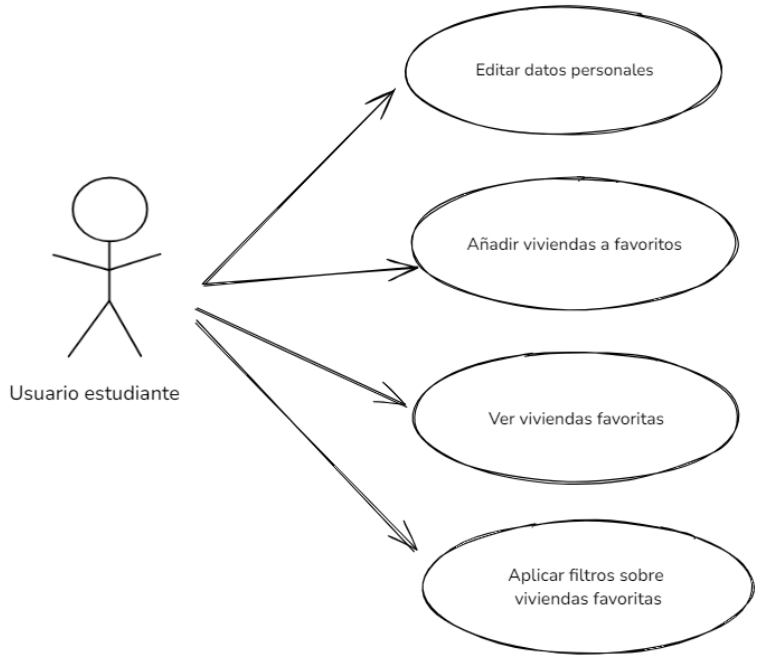



*Fuente: Elaboración propia utilizando excalidraw.com*

#### *Casos de uso de usuario estudiante*

Estudiantes son los usuarios que han iniciado sesión guardar favoritos, contactar con propietarios y gestionar su perfil, además de las funciones de usuarios no autentificados.

*Figura 18. Casos de uso de usuario estudiante*

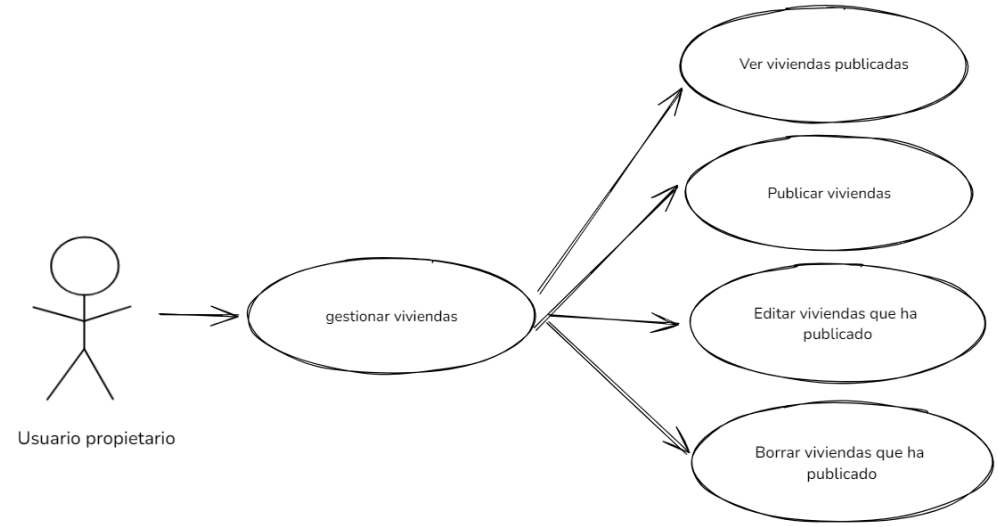



*Fuente: Elaboración propia utilizando excalidraw.com*

### *Casos de uso de usuario propietario*

Los propietarios son los usuarios que pueden realizar todas las funciones de usuarios estudiantes, pero además tienen permiso para añadir/editar detalles de las propiedades viviendas que han publicado.

*Figura 19. Casos de uso de usuario propietario*




*Fuente: Elaboración propia utilizando excalidraw.com*

#### *Casos de uso de usuario administrador*

Es el usuario encargado de la gestión general de la plataforma, puede listar, editar y borrar todas las viviendas publicadas en la plataforma y también puede gestionar los otros usuarios.

*Figura 20. Casos de uso de usuario administrador*


*Fuente: Elaboración propia utilizando excalidraw.com*

### 3.2.2 Fichas de casos de uso

#### *Usuario no autentificado*

| | | |
|---|---|---|
| Caso de uso | Crear cuenta | |
| Precondición | El usuario aún no ha iniciado la sesión | |
| Descripción | Acción de crear una nueva cuenta para el usuario | |
| Secuencia normal | Paso | Acción |
| | 1 | El usuario selecciones si es un propietario o estudiante |
| | 2 | El usuario rellena los datos personales |
| | 3 | El sistema crea una cuenta de usuario y redirecciona al usuario en la página principal |
| Postcondición | Al usuario se le envía un correo para que el usuario confirme su cuenta | |
| Excepciones | - | |

| | | |
|---|---|---|
| Caso de uso | Iniciar sesión | |
| Precondición | El usuario aún no ha iniciado la sesión | |
| Descripción | Acción de iniciar sesión | |
| Secuencia normal | Paso | Acción |
| | 1 | El usuario solicita iniciar sesión pulsando el botón correspondiente |
| | 2 | El sistema muestra un formulario |

| | | |
|---|---|---|
| | 3 | El usuario rellena el usuario y la contraseña y envía el formulario |
| | 4 | El sistema valida si el usuario existe y lo redirecciona a la página principal |
| Postcondición | Al usuario se le envía un correo para que el usuario confirme su cuenta | |
| Excepciones | Paso | Acción |
| | 4 | Si el usuario no existe el sistema avisa al usuario mostrándole un error y no le dejará entrar |

| | | |
|---|---|---|
| Caso de uso | Consultar viviendas | |
| Precondición | - | |
| Descripción | Acción de ver las viviendas existentes en la plataforma | |
| Secuencia normal | Paso | Acción |
| | 1 | El usuario entra en la plataforma |
| | 2 | El usuario selecciona el tipo de la vivienda que le interesa y la ciudad donde busca el alojamiento |
| | 3 | El sistema muestra las viviendas existentes con paginación y los muestra en el mapa |
| Postcondición | - | |
| Excepciones | - | |

| | | |
|---|---|---|
| Caso de uso | Aplicar filtros sobre las viviendas | |
| Precondición | Consultar viviendas | |
| Descripción | Es la acción para filtrar las viviendas por diferentes criterios | |
| Secuencia normal | Paso | Acción |
| | 1 | El usuario rellena el formulario donde introduce los criterios de búsqueda que le interesan |
| | 2 | El sistema muestra las viviendas que cumplen criterios de búsqueda con paginación y los muestra en el mapa |
| Postcondición | - | |
| Excepciones | - | |

| | | |
|---|---|---|
| Caso de uso | Ver detalles de la vivienda | |
| Precondición | Consultar viviendas | |
| Descripción | Es la acción para ver detalles de una vivienda | |
| Secuencia normal | Paso | Acción |
| | 1 | El usuario hace click sobre la vivienda que le interesa |

| | | |
|---|---|---|
| | 2 | El sistema muestra los detalles de la vivienda |
| Postcondición | - | |
| Excepciones | - | |

*Usuario estudiante*

| | | |
|---|---|---|
| Caso de uso | Editar datos personales | |
| Precondición | Iniciar sesión | |
| Descripción | Acción de editar datos personales de usuario | |
| Secuencia normal | Paso | Acción |
| | 1 | El usuario entra en su perfil de usuario pulsando el botón correspondiente |
| | 2 | El usuario cambia los datos personales |
| | 3 | El sistema cambia esos datos y notifica al usuario |
| Postcondición | Si se actualiza el correo al usuario se le envía un correo para que el usuario confirme su cuenta | |
| Excepciones | - | |

| | | |
|---|---|---|
| Caso de uso | Añadir viviendas a favoritos | |
| Precondición | • Iniciar sesión<br>• Consultar viviendas | |
| Descripción | Acción de añadir viviendas a favoritos | |
| Secuencia normal | Paso | Acción |
| | 1 | El usuario añade la vivienda a favoritos haciendo click sobre el botón correspondiente |
| | 2 | La plataforma guarda la vivienda en favoritos y que muestra al usuario un icono diferente que muestre el estado cambiado |
| Postcondición | - | |
| Excepciones | - | |

| | | |
|---|---|---|
| Caso de uso | Ver viviendas favoritas | |
| Precondición | • Iniciar sesión<br>• Consultar viviendas | |
| Descripción | Acción de ver viviendas favoritas | |
| Secuencia normal | Paso | Acción |
| | 1 | El usuario pulsa el botón para listar viviendas favoritas |
| | 2 | El sistema muestra al usuario las viviendas favoritas |
| Postcondición | - | |
| Excepciones | - | |

| Caso de uso | Aplicar filtros sobre viviendas favoritas | |
|---|---|---|
| Precondición | • Iniciar sesión<br>• Ver viviendas favoritas | |
| Descripción | Acción de aplicar criterios de búsqueda sobre viviendas favoritas | |
| Secuencia normal | Paso | Acción |
| | 1 | El usuario rellena el formulario donde introduce los criterios de búsqueda que le interesan |
| | 2 | El sistema muestra las viviendas favoritas que cumplen criterios de búsqueda con paginación y los muestra en el mapa |
| Postcondición | - | |
| Excepciones | - | |

*Usuario propietario*

| Caso de uso | Ver viviendas publicadas | |
|---|---|---|
| Precondición | • Iniciar sesión con rol propietario | |
| Descripción | Acción de publicar una vivienda | |
| Secuencia normal | Paso | Acción |
| | 1 | El usuario entra en área de clientes pulsando el botón correspondiente en el menú de la aplicación |
| | 2 | El sistema redirecciona al usuario al área de clientes |
| | 3 | El usuario entra en el apartado “Viviendas” en el menú del área de los clientes |
| | 4 | El sistema muestra las viviendas que ha creado el usuario |
| Postcondición | - | |
| Excepciones | - | |

| Caso de uso | Publicar viviendas | |
|---|---|---|
| Precondición | • Iniciar sesión con rol propietario<br>• Ver viviendas publicadas | |
| Descripción | Acción de publicar una vivienda | |
| Secuencia normal | Paso | Acción |
| | 1 | Desde “Ver viviendas publicadas” el usuario pulsa el botón de publicar una vivienda nueva |
| | 2 | El sistema muestra el formulario para crear una vivienda nueva |
| | 3 | El usuario rellena y envía el formulario pulsando el botón correspondiente |

| | | |
|---|---|---|
| | 4 | El sistema crea la vivienda y redirecciona al usuario a “Ver detalles de la vivienda” |
| Postcondición | - | |
| Excepciones | - | |

| | | |
|---|---|---|
| Caso de uso | Editar viviendas que ha publicado | |
| Precondición | • Iniciar sesión con rol propietario<br>• Ver viviendas publicadas | |
| Descripción | Acción de editar una vivienda | |
| Secuencia normal | Paso | Acción |
| | 1 | Desde “Ver viviendas publicadas” el usuario pulsa el botón de editar una vivienda publicada |
| | 2 | El sistema muestra el formulario con datos previamente rellenados |
| | 3 | El usuario modifica los datos y envía el formulario pulsando el botón correspondiente |
| | 4 | El sistema actualiza los datos de la vivienda y redirecciona al usuario a “Ver detalles de la vivienda” |
| Postcondición | - | |
| Excepciones | - | |

| | | |
|---|---|---|
| Caso de uso | Borrar viviendas que ha publicado | |
| Precondición | • Iniciar sesión con rol propietario<br>• Ver viviendas publicadas | |
| Descripción | Acción de borrar una vivienda | |
| Secuencia normal | Paso | Acción |
| | 1 | Desde “Ver viviendas publicadas” el usuario pulsa el botón de borrar una vivienda publicada |
| | 2 | El sistema muestra un modal avisando de que esta acción es irreversible |
| | 3 | El usuario confirma la intención de borrar |
| | 4 | El sistema borra la vivienda de la plataforma |
| Postcondición | - | |
| Excepciones | - | |

### *Usuario administrador*

| | | |
|---|---|---|
| Caso de uso | Ver usuarios | |
| Precondición | • Iniciar sesión con rol administrador | |
| Descripción | Acción de ver todos usuarios | |
| | Paso | Acción |

| | | |
|---|---|---|
| Secuencia normal | 1 | El usuario entra en área de clientes pulsando el botón correspondiente en el menú de la aplicación |
| | 2 | El sistema redirecciona al usuario al área de clientes |
| | 3 | El usuario entra en el apartado “Usuarios” en el menú del área de los clientes |
| | 4 | El sistema muestra los usuarios registrados en la plataforma |
| Postcondición | - | |
| Excepciones | - | |

| | | |
|---|---|---|
| Caso de uso | Borrar usuario | |
| Precondición | • Iniciar sesión con rol administrador<br>• Ver usuarios | |
| Descripción | Acción de borrar un usuario | |
| Secuencia normal | Paso | Acción |
| | 1 | Desde “Ver usuarios” el usuario pulsa el botón de borrar una vivienda publicada |
| | 2 | El sistema muestra un modal avisando de que esta acción es irreversible |
| | 3 | El usuario confirma la intención de borrar |
| | 4 | El sistema actualiza elimina al usuario de la plataforma |
| Postcondición | - | |
| Excepciones | - | |

# 4. Diseño de la solución

A continuación, se detallan los componentes clave de la arquitectura del sistema, el diseño detallado del mismo y las tecnologías utilizadas para su desarrollo.

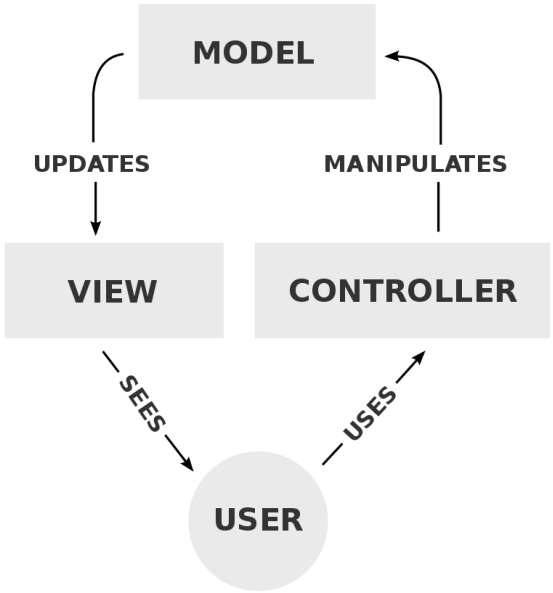
## 4.1 Arquitectura del Sistema

La arquitectura del sistema para el portal inmobiliario dirigido a la comunidad universitaria se basa en el patrón de diseño Modelo-Vista-Controlador (MVC). MVC es una arquitectura de software que divide una aplicación en tres componentes principales: el Controlador, la Vista y el Model. Esta división permite un desarrollo modular y disminuye el coste del mantenimiento y la escalabilidad del sistema.

- **Modelo**: el modelo representa la lógica de negocio y la estructura de datos de la aplicación. Es responsable de manejar los datos y las reglas de negocio, así como la interacción con la base de datos. En nuestro caso, el modelo incluye las entidades de la base de datos que representan propiedades, usuarios y otros datos relevantes para el portal inmobiliario. El modelo puede ser manipulado por el controlador y es capaz de actualizar la vista con los datos procesados.
- **Vista**: la vista es la interfaz de usuario. Su principal responsabilidad es presentar los datos del modelo de manera adecuada y enviar las interacciones del usuario al controlador. Las vistas en nuestro sistema son las páginas web diseñadas con HTML5, Bootstrap 5 y jQuery, que los usuarios utilizan para interactuar con el portal inmobiliario. La vista se actualiza según los cambios en el modelo para reflejar la información más reciente y relevante para el usuario.
- **Controlador**: el controlador es el componente intermediario entre el modelo y la vista. Recibe los datos del usuario a través de la vista, procesa estas entradas y actualiza el modelo en consecuencia. En nuestro portal inmobiliario, los controladores se encargan de gestionar las solicitudes HTTP, coordinar las acciones del usuario, y actualizar la vista con los resultados pertinentes. El controlador es el encargado de

manejar la lógica de la aplicación, dirigiendo el flujo de datos entre el modelo y la vista.

*Figura 21. Arquitectura MVC*




*Fuente: Wikipedia*

El flujo de interacción en la arquitectura MVC sigue una secuencia clara: el usuario interactúa con la vista, proporcionando entradas como búsquedas de propiedades o registros en el portal. La vista, a su vez, envía estas solicitudes al controlador. El controlador procesa las solicitudes, interactúa con el modelo para recuperar o actualizar los datos necesarios y luego actualiza la vista en consecuencia. Si hay cambios en los datos, el modelo notifica a la vista para que presente la información actualizada al usuario.

Este patrón de diseño asegura una clara separación de responsabilidades, facilitando el mantenimiento y escalabilidad del sistema. Utilizando Symfony 7.0, se implementa este patrón de manera eficiente, asegurando un desarrollo ordenado y una fácil gestión del portal inmobiliario. La modularidad y la organización proporcionada por MVC permiten que cada componente de la aplicación se desarrolle y mantenga de manera independiente, mejorando la colaboración y la calidad del código.

## 4.2 Tecnología utilizada

Para el desarrollo del portal inmobiliario dirigido a la comunidad universitaria, se han utilizado diversas tecnologías. A continuación, se explica en detalle cada una de estas.

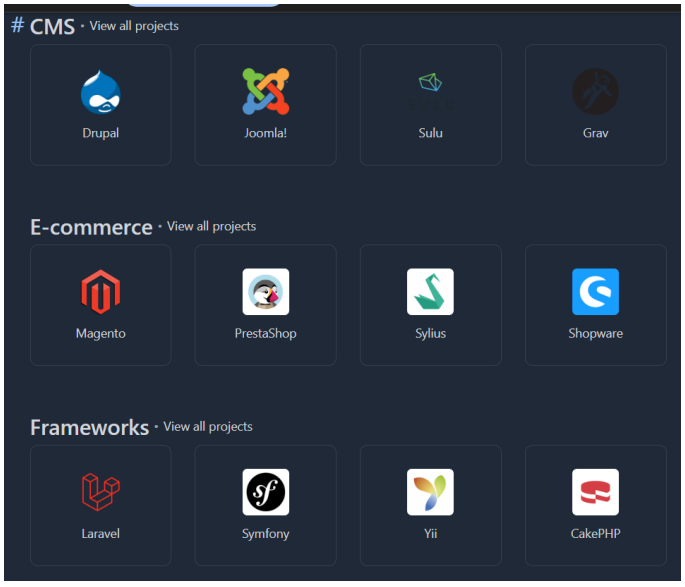
### 4.2.1 Backend

#### *Symfony 7.0*

Symfony comenzó como un proyecto interno de la empresa francesa SensioLabs. Su primera versión pública, Symfony 1.0, fue lanzada en 2007. Desde entonces, el framework ha pasado por múltiples versiones, con importantes mejoras y nuevas funcionalidades introducidas en cada una de ellas. La versión 2.0, lanzada en 2011, marcó un punto de inflexión al reescribir gran parte del núcleo para ser más flexible y modular. Symfony 3.0 y 4.0 continuaron esta tendencia de modernización y optimización. La versión 5.0, lanzada en 2019, se centró en mejorar la eficiencia del código. Symfony 7.0, la versión más reciente, sigue construyendo sobre esta base sólida, introduciendo nuevas características y mejoras de rendimiento.

Symfony es utilizado por numerosas empresas y proyectos de renombre mundial debido a su robustez y flexibilidad. La imagen proporcionada muestra varios de estos proyectos en diferentes categorías.

*Figura 22. Algunos proyectos que utilizan componentes de Symfony*



*Fuente: (Symfony, 2024)*

### *PHP 8.2*

PHP (Hypertext Preprocessor) es un lenguaje de programación de código abierto que se usa ampliamente para el desarrollo de páginas web. Desde su creación en 1995 por Rasmus Lerdorf, PHP ha evolucionado considerablemente, convirtiéndose en una de las principales tecnologías para el desarrollo web. PHP comenzó como un conjunto de scripts escritos en lenguaje C, conocidos como "Personal Home Page Tools". Desde su lanzamiento inicial, ha pasado por numerosas versiones, cada una introduciendo mejoras significativas en términos de funcionalidad, rendimiento y seguridad. (PHP, 2024)

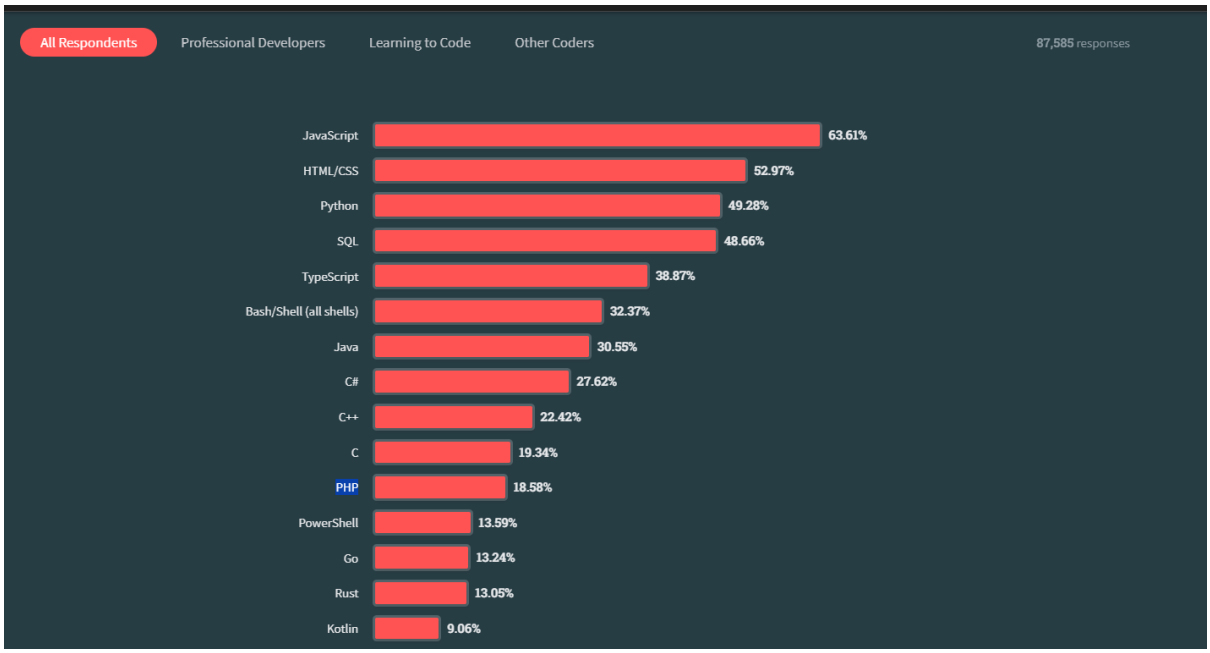
Las versiones 5 y 7 fueron especialmente importantes, con la versión 7.0 ofreciendo una mejora drástica en el rendimiento. La versión 8.0, lanzada en noviembre de 2020, introdujo la compilación JIT (Just-In-Time) y otras características avanzadas. PHP 8.2, la última versión estable, continúa esta tradición de innovación y optimización.

PHP es conocido por su sintaxis simple y la facilidad de aprendizaje, lo que lo hace accesible para desarrolladores principiantes y profesionales. Es utilizado por una gran cantidad de aplicaciones web, incluyendo plataformas populares como WordPress, Facebook y Wikipedia. PHP se puede ejecutar en casi cualquier servidor web y sistema operativo, proporcionando flexibilidad y facilidad de despliegue. La comunidad de PHP y sus desarrolladores principales lanzan actualizaciones periódicas que incluyen mejoras de seguridad, nuevas funcionalidades y optimizaciones de rendimiento.

PHP 8.2 introduce características importantes como la compilación JIT (Just-In-Time), que mejora el rendimiento al compilar partes del código PHP en tiempo de ejecución, y los tipos de unión, que permiten mayor flexibilidad en la definición de tipos de datos. También incluye las expresiones match, que proporcionan una sintaxis más clara y concisa para manejar condicionales, así como varias optimizaciones internas que hacen que las aplicaciones sean más rápidas y eficientes. Además, PHP 8.2 ofrece actualizaciones y mejoras constantes en la seguridad del lenguaje.

La encuesta Stack Overflow de 2023 destaca a PHP entre los lenguajes de programación más utilizados, recopilando datos de desarrolladores de todo el mundo. Se encuentra en undécima posición por encima de otros lenguajes populares como Ruby, Go y Kotlin.

*Figura 23. Tecnologías más populares según Stackoverflow*




*Fuente: (Stackoverflow, 2023)*

Empresas y proyectos de gran envergadura confían en PHP para sus operaciones. Facebook, originalmente construido con PHP, ahora utiliza una versión modificada llamada Hack. Wikipedia, la enciclopedia en línea, utiliza PHP para manejar millones de consultas diarias. WordPress, el CMS más popular del mundo, está basado en PHP y alimenta alrededor del 40% de todos los sitios web. Slack también utiliza PHP para partes de su backend. PHP 8.2, con sus mejoras y características avanzadas, continúa consolidando la posición de PHP como una herramienta esencial para el desarrollo web, ofreciendo un equilibrio entre facilidad de uso, flexibilidad y rendimiento.

### *MariaDB 10*

MariaDB es un gestor de bases de datos relacional (RDBMS) que se originó como fork de MySQL. Fue creado por los desarrolladores originales de MySQL debido a preocupaciones sobre la adquisición de MySQL por parte de Oracle Corporation en 2009. La historia de MariaDB comienza con el objetivo de

mantener un RDBMS verdaderamente abierto y comunitario, asegurando que las mejoras y el desarrollo del software permanezcan accesibles para todos.

El nombre "MariaDB" proviene de la hija de Michael "Monty" Widenius, uno de los fundadores de MySQL y un desarrollador clave en la creación de MariaDB. El lanzamiento inicial de MariaDB se produjo en 2009, casi simultáneamente con la adquisición de Sun Microsystems (y con ello MySQL) por parte de Oracle. Desde su creación, MariaDB ha evolucionado rápidamente, ganando una sólida base de usuarios y contribuyentes. (MariaDB Foundation, 2024)

MariaDB 10.0, lanzado en marzo de 2014, fue una versión clave que marcó una evolución significativa en el proyecto. Esta versión introdujo muchas características nuevas y mejoras de rendimiento que no estaban presentes en MySQL, lo que diferenció aún más a MariaDB como un sistema de bases de datos robusto e independiente. Algunas de las mejoras clave en MariaDB 10.0 incluyen:

- **Replicación global transaction ID (GTID):** proporciona una mejor manera de realizar la replicación de datos, mejorando la flexibilidad y la gestión en entornos distribuidos.
- **Multi-source replication:** permite la replicación de múltiples maestros en un solo esclavo, facilitando la consolidación de datos de varias fuentes.
- **Storage engines avanzados:** inclusión de motores de almacenamiento como Aria y CONNECT, que proporcionan capacidades adicionales y flexibilidad en el manejo de datos.
- **Mejoras en el rendimiento:** varias optimizaciones internas que mejoraron significativamente la velocidad y eficiencia del sistema.

### *DBeaver*

DBeaver es una aplicación de administración de bases de datos que se ha ganado una reputación destacada por su capacidad para soportar una extensa variedad de gestores de bases de datos (DBMS)

Entre las características más destacadas de DBeaver se encuentran:

- **Interfaz de usuario amigable:** DBeaver ofrece una interfaz gráfica intuitiva que facilita la navegación y la administración de bases de datos. Su diseño está orientado a mejorar la productividad de los usuarios.
- **Soporte Multi-DB:** tiene soporte de gestión de varias bases de datos desde una sola aplicación, lo que es especialmente útil para desarrolladores y administradores que trabajan con diversos sistemas de bases de datos.
- **Funciones avanzadas de SQL:** incluye un editor SQL con resaltado de sintaxis, autocompletado de código, formateo de consultas y análisis de dependencias, que ayudan a escribir y optimizar consultas SQL de manera eficiente.
- **Diagramas ER:** DBeaver permite la generación de diagramas entidad-relación (ER), lo que facilita la visualización y comprensión de la estructura de las bases de datos.
- **Conectividad extensa:** soporte para conectarse a bases de datos locales y remotas, proporcionando flexibilidad en la gestión de entornos de desarrollo y producción.
- **Exportación e importación de datos:** ofrece herramientas para exportar e importar datos en varios formatos, facilitando la transferencia y migración de datos entre diferentes sistemas.

DBeaver ha ganado popularidad en la comunidad de desarrolladores y administradores de bases de datos gracias a su robustez y su constante evolución. Es ampliamente utilizado en diversas industrias, incluyendo tecnología, finanzas, educación y salud, debido a su capacidad para manejar tareas complejas de administración de bases de datos de manera eficiente y efectiva.

### 4.2.2 Frontend

#### *Twig*

Twig es un motor de plantillas para PHP desarrollado los creadores de Symfony. Desde su lanzamiento en 2009, Twig se ha convertido en uno de los motores de plantillas más famosos en el ecosistema PHP, conocido por su simplicidad y seguridad. Twig permite a los desarrolladores separar la lógica de

presentación del código de la aplicación, facilitando la creación de vistas dinámicas y manteniendo el código limpio y organizado.

Twig ha sido adoptado por una gran cantidad de desarrolladores y proyectos debido a sus características avanzadas y su facilidad de uso. Es especialmente popular en el ecosistema Symfony, pero también es utilizado por otros frameworks y CMS debido a su versatilidad.

### *Bootstrap 5*

Bootstrap es un framework de CSS desarrollado por Twitter. Desde su lanzamiento inicial en 2011, Bootstrap ha revolucionado el desarrollo frontend al proporcionar un conjunto coherente de herramientas para construir interfaces de usuario responsivas y modernas. (Bootstrap, 2024)

La versión más reciente, Bootstrap 5, continúa esta tradición de simplificación y estandarización del desarrollo web, ofreciendo nuevas funcionalidades y mejoras significativas sobre sus predecesores.

### *jQuery*

jQuery es una biblioteca de JS que simplifica la manipulación del DOM, las animaciones y las solicitudes AJAX. Fue creado por John Resig y lanzado en enero de 2006. Desde entonces, jQuery es una de las bibliotecas de JavaScript más populares en el desarrollo web, debido a su simplicidad y su capacidad para resolver muchas de las dificultades asociadas con la manipulación del DOM y la compatibilidad entre navegadores.

jQuery ha sido adoptado por multitud de sitios web y sigue siendo una herramienta fundamental en el desarrollo web, a pesar de la creciente popularidad de frameworks modernos como React, Angular y Vue.js. Su facilidad de uso y compatibilidad con navegadores lo hacen especialmente útil para proyectos que requieren una manipulación del DOM sencilla y eficient

### *VanillaJS*

VanillaJS se refiere al uso de JavaScript puro, sin la ayuda de bibliotecas o frameworks adicionales como jQuery, React, Angular o Vue.js. Es simplemente JavaScript en su forma más básica, sin ninguna abstracción adicional. El término "VanillaJS" es a menudo usado de manera humorística para enfatizar

que no se necesita nada más que JavaScript puro para crear aplicaciones web eficientes y funcionales.

A pesar de la prevalencia de bibliotecas y frameworks, VanillaJS sigue siendo fundamental en el desarrollo web. Muchos desarrolladores prefieren empezar con JavaScript puro antes de decidir si necesitan una biblioteca adicional. La filosofía de "menos es más" es frecuentemente aplicada, evitando la sobrecarga innecesaria de herramientas externas.

#### *HTML5*

HTML5 es la quinta revisión importante del lenguaje HTML (HyperText Markup Language) que se emplea para estructurar y presentar contenido en la web. Es Introducido en octubre de 2014 y representa un avance significativo respecto a las versiones anteriores, incorporando nuevas características y elementos que facilitan el desarrollo de aplicaciones web modernas y dinámicas.

### 4.2.3 Herramientas auxiliares

#### *Git*

Git es un sistema de control de versiones que facilita la gestión del código fuente a lo largo del desarrollo del proyecto. Git fue diseñado para ser rápido, eficiente y capaz de manejar proyectos grandes con facilidad.

Para la realización del proyecto, se utiliza la estrategia conocida como GitFlow que es un modelo de branching (ramificación) para Git propuesto por Vincent Driessen en 2010. Este flujo de trabajo define un conjunto de prácticas para gestionar las ramas y la liberación de versiones de software, haciendo el proceso más estructurado y eficiente. Git Flow utiliza ramas específicas para diferentes propósitos:

- **Main branch (rama principal):** contiene el código en producción, siempre estable.
- **Develop branch (rama de desarrollo):** contiene el código que se prepara para la próxima versión, integrando las nuevas características.
- **Feature branches (ramas de funcionalidad):** cada nueva funcionalidad se desarrolla en una rama separada que se origina en develop y se fusiona de nuevo en develop al completarse.

- **Release branches (ramas de liberación):** se crean a partir de develop cuando se prepara una nueva versión para producción. Permiten realizar pruebas y ajustes finales antes de fusionarse en main.
- **Hotfix branches (ramas de corrección rápida):** se crean a partir de main para corregir errores críticos en producción y se fusionan de nuevo en main y develop.

Git Flow mejora la organización y el manejo de versiones en proyectos grandes, facilitando la colaboración y el mantenimiento de la estabilidad del código.

### *Visual Studio Code*

Visual Studio Code (VSCode) es un editor de código creado por Microsoft que ha conseguido una inmensa popularidad desde su lanzamiento en 2015. Este editor es conocido por su flexibilidad, extensibilidad y su soporte para gran cantidad de lenguajes de programación, convirtiéndose en una aplicación esencial para desarrolladores de todo el mundo.

### *Trello*

Trello es una aplicación de gestión de proyectos basada en la web que utiliza el sistema Kanban para organizar tareas de manera visual e intuitiva. El sistema Kanban es una herramienta de gestión visual del trabajo que fue popularizada por Toyota. Kanban utiliza tarjetas y tableros para representar y gestionar el trabajo en un flujo continuo. Los principios clave de Kanban incluyen visualizar el trabajo y gestionar el flujo de trabajo.

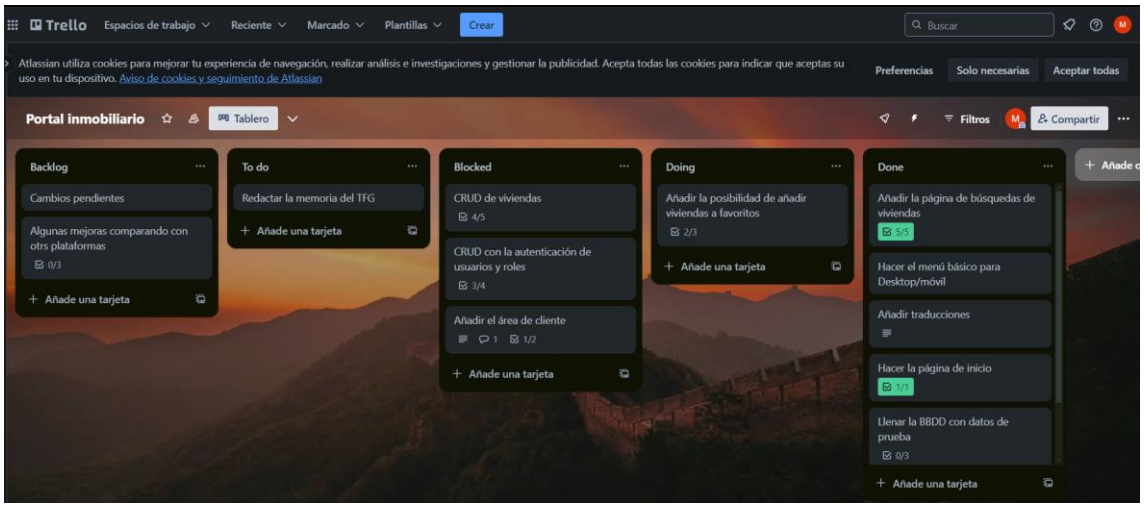
El tablero de Trello para el proyecto del portal inmobiliario dirigido a la comunidad universitaria está organizado utilizando el sistema Kanban, conocido por su efectividad en la gestión visual de tareas. Este tablero lo he dividido en varias columnas que reflejan el estado y progreso de las tareas:

- **Backlog**: la columna Backlog contiene todas las tareas pendientes que aún no han sido priorizadas o asignadas para comenzar. Es una lista de todas las funcionalidades y mejoras planeadas para el futuro.
- **To Do (por hacer)**: incluye las tareas que han sido seleccionadas del backlog y priorizadas para ser trabajadas a continuación. Estas tareas

están listas para ser asignadas a los miembros del equipo y comenzar su desarrollo. Actualmente, no hay tarjetas específicas en esta columna, pero aquí se moverán las tareas del backlog una vez que estén listas para comenzar. Esta columna organiza y prioriza las tareas inmediatas, proporcionando un enfoque claro sobre lo que debe hacerse a continuación.

- **Doing (en progreso)**: contiene las tareas que están actualmente en desarrollo. Los miembros del equipo trabajan activamente en estas tareas. Un ejemplo de tarjeta en esta columna es "Hacer la página de inicio". Esta columna visualiza las tareas en las que el equipo está trabajando en este momento, facilitando el seguimiento del progreso y la identificación de posibles bloqueos.
- **Done (hecho)**: incluye todas las tareas que han sido completadas. Representa el trabajo finalizado y proporciona una visión clara de los logros del equipo. Actualmente, no hay tarjetas específicas en esta columna, pero aquí se moverán las tareas una vez que se hayan completado. Esta columna permite ver y celebrar el trabajo completado, además de asegurar que todas las tareas pasen por una revisión final antes de considerarse terminadas.

*Figura 24. Estructura del Kanban del proyecto*




*Fuente: Espacio del trabajo en Trello para el proyecto*

#### *Xampp*

XAMPP es una solución de servidor web multiplataforma y de código abierto que simplifica el desarrollo y la prueba de aplicaciones web localmente. XAMPP, que significa "X (cualquier sistema operativo), Apache, MySQL, PHP y Perl", fue desarrollado por Apache Friends y es ampliamente utilizado por desarrolladores debido a su facilidad de instalación y configuración.

En el proyecto del portal inmobiliario dirigido a la comunidad universitaria, se ha decidido utilizar XAMPP por varias razones. Primero, XAMPP proporciona un entorno de desarrollo completo y listo para usar, lo que acelera el inicio del proyecto. La facilidad de instalación y configuración permite al equipo concentrarse en el desarrollo del portal sin preocuparse por los detalles técnicos de la configuración del servidor.

Utilizo Windows 11 como sistema operativo principal y, aunque Docker es una herramienta poderosa para la conterización de aplicaciones, he encontrado que Docker con Symfony no funciona del todo bien en Windows debido a problemas de compatibilidad con WSL 2 (Windows Subsystem for Linux). Estos problemas pueden incluir rendimiento lento y errores de configuración que dificultan el desarrollo eficiente.

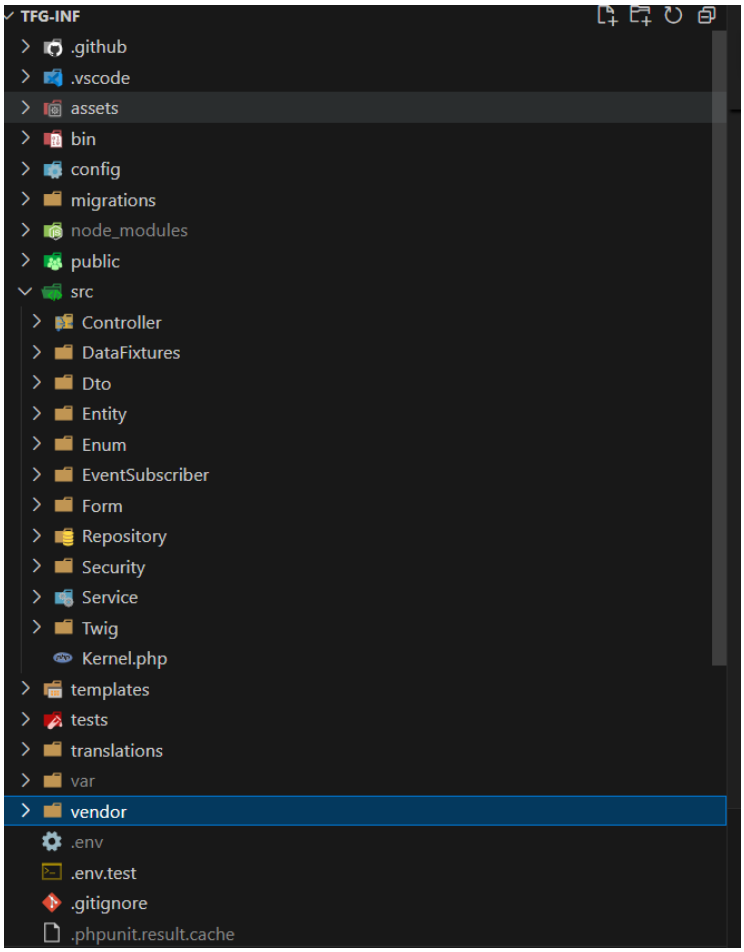
Dado que el rendimiento es una preocupación importante, XAMPP ofrece una solución más estable y rápida en el entorno Windows. La ejecución nativa de Apache, MySQL y PHP en Windows a través de XAMPP elimina las sobrecargas y los problemas asociados con la virtualización de contenedores, proporcionando un entorno de desarrollo más rápido y confiable.

## 4.3 Diseño detallado

### 4.3.1 Estructura de proyecto

El diseño detallado del sistema incluye la estructura de directorios del framework Symfony 7.0, cada uno con un propósito específico para organizar y gestionar el código de manera eficiente.

*Figura 25. Estructura de carpetas del proyecto*




*Fuente: Elaboración propia*

- **src/**: La carpeta src (source) contiene el código fuente principal de mi aplicación Symfony. Dentro de esta carpeta, se encuentran varias subcarpetas que organizan el código según diferentes responsabilidades.
    - **Controller**: almacena los controladores que gestionan las solicitudes HTTP y coordinan las respuestas. Los controladores actúan como intermediarios entre el modelo y la vista.
    - **DataFixtures:** Los fixtures son útiles para poblar tu base de datos con datos iniciales o de prueba durante el desarrollo.
    - **Dto (data transfer object):** contiene objetos de transferencia de datos que se utilizan para transportar datos entre diferentes capas de la aplicación.

    - **Entity**: define las entidades de la base de datos utilizando Doctrine ORM. Las entidades representan las tablas de la base de datos y las interrelaciones entre ellas.
    - **Enum:** enumeraciones que definen un conjunto de constantes relacionadas. Las enumeraciones son útiles para representar valores finitos y bien definidos en la aplicación, como por ejemplo el rol de usuario o tipo de cama en la habitación.
    - **EventSubscriber:** suscriptores de eventos que escuchan y responden a eventos dentro de tu aplicación. Además, en este proyecto, una de las funciones de los suscriptores de eventos es implementar el soporte multilingüe, escuchando eventos relacionados con el cambio de idioma y aplicando la localización adecuada en toda la aplicación.
    - **Repository**: clases que realizan peticiones a la base de datos. Los repositorios se encargan de recuperar y persistir datos en la base de datos de manera eficiente.
    - **Security:** clases y configuraciones relacionadas con la seguridad de tu aplicación, como autenticación y autorización.
    - **Service:** servicios que encapsulan la lógica de negocio de mi aplicación. Por ejemplo, tiene el código necesaro para sacar los datos de la dirección de la vivienda a partir de un identificador único de Google.
    - **Form**: los formularios utilizados en la aplicación. Estos formularios son usados para validar y procesar los datos enviados por los usuarios.
    - **Service**: servicios que encapsulan la lógica de negocio. Los servicios permiten reutilizar y organizar la lógica de negocio de manera modular.
    - **Twig**: extensiones de Twig, por ejemplo, contiene la lógica para calcular y traducir el tiempo transcurrido desde la creación del anuncio en formato amigable para el usuario
- **config/**: archivos de configuración de la aplicación.
- **public/**: archivos públicos accesibles directamente (CSS, JS, imágenes).
- **images/**: imágenes utilizadas en la aplicación.

- **migrations/**: archivos de migración para la gestión de cambios en la base de datos.
- **assets/**: carpeta donde se almacenan los recursos estáticos que serán procesados y empaquetados para el frontend.
    - **styles/**: archivos CSS sin procesar, que pueden ser compilados y minificados.
    - **js/**: archivos JavaScript sin procesar. Aquí se colocan los scripts que luego serán compilados y minificados.
    - **images/**: imágenes originales antes de ser optimizadas y empaquetadas para la producción.

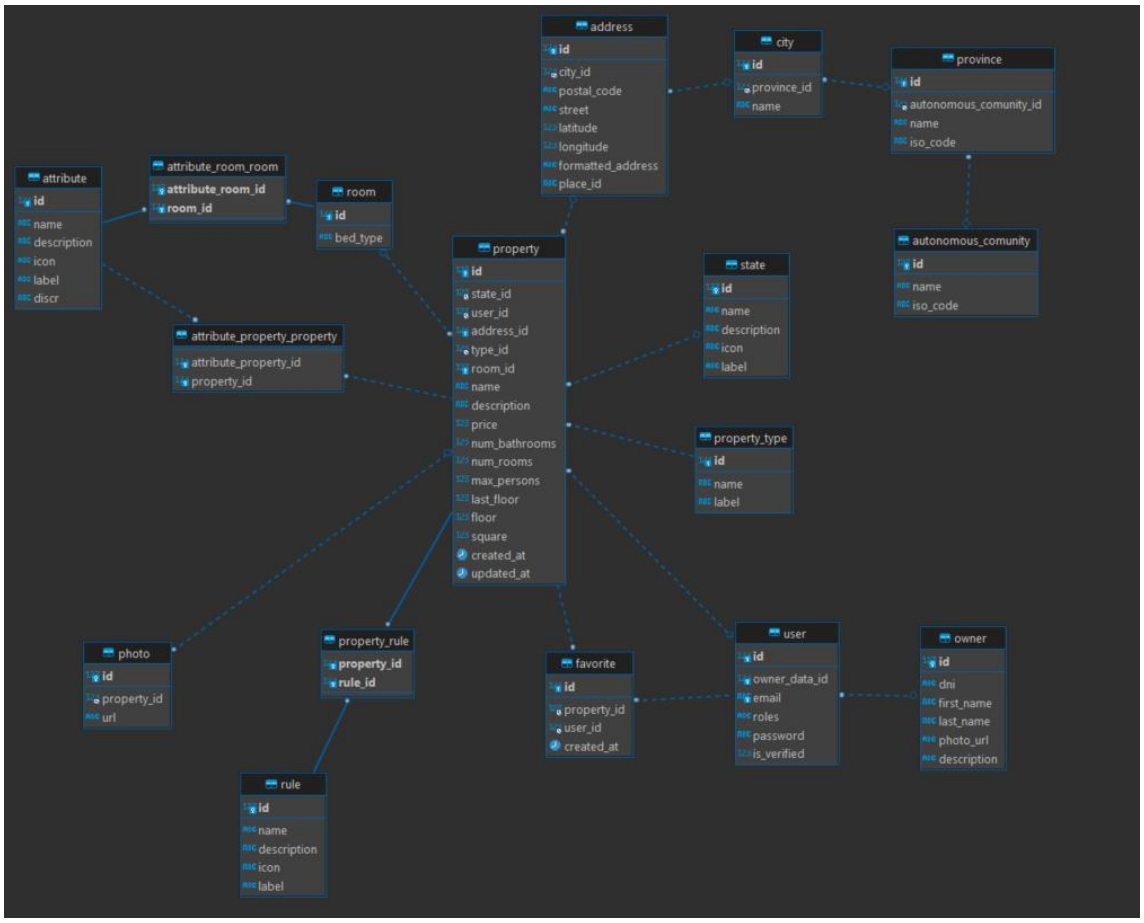
Esta estructura modular facilita el desarrollo colaborativo y el mantenimiento del código, permitiendo que cada desarrollador trabaje en áreas específicas de la aplicación sin interferir en otros componentes.

### 4.3.2 Modelo de datos

Un modelo Entidad-Relación (ER) es una visualización gráfica que ilustra las entidades relevantes dentro de un dominio de negocio y las relaciones entre ellas. Este modelo es importante en el diseño de bases de datos relacionales, ya que ayuda a estructurar y organizar los datos de manera lógica y eficiente. Utilizar un modelo ER permite visualizar y planificar cómo se almacenarán y accederán a los datos, facilitando la comprensión y comunicación del diseño de la base de datos para desarrolladores y para otros interesados en el proyecto.

La imagen proporcionada muestra el modelo ER de la base de datos del portal inmobiliario. A continuación, se explica cada una de las entidades y sus relaciones:

*Figura 26. Diagrama ER de la base de datos*




*Fuente: DBeaver*

- **property** (propiedad): entidad principal que representa una propiedad en el portal.
    - **id**: identificador único de la propiedad.
    - **state_id**: relación con el estado de la propiedad.
    - **user_id**: referencia al usuario (propietario) que creó la propiedad.
    - **address_id**: relación con la dirección de la propiedad.
    - **type_id**: referencia al tipo de propiedad.
    - **room_id**: referencia al tipo de habitación asociada a la propiedad.
    - **description**: descripción de la propiedad.
    - **price**: precio de alquiler.
    - **num_bathrooms**: número de baños.
    - **num_rooms**: número de habitaciones.
    - **max_persons**: número máximo de personas.
    - **last_floor:** indica si está en el último piso.

    - **floor**: número de planta.
    - **square**: metros cuadrados.
    - **created_at:** fecha de creación de la propiedad.
    - **updated_at**: fecha de la última actualización de la propiedad.
- **address** (dirección): contiene la información de la dirección de una propiedad.
    - **id**: identificador único de la dirección.
    - **city_id**: relación con la ciudad.
    - **postal_code**: código postal.
    - **street**: calle.
    - **latitude**: latitud geográfica.
    - **longitude**: longitud geográfica.
    - **formatted_address**: dirección formateada.
    - **place_id**: identificador del lugar (puede ser un identificador externo, como de Google Places).
- **city** (ciudad): información sobre las ciudades.
    - **id**: identificador único de la ciudad.
    - **province_id**: relación con la provincia.
    - **name**: nombre de la ciudad.
    - **code**: código de la ciudad.
- **province** (provincia): información sobre las provincias.
    - **id**: identificador único de la provincia.
    - **autonomous_comunity_id**: relación con la comunidad autónoma.
    - **name**: nombre de la provincia.
    - **iso_code:** código ISO de la provincia.
- **autonomous_comunity** (comunidad autónoma): información sobre las comunidades autónomas.
    - **id**: identificador único de la comunidad autónoma.
    - **name**: nombre de la comunidad autónoma.
    - **iso_code**: código ISO de la comunidad autónoma.
- **state** (estado): describe el estado de una propiedad.
    - **id**: identificador único del estado.

    - **name:** nombre del estado.
    - **description:** descripción del estado.
    - **icon:** icono representativo del estado.
    - **label:** Etiqueta del estado.
- **property_type** (tipo de propiedad): los tipos de la propiedad como habitación o piso entero.
    - **id**: identificador único del equipamiento.
    - **name**: nombre del equipamiento.
    - **label:** etiqueta del estado.
- **Attribute:** (características): características de la propiedad
    - **id**: identificador único del atributo.
    - **name**: nombre del atributo.
    - **description**: descripción del atributo.
    - **icon**: icono representativo del atributo.
    - **label**: etiqueta del atributo.
- **attribute_property_property** (características del piso)
    - **attribute_property_id**: identificador del atributo asociado a la propiedad.
    - **property_id**: identificador de la propiedad asociada al atributo.
- **attribute_room_room** (características de la habitación)
    - **attribute_room_id**: identificador del atributo asociado a la habitación.
    - **room_id**: identificador de la habitación asociada al atributo.
- **room** (propiedades propias de la habitación)
    - **id**: identificador único de la habitación.
    - **bed_type**: tipo de cama de la habitación.
- **photo** (fotos): información sobre archivos relacionados con las propiedades.
    - **id**: identificador único del archivo.
    - **property_id**: relación con la propiedad.
    - **url**: url del archivo.
- **rule** (regla): reglas o normas aplicables a las propiedades.
    - **id**: identificador único de la regla.

    - **name**: nombre de la regla.
    - **description**: descripción de la regla.
    - **icon**: ícono representativo de la regla.
    - **label:** etiqueta de la regla.
- **property_rule**: relación muchos a muchos entre propiedades y reglas.
    - **rule_id**: identificador de la regla.
    - **property_id**: identificador de la propiedad.
- **favorite:** propiedades que usuarios añaden a favoritos
    - **id**: identificador único del favorito.
    - **user_id**: identificador del usuario que marcó la propiedad como favorita.
    - **property_id**: identificador de la propiedad marcada como favorita.
    - **created_at**: fecha en que se marcó como favorita.
- **user:**
    - **id**: identificador único del usuario.
    - **owner_data_id**: referencia a los datos del propietario.
    - **email**: correo electrónico del usuario.
    - **roles**: roles del usuario (array de roles).
    - **password**: contraseña del usuario.
    - **is_verified**: indica si el usuario ha verificado su cuenta.
- **owner:**
    - **id**: identificador único del propietario.
    - **dni**: documento nacional de identidad del propietario.
    - **first_name**: primer nombre del propietario.
    - **last_name**: apellido del propietario.
    - **photo_url**: url de la foto del propietario.
    - **description**: descripción del propietario.

Este esquema organiza la información de manera eficiente, permitiendo la gestión detallada de propiedades, ubicaciones, equipamientos y reglas. La estructura relacional facilita consultas complejas y garantiza la integridad de los datos, cruciales para el correcto funcionamiento del portal inmobiliario.

# 5. Desarrollo de la solución propuesta

Para la realización de este TFG se han seguido los consejos que ofrece la comunidad de Symfony en su documentación. Como parte de las buenas prácticas, se ha utilizado Symfony CLI y la consola de Symfony con los comandos que ayudan a evitar errores humanos y aumentar la productividad en general.

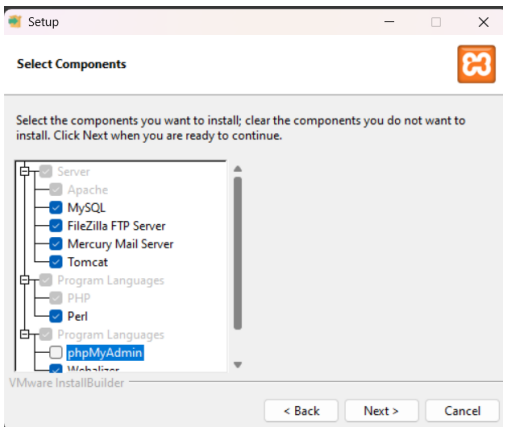
En las secciones siguientes se detallará el desarrollo algunos componentes de la app que considero más importantes incluyendo capturas de pantalla de fragmentos de código y los comandos utilizados para ilustrar el proceso.

## 5.1 Fase inicial

### 5.1.1 Instalación de Xampp

La fase inicial para el desarrollo de cualquier proyecto incluido este es la configuración del entorno local para el trabajo, por lo que descargamos Xampp con php 8.2 y los componentes básicos para el funcionamiento de la app.

*Figura 27. Instalador de Xampp para windows*

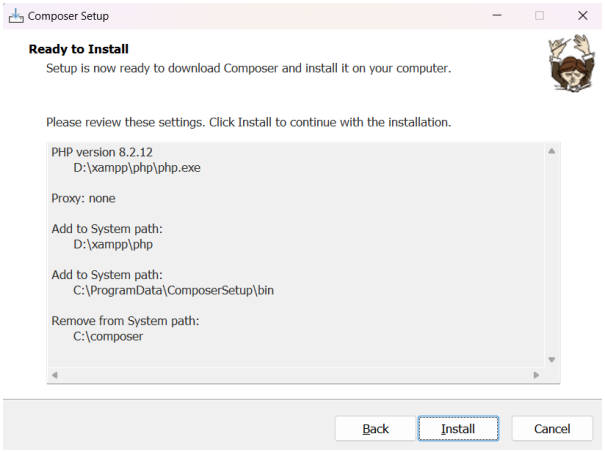



*Fuente: Instalador de Xampp*

### 5.1.2 Instalación de Composer

Composer es una herramienta fundamental en el ecosistema PHP, especialmente cuando se trabaja con frameworks modernos como Symfony, por lo que es necesario instalarlo en entorno local.

*Figura 28. Instalador de Composer para windows*




*Fuente: Instalador de Xampp*

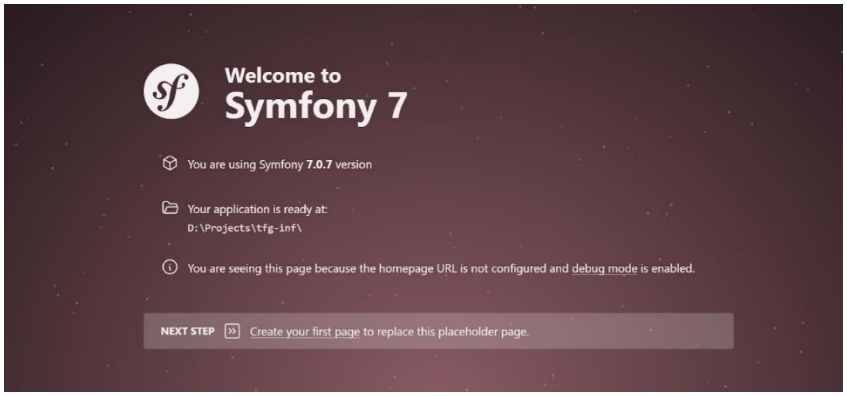
### 5.1.3 Instalación de Symfony CLI

Symfony CLI (Command Line Interface) es una herramienta que facilita una variedad de tareas relacionadas con el desarrollo y el mantenimiento de aplicaciones basadas en Symfony. Utilizaremos esta herramienta para crear el proyecto Symfony e iniciar el servidor web local optimizado.

Lanzamos los siguientes comandos para iniciar un nuevo proyecto e iniciamos el servidor localmente

- *scoop install symfony-cli  # Instalamos symfony-cli*
- *symfony new tfg-inf --version="7.0.*" –webapp  # Creamos nuevo proyecto*
- *symfony server:start  # Iniciamos el servidor localmente*

*Figura 29. La página por defecto que crea Symfony*




*Fuente: Elaboración propia a partir de Symfony CLI*

## 5.2 Fase de desarrollo

En esta sección, se describe el desarrollo del proyecto, centrándose especialmente en la lógica para la gestión de viviendas ya que es una de las entidades principales del proyecto.

### 5.2.1 Definición de las entidades

En primer lugar, necesitamos definir las entidades para persistirlos en la base de. Utilizaremos Doctrine, el ORM (Object-Relational Mapper) de Symfony, para este propósito, creamos la entidad con el comando explicado anteriormente:

- php bin/console make:entity Property

Este comando iniciará un ayudante interactivo en la terminal que solicitará los atributos de la entidad, como descripción, precio, número de habitaciones, entre otros.

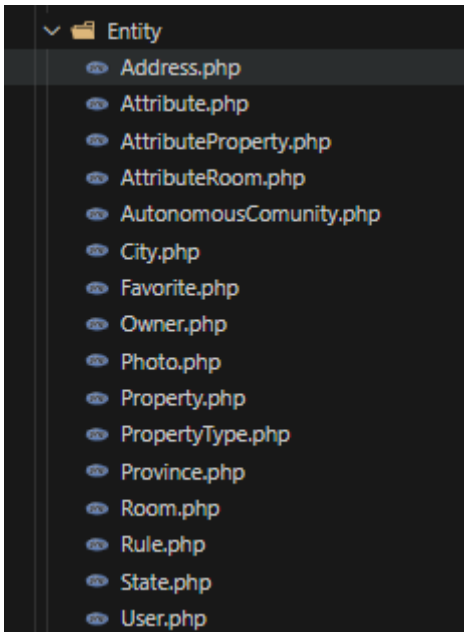
*Figura 30. La entidad de Viviendas de la aplicación*

```
Entity > Property.php > Property
#[ORM\Entity(repositoryClass: PropertyRepository::class)]
#[HasLifecycleCallbacks]
class Property
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    private ?int $id = null;

    #[ORM\Column(type: Types::TEXT, nullable: true)]
    private ?string $description = null;

    #[ORM\Column(nullable: true)]
    private ?float $price = null;

    #[ORM\Column(nullable: true)]
    private ?int $numBathrooms = null;

    #[ORM\Column(nullable: true)]
    private ?int $numRooms = null;

    #[ORM\Column(nullable: true)]
    private ?bool $lastFloor = null;      You, 3 weeks ago • rename lastPlant to lastFloor and fix property …

    #[ORM\Column(nullable: true)]
    private ?int $floor = null;

    #[ORM\Column(nullable: true)]
    private ?int $square = null;

    #[ORM\Column(type: 'datetime', nullable: true)]
    private DateTimeInterface $createdAt;

    #[ORM\Column(type: 'datetime', nullable: true)]
    private DateTimeInterface $updatedAt;

    #[ORM\ManyToOne(inversedBy: 'properties')]
    private ?State $state = null;

    #[ORM\ManyToOne(inversedBy: 'properties')]
    private ?User $user = null;

    /**
     * @var Collection<int, Rule>
     */
    #[ORM\ManyToMany(targetEntity: Rule::class, inversedBy: 'properties')]
    private Collection $rules;

    #[ORM\OneToOne(inversedBy: 'property', cascade: ['persist', 'remove'], orphanRemoval: true)]
    #[ORM\JoinColumn(nullable: true)]
    private ?Address $address = null;

    /**
     * @var Collection<int, Photo>
     */
    #[ORM\OneToMany(targetEntity: Photo::class, mappedBy: 'property', cascade: ['persist', 'remove'])]
    private Collection $photos;
```

*Fuente: Elaboración propia*

Esta herramienta también permite crear otras entidades auxiliares de las viviendas y las relaciones con esas entidades.

*Figura 31. Todas las entidades del modelo*




*Fuente: Elaboración propia*

Las entidades en Symfony no solo representan datos en el código, sino que también pueden usarse para crear automáticamente las tablas correspondientes en la base de datos. Para ello, se usa el comando

- php bin/console doctrine:schema:update -force

### 5.2.2 Elaboración del CRUD de las viviendas

En esta sección se ha detallado cómo se ha desarrollado el CRUD de viviendas utilizando las herramientas proporcionadas por el framework.

Utilizamos el siguiente comando para elaboración del crud

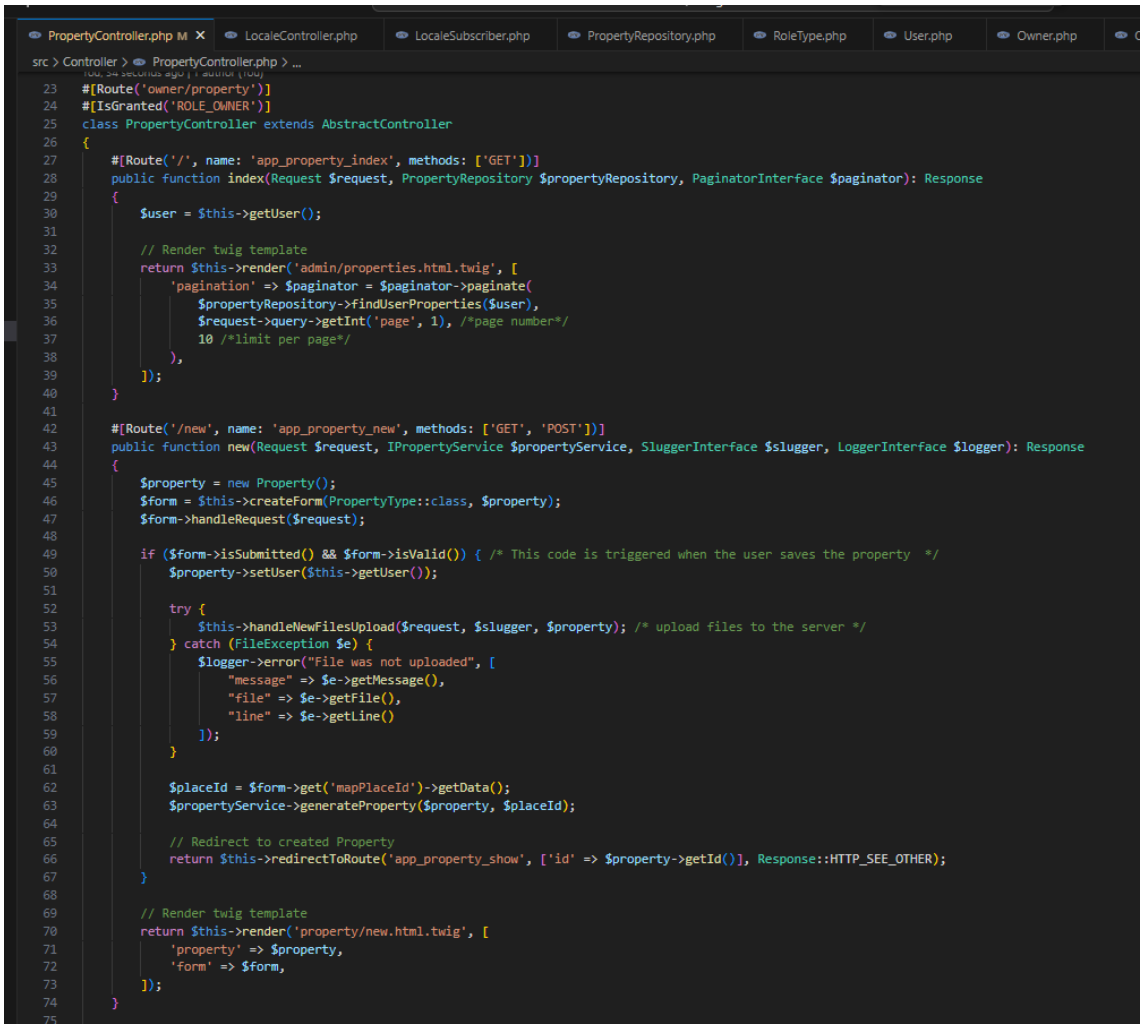
- php bin/console make:crud Property

Este comando creará unas plantillas que posteriormente modificaremos a nuestro gusto, concretamente va a crear:

- Un controlador con las acciones http necesarias para el CRUD (GET, POST, PUT)
- Formularios que ayudan a visualizar los datos persistidos en el modelo en los inputs adecuados
- Plantillas Twig que contienen el código del frontend de la aplicación

A continuación, se adjunta la imagen con los endpoints para crear nuevas viviendas y para listarlas en el panel de administrador del usuario.

*Figura 32. El controlador de las viviendas*




*Fuente: Elaboración propia*

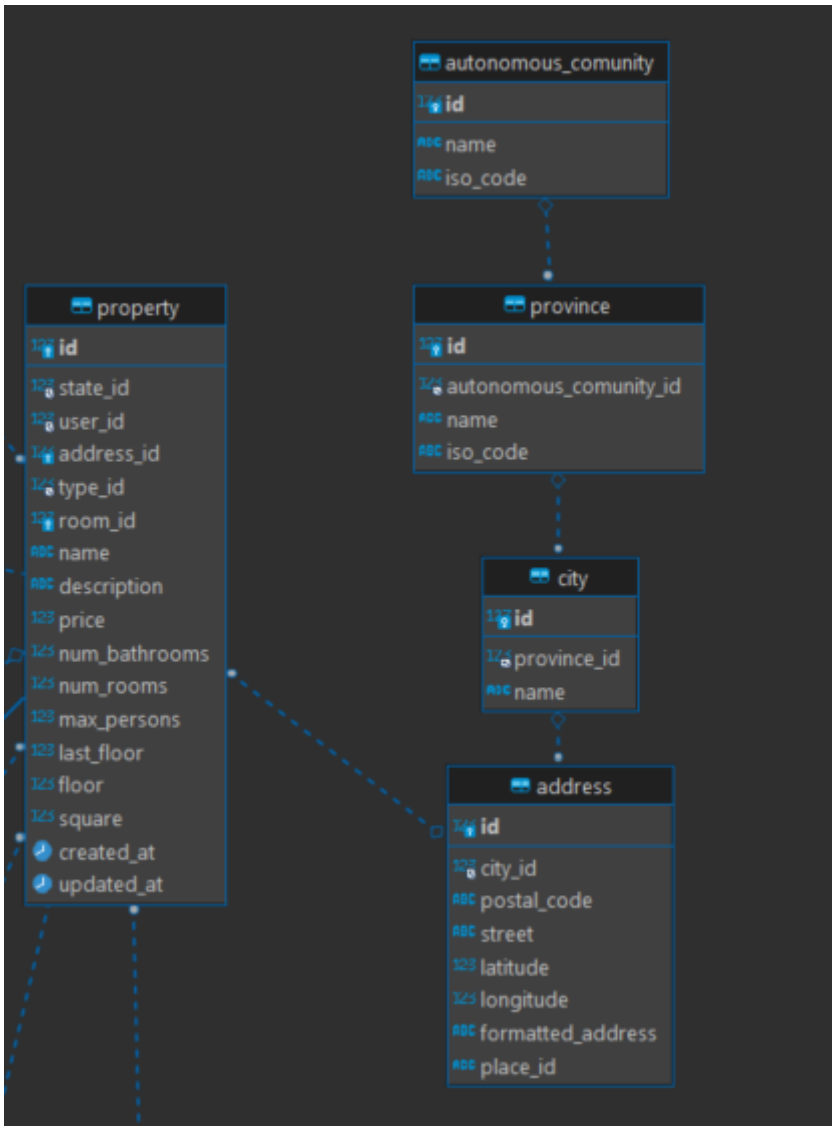
### *Guardado de la ubicación de usuarios a partir de un id de lugar de Google*

Según Instituto Geográfico Nacional la organización territorial de España consta de tres niveles (Instituto Geográfico Nacional, s.f.):

- El Estado
- Las comunidades autónomas
- Las entidades locales

Como consecuencia de esto, para identificar correctamente la ubicación de las viviendas tenemos varias tablas en nuestra base de datos uno para la dirección (Address) y otras tres para la Comunidad autónoma, Provincia y las entidades locales.

*Figura 33. Las tablas de la Base de datos relacionados con la ubicación*



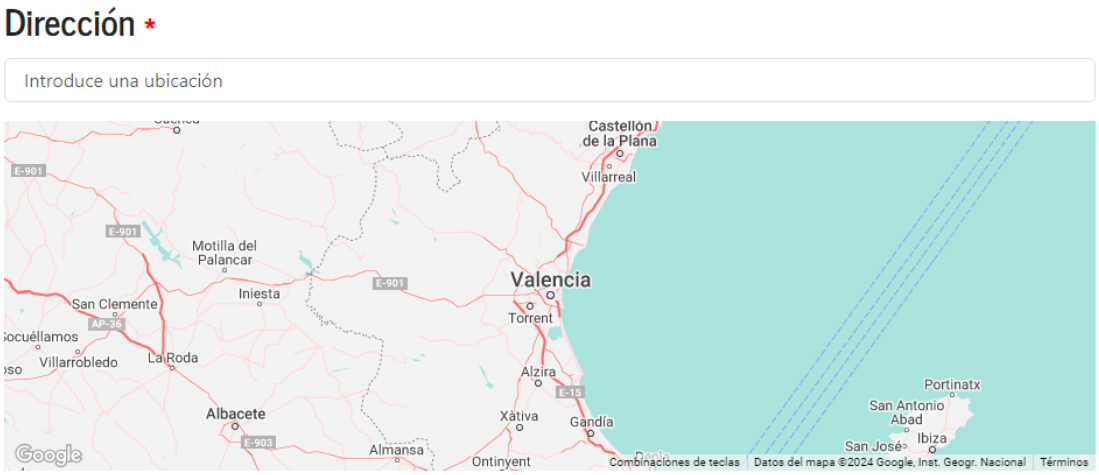

*Fuente: Elaboración propia*

La solución más simple es hacer 3 desplegables en la vista de creación de viviendas y además añadir un input donde el usuario introduzca la dirección completa de la vivienda. Esta solución incrementa las posibilidades de hacer un error en la introducción de datos y empeora la experiencia del usuario por lo tanto tenemos que simplificarlo para el usuario de la siguiente forma.

1. **Integración del Campo de Texto con Autorellenado en el Frontend:**

a. En el frontend del portal inmobiliario, se ha incorporado un campo de texto que utiliza la funcionalidad de Autocomplete de la API de Google Places.

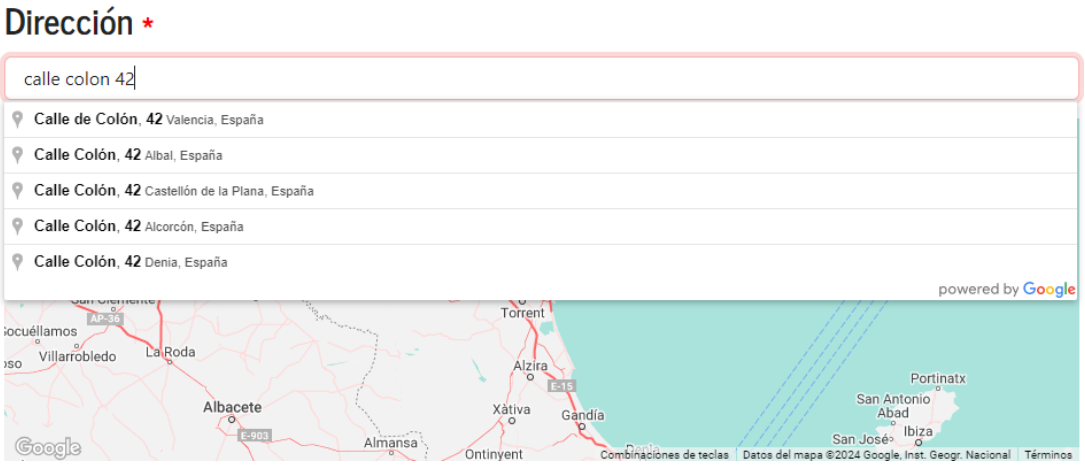
*Figura 34. La dirección de la vivienda en el formulario para crear una nueva vivienda*




*Fuente: Elaboración propia*

Mientras el usuario escribe en el campo, se muestran sugerencias de lugares en tiempo real, proporcionadas por la API de Google.

*Figura 35. Cambio de dirección de la vivienda en el formulario para crear/editar una vivienda*



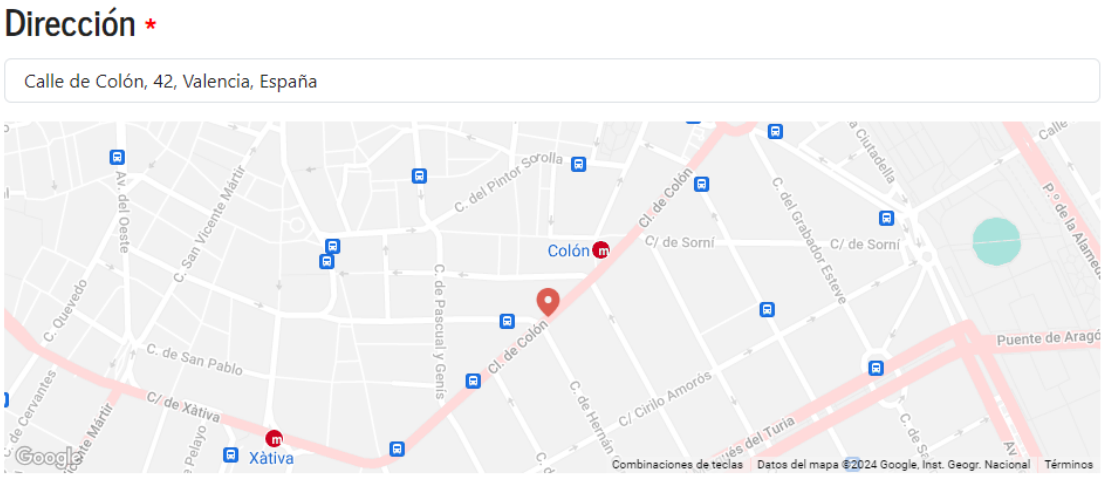

*Fuente: Elaboración propia*

2. **Selección de lugar y obtención del placeId:**

   a. Cuando el usuario selecciona una sugerencia, se captura el identificador único del lugar (placeId), proporcionado por Google.

b. Este placeId es una cadena única que identifica un lugar específico y se utiliza para obtener detalles adicionales sobre ese lugar.

*Figura 36. Selección de dirección de la vivienda en el formulario para crear/editar una vivienda*



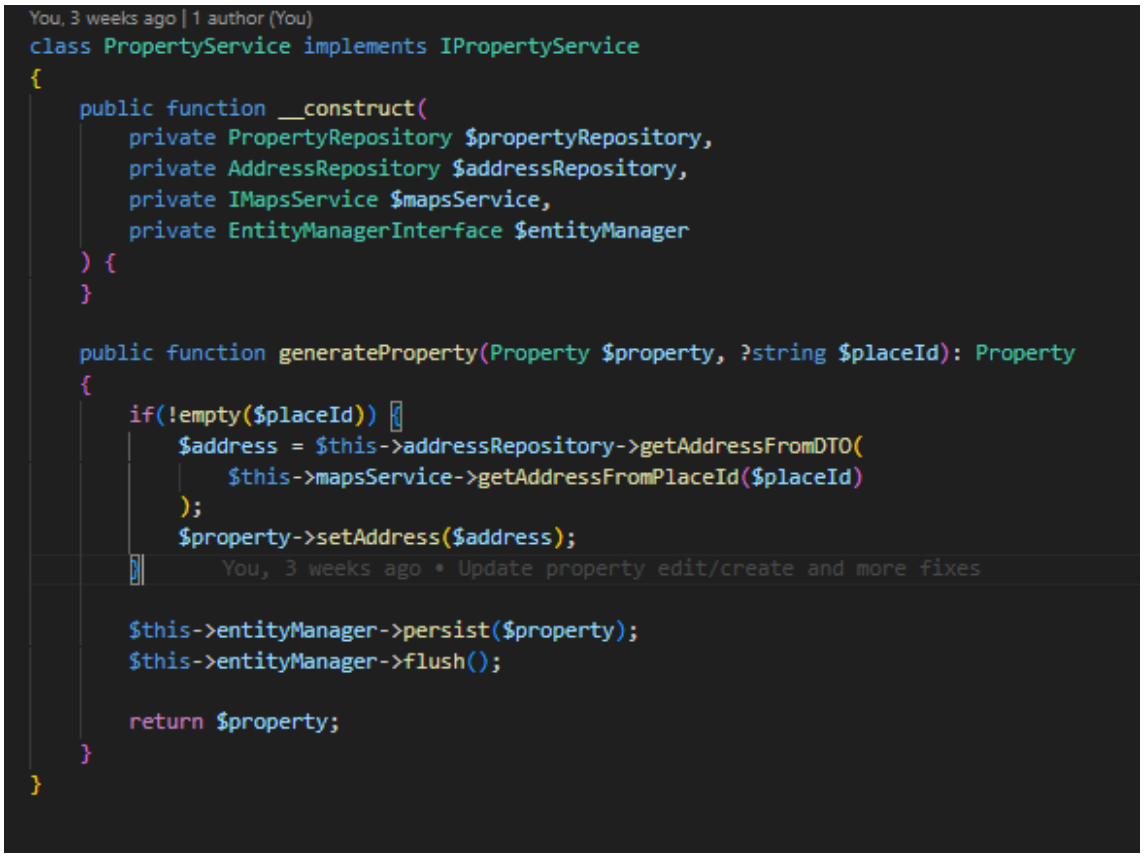

*Fuente: Elaboración propia*

3. **Envío del placeId al backend:**

    a. El placeId seleccionado se envía desde el frontend al backend del portal mediante una solicitud HTTP cuando el formulario se envia al backend

    b. El backend recibe este placeId para proceder con la obtención de detalles adicionales.

*Figura 37. Código para generar dirección de la vivienda*



```
You, 3 weeks ago | 1 author (You)
class PropertyService implements IPropertyService
{
    public function __construct(
        private PropertyRepository $propertyRepository,
        private AddressRepository $addressRepository,
        private IMapsService $mapsService,
        private EntityManagerInterface $entityManager
    ) {
    }

    public function generateProperty(Property $property, ?string $placeId): Property
    {
        if(!empty($placeId)) {
            $address = $this->addressRepository->getAddressFromDTO(
                $this->mapsService->getAddressFromPlaceId($placeId)
            );
            $property->setAddress($address);
        }       You, 3 weeks ago • Update property edit/create and more fixes

        $this->entityManager->persist($property);
        $this->entityManager->flush();

        return $property;
    }
}
```

*Fuente: Elaboración propia*

4. **Llamada a la API de Google Places desde el Backend:**

    a. En el backend, se realiza una llamada a la API de Google Places. Esta llamada se realiza mediante una solicitud HTTP a la API de Google, enviando el placeId y la clave de API correspondiente.

*Figura 38. El servicio encargado de recuperar un DTO con campos necesarios de la dirección desde un placeId de google*

```php
class GoogleMapsService implements IMapsService

    public function __construct(
        private HttpClientInterface $client,
        private ParameterBagInterface $params
    ) {
        $this->baseUrl = $params->get('google_api')['base_url'];
        $this->apiKey = $params->get('google_api')['api_key'];
    }

    public function getAddressFromPlaceId(string $placeId) : AddressDTO
    {
        $result = $this->fetchPlaceDetails($placeId);       // Fetch place details from google api

        return $this->buildPlaceAddress($result['result']); // build and return AddressDTO from result
    }


    private function fetchPlaceDetails(string $placeId) {
        $response = $this->client->request(
            'GET',
            sprintf("%s/maps/api/place/details/json?key=%s&place_id=%s", $this->baseUrl, $this->apiKey, $placeId)
        );
        $content = json_decode($response->getContent(), true);

        return $content;
    }

    private function buildPlaceAddress(array $result) : AddressDTO
    {
        $address = new AddressDTO();
        $addressComponents = $result['address_components'];
        foreach ($addressComponents as $addressComponent) {
            if(in_array('route', $addressComponent['types'])) {
                $address->setStreet($addressComponent['long_name']);
            }else if(in_array('locality', $addressComponent['types'])) {
                $address->setCity($addressComponent['short_name']);
            }else if(in_array('administrative_area_level_2', $addressComponent['types'])) {
                $address->setProvince($addressComponent['short_name']);      You, 3 weeks ago • Update property edit
            }else if(in_array('administrative_area_level_1', $addressComponent['types'])) {
                $address->setAutoconomousComunity($addressComponent['short_name']);
            }else if(in_array('country', $addressComponent['types'])) {
                $address->setCountry($addressComponent['short_name']);
            }else if(in_array('postal_code', $addressComponent['types'])) {
                $address->setPostalCode($addressComponent['long_name']);
            }
        }

        $address->setLatitude($result['geometry']['location']['lat']);
        $address->setLongitude($result['geometry']['location']['lng']);
        $address->setFormattedAddress($result['formatted_address']);
        $address->setFormattedAddress($result['formatted_address']);
        $address->setPlaceId($result['place_id']);
        return $address;
    }
}
```

*Fuente: Elaboración propia*

5. **Extracción de datos de ubicación:**

    a. La respuesta de la API de Google Places contiene información detallada sobre el lugar, incluyendo datos como la ciudad, comunidad autónoma y provincia.

b. Estos datos se extraen de la respuesta JSON proporcionada por la API.

6. **Procesamiento y almacenamiento de datos:**
    a. Los datos de ubicación extraídos (ciudad, comunidad autónoma, provincia) se procesan y se almacenan en la base de datos del portal inmobiliario.

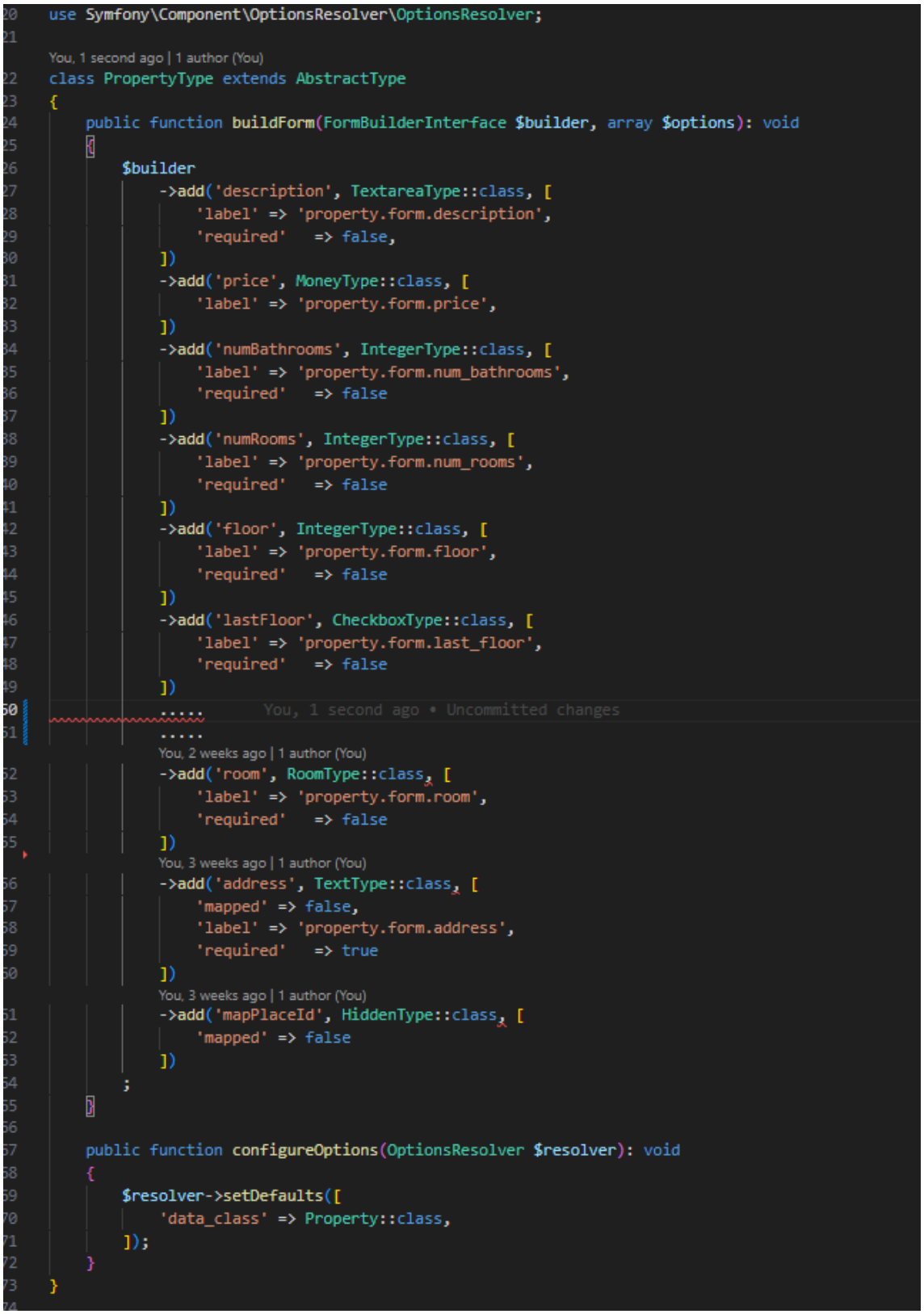
#### *5.2.2.1 Búsqueda de viviendas*

Para la búsqueda de viviendas se decidió crear un formulario específico para los filtros de búsqueda. Este formulario incluye diversos campos que permiten a los usuarios especificar sus criterios de búsqueda, tales como el número de habitaciones, precio y tipo de alojamiento.

### 5.2.3 Formularios

Los formularios de Symfony facilitan la creación, validación y procesamiento de formularios HTML. Resuelven problemas de repetición de código, validación de datos, manejo de datos, generación de HTML, seguridad (protección CSRF) e integración con otros componentes del framework. En la siguiente imagen adjunto el formulario para creación/edición de viviendas en la plataforma.

*Figura 39. Formulario para creación edición de viviendas*

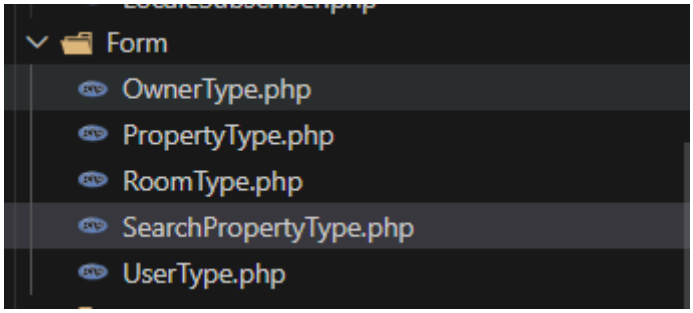


```php
use Symfony\Component\OptionsResolver\OptionsResolver;

You, 1 second ago | 1 author (You)
class PropertyType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options): void
    {
        $builder
            ->add('description', TextareaType::class, [
                'label' => 'property.form.description',
                'required'   => false,
            ])
            ->add('price', MoneyType::class, [
                'label' => 'property.form.price',
            ])
            ->add('numBathrooms', IntegerType::class, [
                'label' => 'property.form.num_bathrooms',
                'required'   => false
            ])
            ->add('numRooms', IntegerType::class, [
                'label' => 'property.form.num_rooms',
                'required'   => false
            ])
            ->add('floor', IntegerType::class, [
                'label' => 'property.form.floor',
                'required'   => false
            ])
            ->add('lastFloor', CheckboxType::class, [
                'label' => 'property.form.last_floor',
                'required'   => false
            ])
            .....       You, 1 second ago • Uncommitted changes
            .....
            You, 2 weeks ago | 1 author (You)
            ->add('room', RoomType::class, [
                'label' => 'property.form.room',
                'required'   => false
            ])
            You, 3 weeks ago | 1 author (You)
            ->add('address', TextType::class, [
                'mapped' => false,
                'label' => 'property.form.address',
                'required'   => true
            ])
            You, 3 weeks ago | 1 author (You)
            ->add('mapPlaceId', HiddenType::class, [
                'mapped' => false
            ])
        ;
    }

    public function configureOptions(OptionsResolver $resolver): void
    {
        $resolver->setDefaults([
            'data_class' => Property::class,
        ]);
    }
}
```

*Fuente: Elaboración propia*

De forma parecida, creamos los formularios para editar/añadir datos de la habitación a la vivienda, para los usuarios, para la búsqueda de propiedades, etc.

*Figura 40. Formularios de symfony en la App*




*Fuente: Elaboración propia*

### 5.2.4 Plantillas twig para renderizar el frontend

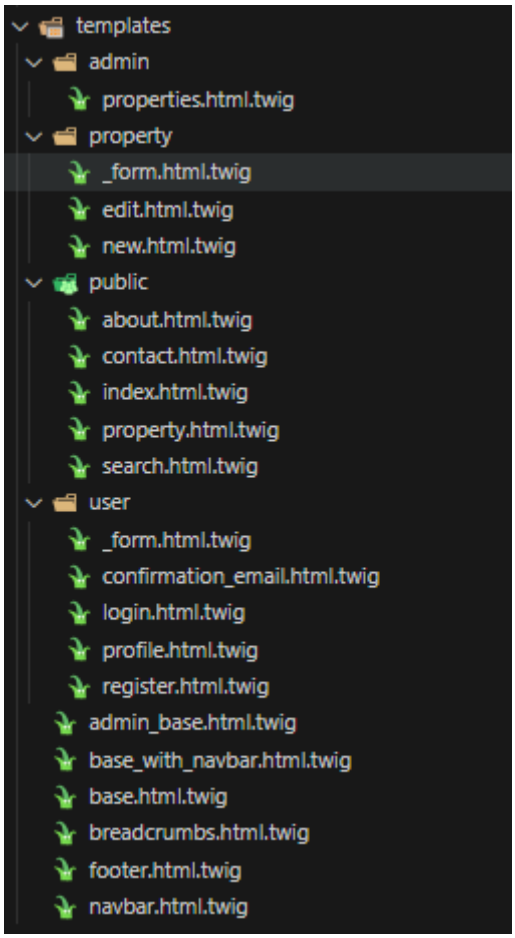
En la siguiente fase modificamos las plantillas de twig aplicándoles los estilos CSS para renderizarlo correctamente en dispositivos de diferentes tamaños.

En la raíz de la carpeta “templates”, están las plantillas base, las otras plantillas heredan o incluyen estás para evitar repetición de código.

- **admin_base.html.twig**: plantilla base para las vistas del administrador. Incluye la barra lateral de administrador.
- **base_with_navbar.html.twig**: plantilla base que incluye una barra de navegación. Utilizada por diversas vistas que requieren navegación superior.
- **base.html.twig**: plantilla base principal. Contiene la estructura HTML básica y bloques que pueden ser extendidos por otras plantillas.
- **breadcrumbs.html.twig**: plantilla para mostrar las migas de pan (breadcrumbs). Ayuda a la navegación mostrando la ubicación actual dentro del sitio.
- **footer.html.twig**: plantilla para el pie de página. Incluye enlaces y información de contacto que se muestra en la parte inferior de todas las páginas.
- **navbar.html.twig**: plantilla para la barra de navegación. Incluye enlaces a las principales secciones del sitio.

En la siguiente imagen aparecen el directorio con todas las plantillas utilizadas en el proyecto.

*Figura 41. Todas las plantillas definidas en el proyecto*




*Fuente: Elaboración propia*

Todas las plantillas tienen una estructura jerárquica que se explica a continuación:

- **base.html.twig:** es la plantilla base principal que contiene la estructura HTML básica de todas las páginas. Otras plantillas pueden extender esta base para mantener una estructura consistente en todo el sitio.
- **navbar.html.twig:** contiene la barra de navegación con enlaces a las principales secciones del sitio.
- **footer.html.twig:** contiene el pie de página con enlaces e información de contacto.
- **base_with_navbar.html.twig:** extiende base.html.twig e incluye navbar y footer para crear una plantilla completa con navegación y pie de

página. Sirve como plantilla base para páginas que requieren navegación y pie de página, asegurando una interfaz consistente. Este es el código de esta plantilla

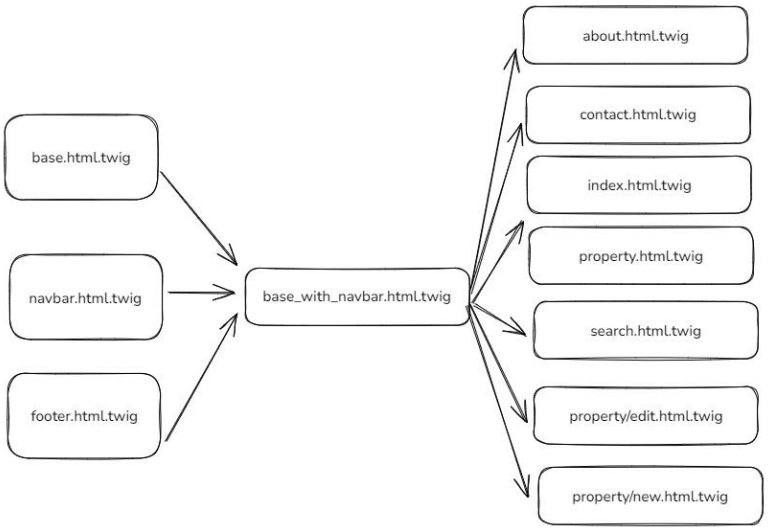
*Figura 42. Código de plantilla base con navbar*

```
You, 2 weeks ago | 1 author (You)
{% extends "base.html.twig" %}
{% block stylesheets %}
    {{ parent() }}
    <link href="{{ asset('css/navbar.css') }}" rel="stylesheet">

{% endblock %}

{% block body %}
    {% include '/navbar.html.twig'%}

    <div class="content">
        {% block content %}
        {% endblock %}
    </div>

    {% include '/footer.html.twig'%}
{% endblock %}       You, 2 weeks ago • Change project structure
```

*Fuente: Elaboración propia*

El esquema completo con la jerarquía de las plantillas disponibles para usuarios no registrados aparece en la siguiente figura:

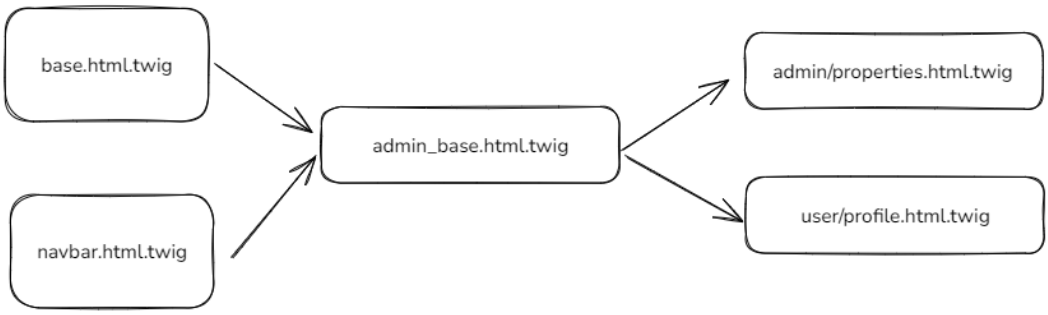
*Figura 43. Jerarquía de plantillas de twig públicas*




*Fuente: Elaboración propia*

La sección administrativa, es ligeramente diferente y tiene su propia platilla llamada admin_base.html.twig, esta vista es necesaria porque la sección administrativa tiene una barra lateral para una mejor navegación en área de clientes. En la siguiente imagen aparece la jerarquía de las plantillas del área de clientes.

*Figura 44. Jerarquía de plantillas de twig para usuarios registrados.*




*Fuente: Elaboración propia en la app excalidraw.com*

## 5.3 Usuarios y autentificación

Para la autenticación de usuarios se utiliza el bundle verify-email-bundle

De Symfony, por lo que el primer paso es instalarlo con el composer con el siguiente comando:

- composer require symfonycasts/verify-email-bundle

Luego configuramos el envío de correos creando el siguiente fichero

*Figura 45. El fichero de configuración para envio de correos*

config > packages > mailer.yaml

You, 3 months ago | 1 author (You)

```yaml
framework:
    mailer:
        dsn: '%env(MAILER_DSN)%'

```

*Fuente: Elaboración propia*

Luego, creamos el formulario para autenticación de usuarios con el siguiente código:

- php bin/console make:registration-form

Este comando crea el controlador para hacer el login, logout de los usuarios y la entidad User para persistirlo en la base de datos. Lo único que falta es cambiar ligeramente la entidad de usuario y cambiar las vistas de twig.

### 5.3.1 Jerarquía de usuarios

En este proyecto, se implementa una jerarquía de roles para gestionar los permisos de los usuarios. Esta jerarquía permite definir diferentes niveles de acceso y responsabilidades dentro de la aplicación, garantizando que los usuarios solo puedan realizar acciones acordes a su rol.

- **Usuario no autentificado:** es el rol más básico para ver las viviendas únicamente.
- **ROLE_USER**: es el rol asignado a todos los usuarios registrados. Tiene permisos limitados y solo puede acceder a las funcionalidades esenciales de la App. Tiene los siguientes permisos:
    - Todos los permisos de usuario no autentificado
    - Acceso al perfil personal.
    - Añadir las viviendas a favoritas.
- **ROLE_OWNER**: este rol se asigna a los propietarios que pueden gestionar sus propias propiedades dentro de la aplicación. Tiene todos los permisos de ROLE_USER, además de permisos adicionales para la gestión de propiedades. Tiene estos permisos:
    - Todos los permisos de ROLE_USER.
    - Crear, editar y eliminar sus propias propiedades.
- **ROLE_ADMIN**: es el rol más alto, asignado a los administradores de la aplicación. Tiene todos los permisos de ROLE_USER y ROLE_OWNER, además de permisos administrativos completos. Tiene estos permisos:
    - Todos los permisos de ROLE_USER y ROLE_OWNER.
    - Gestión de todos los usuarios (crear, editar, eliminar).
    - Gestión de todas las viviendas.

## 5.4 Paginación

La paginación es una funcionalidad importante a largo plazo, ya que permite manejar y visualizar datos de forma más eficiente. Para simplificar el proceso de paginación, se ha decidido usar el KnpPaginatorBundle, que es un paquete que facilita la integración de la paginación en proyectos de Symfony.

En primer lugar, instalamos el plugin con composer con el siguiente comando:

- composer require knplabs/knp-paginator-bundle

Luego adaptamos los controladores para utilizarlo. Esto implica modificar la consulta que obtiene la lista de viviendas para que devuelva un conjunto de datos paginados en lugar de todos los resultados a la vez. KnpPaginatorBundle se encarga de dividir los resultados en páginas y manejar la lógica de navegación entre ellas. Por ejemplo, en la página que muestra las viviendas en la página principal se queda de la siguiente forma:

*Figura 46. Un trozo del PublicController de las propiedades con la paginación*

```
$builder = $propertyRepository->buildFindByCityAndRoomQuery($city, $type, $searchData);
$paginator = $paginator->paginate(
    $builder, /* query NOT result */
    $request->query->getInt('page', 1), /*page number*/
    12 /*limit per page*/
);
```

*Fuente: Elaboración propia*

En la vista, se ajustamos la presentación de los datos para mostrar los resultados paginados. Esto incluye la visualización de la lista de viviendas por página y la implementación de controles de navegación (como enlaces de "anterior" y "siguiente") para permitir a los usuarios moverse entre las diferentes páginas de resultados. El KnpPaginatorBundle proporciona herramientas para facilitar la creación de estos controles de navegación. El código en Twig será el siguiente:

*Figura 47. Código para pegar controles de navegación*

```
<div class="d-flex justify-content-center">
    {{ knp_pagination_render(pagination) }}
</div>
```

*Fuente: Elaboración propia*

## 5.5 Symfony fixtures

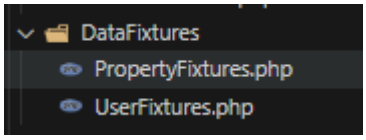
Los fixtures en Symfony son una herramienta muy útil para cargar datos iniciales o de prueba en la base de datos. Usan el paquete DoctrineFixturesBundle y permiten predefinir datos que se pueden insertar en la base de datos de manera automática. Estos fixtures son especialmente útiles en entornos de desarrollo y prueba para simular un entorno de datos realista.

Para cargar esos datos de prueba, simplemente hay que ejecutar el siguiente comando:

- php bin/console doctrine:fixtures:load

En la aplicación se usan los fixtures para crear usuarios y viviendas en la plataforma en entorno local.

*Figura 48. Las fixtures definidas en el proyecto*




*Fuente: Elaboración propia*

La clase PropertyFixtures define y carga viviendas de prueba en entorno local

*Figura 49. Código de la clase PropertyFixtures para inicializar el entorno demo con viviendas*

```php
public function load(ObjectManager $manager): void
{
    for ($i = 1; $i <= 13; $i++) {
        $property = new Property();
        $property->setDescription("<p>Es la descripción larga de pruebas para la propiedad $i</p>");
        $property->setType($this->propertyTypeRepository->findOneBy(['name' => 'room']));
        $property->setNumRooms(rand(1, 6));
        $property->setNumBathrooms(min(rand(1, 3), $property->getNumRooms()));
        $property->setFloor(rand(1, 9));
        $property->setSquare(20 * $property->getNumRooms() + rand(0, 9));
        $property->addAttributeProperty($this->attributePropertyRepository->findOneBy(['name' => 'elevator']));

        $room = new Room();
        $room->addAttributeRoom($this->attributeRoomRepository->findOneBy(['name' => 'desk']));
        $room->addAttributeRoom($this->attributeRoomRepository->findOneBy(['name' => 'chair']));      You, last week • R
        $room->setBedType(BedType::Individual);

        $property->setRoom($room);

        $photo1 = new Photo();
        $imageName1 = $this->findFilesWithPrefix("sample_property_" . $i . ".");
        $photo1->setUrl("images/$imageName1");
        $property->addPhoto($photo1);

        $photo2 = new Photo();
        $imageName2 = $this->findFilesWithPrefix("sample_property_" . $i+1 . ".");
        $photo2->setUrl("images/$imageName2");
        $property->addPhoto($photo2);

        $property->setAddress($this->generateRandomAddress($i));
        $property->setUser($this->userRepository->findOneBy(['email' => 'owner@test.com']));
        $property->setPrice(rand(100, 1000));
        $manager->persist($property);
    }

    $manager->flush();
}
```

*Fuente: Elaboración propia*

## 5.6 Migraciones

Las migraciones en Symfony son una herramienta proporcionada por Doctrine para gestionar cambios en la base de datos de manera controlada y versionada. Permiten definir y aplicar cambios en la estructura de la base de datos (como la creación o modificación de tablas y columnas) mediante scripts de migración.

En este proyecto, las migraciones se usan para precargar datos iniciales críticos en la base de datos, como provincias, comunidades autónomas, características de las habitaciones y características de las viviendas. Estos datos son esenciales tanto en el entorno de preproducción como en el de producción.

*Figura 50. Ejemplo de carga de comunidades autonomas con Migraciones de Doctrine*

```
// Autonomous comunity
$this->addSql(<<<SQL
INSERT INTO `autonomous_comunity` (`id`, `name`, `iso_code`)
    VALUES
    (1,'Comunidad Valenciana', 'VC'),
    (2,'Andalucía', 'AN'),
    (3,'Aragón', 'AR'),
    (4,'Cantabria', 'CB'),
    (5,'Castilla y León', 'CL'),
    (6,'Castilla-La Mancha', 'CM'),
    (7,'Cataluña', 'CT'),
    (8,'Ceuta', 'CE'),
    (9,'Comunidad de Madrid', ''),
    (10,'Extremadura', 'EX'),
    (11,'Galicia', 'GA'),
    (12,'Islas Baleares', 'IB'),
    (13,'Islas Canarias', 'CN'),
    (14,'La Rioja', 'RI'),
    (15,'Melilla', 'ML'),
    (16,'País Vasco', 'PV'),
    (17,'Navarra', 'NC'),
    (18,'Principado de Asturias', 'AS'),
    (19,'Región de Murcia', 'MC');
SQL);
```

*Fuente: Elaboración propia*

## 5.7 Soporte multilingüe

Este proyecto está dirigido a estudiantes universitarios. Este público incluye no solo a estudiantes locales, sino también a una gran cantidad de estudiantes extranjeros, como aquellos que participan en programas de intercambio académico como Erasmus. La diversidad lingüística de los usuarios hace que el soporte multilingüe sea una característica imprescindible.

En primer lugar, tenemos que configurar que el idioma por defecto será castellano.

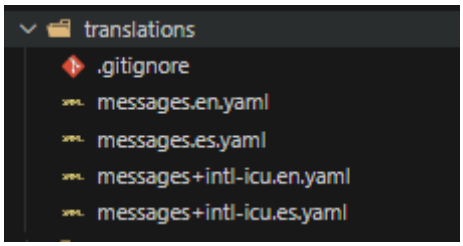
*Figura 51. Configuración de soporte multilingüe en la aplicación*

```
You, 4 weeks ago | 1 author (You)
framework:
    default_locale: es
    translator:
        default_path: '%kernel.project_dir%/translations'
        fallbacks:
            - en
        providers:
```

*Fuente: Elaboración propia*

Una vez hecha la configuración, tenemos que crear los ficheros de traducción para cada idioma:

*Figura 52. Contenido de la carpeta con mensajes de traducción*




*Fuente: Elaboración propia*

El contenido de esos ficheros es simplemente un array clave-valor muy simple.

*Figura 53. Un trozo de código del fichero de traducciones en castellano*

```
You, 11 hours ago | 1 author (You)
form:
  save: Guardar
  edit: Editar
  return: Volver
base:
  brand: UniWave
  search: Buscar
  client_area: Área de clientes
  favorites: Favoritos
  logout: Salir
footer:
  text: Portal Inmobiliario Universitario. Todos los derechos reservados.
user:
  access: Entrar
  type:
    owner: Propietario
    renter: Inquilino
  form:
    owner: Datos del propietario
    role_title: Rol
    create_title: Crear cuenta
    update_title: Actualizar cuenta
    create: Crear
    update: Actualizar
    role_placeholder: Por favor seleccione el rol del usuario
    general_data: Datos generales
owner:
```

*Fuente: Elaboración propia*

En algunos casos, las traducciones deben tener una lógica más compleja, por ejemplo, para las tarjetas de cada vivienda, se usa una etiqueta con el tiempo transcurrido desde la creación. Sin embargo, las unidades deben ser dinámicas, es decir, si han transcurrido 3 días, no tiene sentido mostrar el resultado en segundos

*Figura 54. Diseñó de la tarjeta de la vivienda*


*Fuente: Elaboración propia*

Para implementar se usan las librerías de symfony para el soporte de ICU (International Components for Unicode) que es una biblioteca de software creada por Unicode Consortium que proporciona funcionalidades para manejar texto y datos en diferentes idiomas y escrituras. Este es el label para la traducción de texto en función de unidades y la cantidad.

*Figura 55. El fichero con traducciones ICU en castellano*

ations > messages+intl-icu.es.yaml

```
time_ago: >-
  {unit, select,
    second {{count, plural, offset:1
      =1    {hace un segundo.}
      other {hace # segundos.}
    }}
    minute {{count, plural, offset:1
      =1    {hace un minuto.}
      other {hace # minutos.}
    }}
    hour {{count, plural, offset:1
      =1    {hace una hora.}
      other {hace # horas.}
    }}
    day {{count, plural, offset:1
      =1    {hace un día.}
      other {hace # días.}
    }}
    month {{count, plural, offset:1
      =1    {hace un mes.}
      other {hace # meses.}
    }}
    year {{count, plural, offset:1
      =1    {hace un año.}
      other {hace # años.}
    }}
    other {{count, plural, offset:1
      other {No definido.}
    }}
  }
```

*Fuente: Elaboración propia*

De esta forma, podemos crear un nuevo filtro para twig que tenga una lógica muy simple

*Figura 56. El filtro avanzado para mostrar unidades en función de unidades y del tiempo transcurrido*

```php
public function timeAgo(\DateTimeInterface $dateTime)
{
    $now = new DateTime();
    $interval = $now->diff($dateTime);

    if ($interval->y > 0) {
        return $this->translator->trans(
            'time_ago',
            ['unit' => 'year', 'count' => $interval->y],
            'messages+intl-icu'
        );
    }
    if ($interval->m > 0) {
        return $this->translator->trans(
            'time_ago',
            ['unit' => 'month', 'count' => $interval->m],
            'messages+intl-icu'
        );
    }
    if ($interval->d > 0) {
        return $this->translator->trans(
            'time_ago',
            ['unit' => 'day', 'count' => $interval->d],
            'messages+intl-icu'
        );
    }
    if ($interval->h > 0) {
        return $this->translator->trans(
            'time_ago',
            ['unit' => 'hour', 'count' => $interval->h],
            'messages+intl-icu'
        );
    }
    if ($interval->i > 0) {
        return $this->translator->trans(
            'time_ago',
            ['unit' => 'minute', 'count' => $interval->i],
            'messages+intl-icu'
        );
    }

    return $this->translator->trans(
        'time_ago',
        ['unit' => 'second', 'count' => $interval->s],
        'messages+intl-icu'
    );
}
```

*Fuente: Elaboración propia*

De esta forma, en Twig simplemente tenemos que llamar a ese filtro

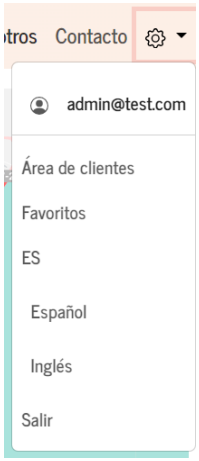
*Figura 57. Llamada al filtro custom desde Twig*

```
<div class="">
    <p class="card-text">
        <small class="text-body-secondary">{{ 'property.form.last_updated'|trans }}
            {{ property.updatedAt|time_ago() }}</small>
    </p>
</div>
```

*Fuente: Elaboración propia*

En cuanto al apartado visual, el usuario puede cambiar el idioma en cualquier momento desde el menú principal de la aplicación:

*Figura 58. Menu con la posibilidad de cambiar el idioma en tiempo real*




*Fuente: Elaboración propia*

# 6. Evaluación y pruebas

Después de crear la aplicación, es esencial asegurarse de que no haya errores en su diseño o visualización. Para lograr esto, la fase de evaluación se centra en probar cómo funciona nuestro sitio web en diferentes entornos y navegadores. El objetivo es garantizar que, una vez publicada la página, los usuarios no enfrenten problemas al utilizarla. Este proceso de evaluación implica realizar varias pruebas, las cuales se explicarán a continuación.

## 6.1 Pruebas de uso

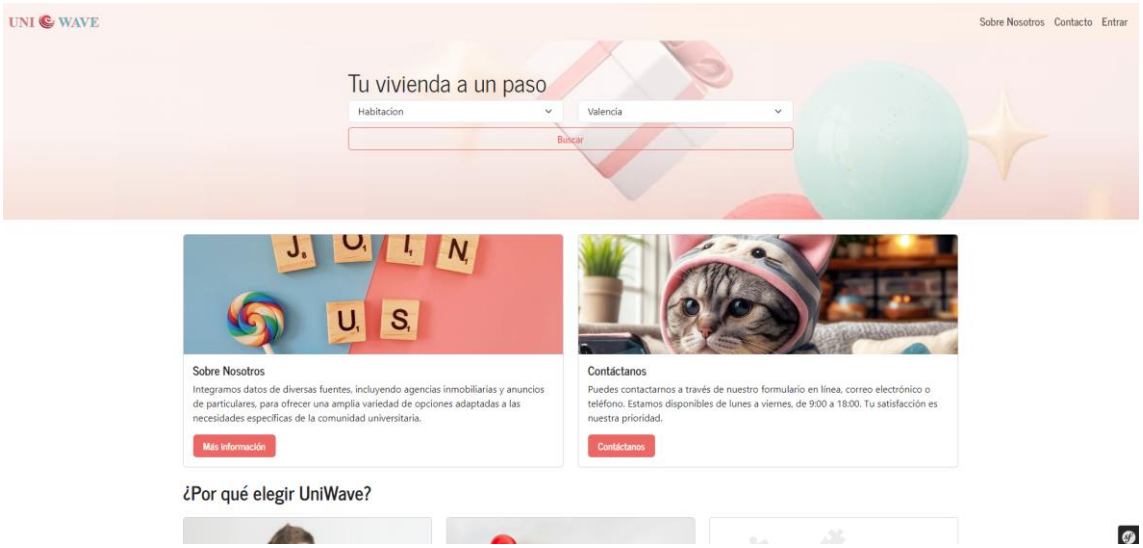
Las pruebas de uso, también conocidas como pruebas de usabilidad, son una técnica utilizada en el diseño de productos para evaluar qué tan fácil y eficiente es para los usuarios interactuar con un producto o sistema. En nuestro caso tenemos 2 tipos de usuarios:

- Usuarios sin cuenta:
    - Pueden ver las viviendas y detalles de las viviendas (no pueden ver datos de los propietarios en las viviendas)
- Estudiantes que buscan el alquiler
    - Todas las acciones anteriores
    - Pueden crear una cuenta
    - Pueden añadir viviendas a favoritos
    - Pueden solicitar la información del propietario
- Propietarios de las viviendas que quieren alquilarlas
    - Todas las acciones anteriores
    - Pueden publicar/editar/borrar sus viviendas

### 6.1.1 Usuario sin cuenta

Comprobamos que esos usuarios pueden entrar a la web y ver las viviendas

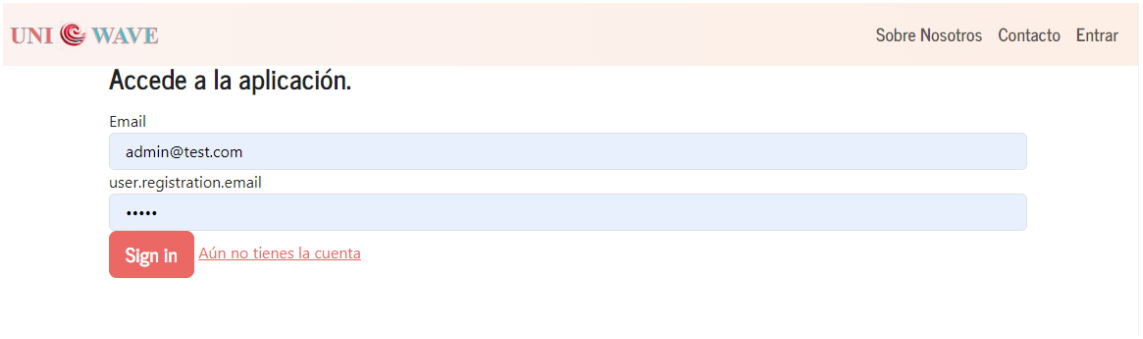
*Figura 59. Vista principal para usuarios sin cuenta*




*Fuente: Elaboración propia*

En la vista con viviendas, los usuarios no registrados pueden utilizar los filtros y ver el mapa interactivo, pero si hacen click en favoritos, entonces la aplicación redirecciona usuario al formulario para el login.

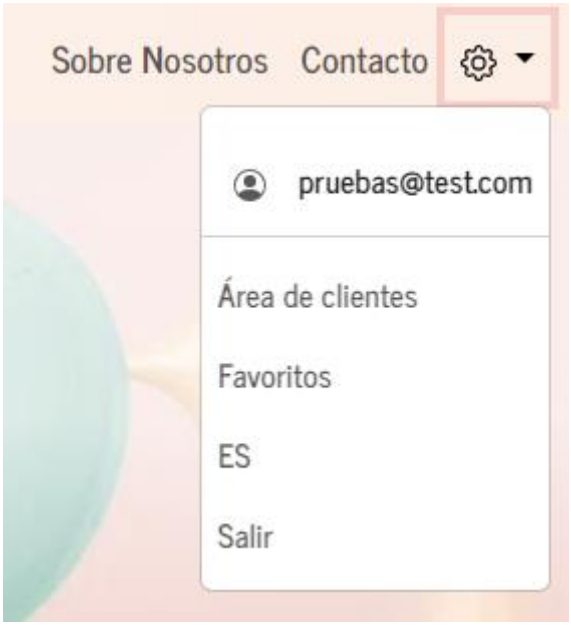
*Figura 60. Al hacer click sobre favoritos la app redirecciona al usuario no authentificado para que inicie la sesión*




*Fuente: Elaboración propia*

Comprobamos también, que el usuario authentificado tiene acceso al menú auxiliar

*Figura 61. Menú auxiliar de usuario autentificados*

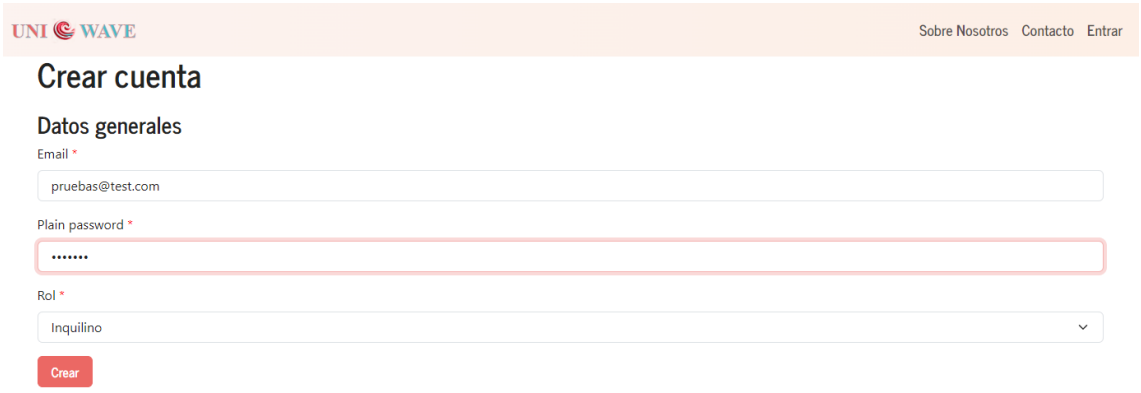



*Fuente: Elaboración propia*

### 6.1.2 Pruebas de uso para estudiantes.

En primer lugar, tenemos que comprobar que la creación de usuarios funciona correctamente, al crear la cuenta redirige al usuario a la pantalla principal

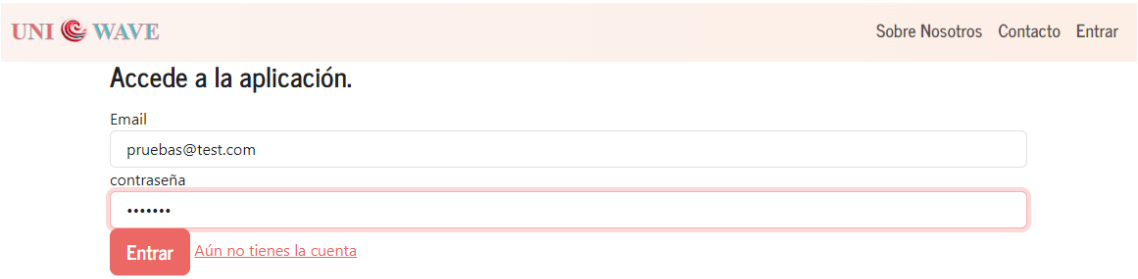
*Figura 62. Crear una nueva cuenta de tipo inquilino*




*Fuente: Elaboración propia*

Comprobamos que el usuario puede entrar en su perfil para editar datos (el correo electrónico)

*Figura 63. Editar datos del perfil del usuario*

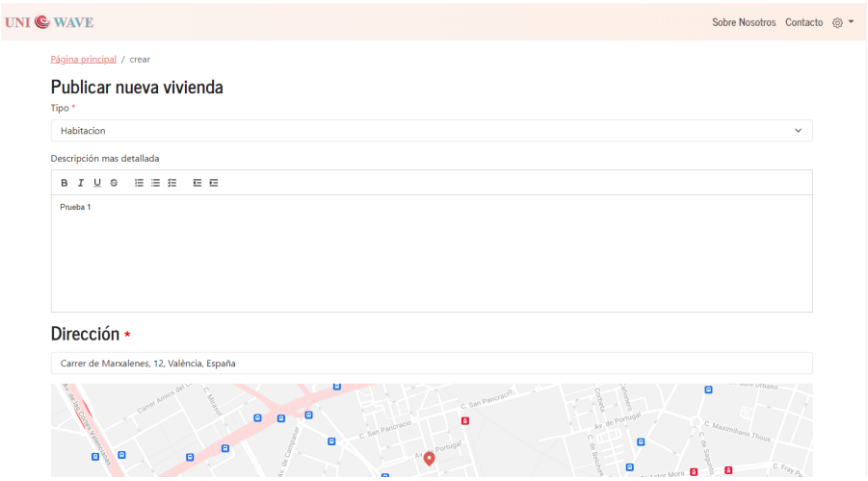



*Fuente: Elaboración propia*

### 6.1.3 Pruebas de uso para propietarios.

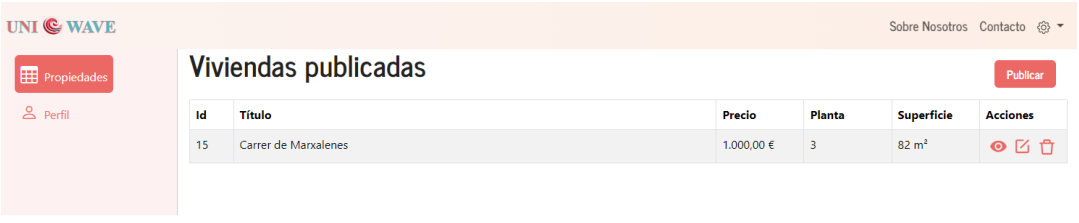
Los propietarios pueden hacer todas las acciones de estudiantes, más todas las acciones relacionadas con el CRUD de las viviendas.

*Figura 64. Creación de viviendas nuevas*



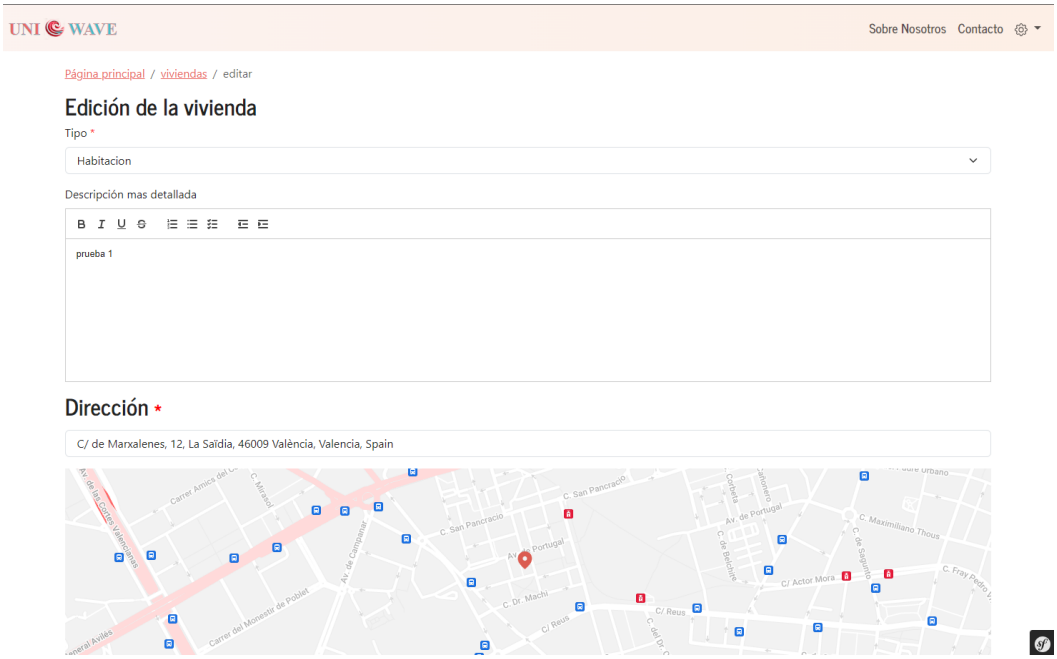

*Fuente: Elaboración propia*

*Figura 65. Área para gestión de viviendas*




*Fuente: Elaboración propia*

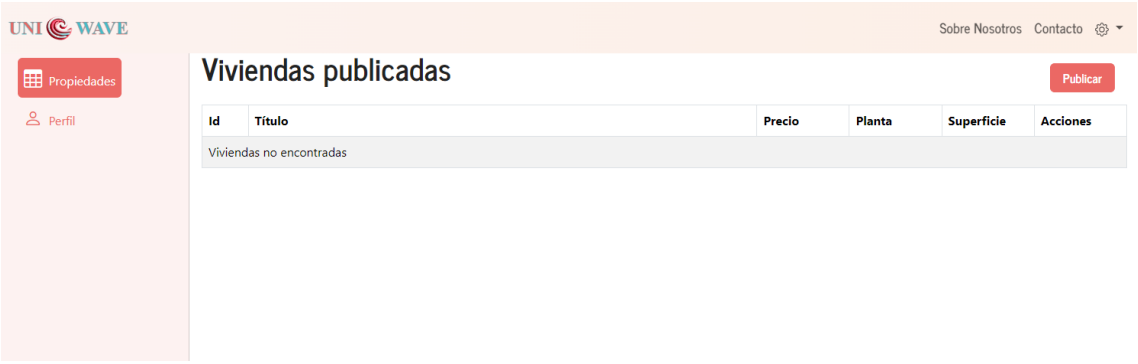
*Figura 66. Edición de las viviendas*




*Fuente: Elaboración propia*

También, es necesario probar que tras borrar la vivienda está se borra correctamente,

*Figura 67.El resultado después del borrado de la vivienda en Marxalenes*



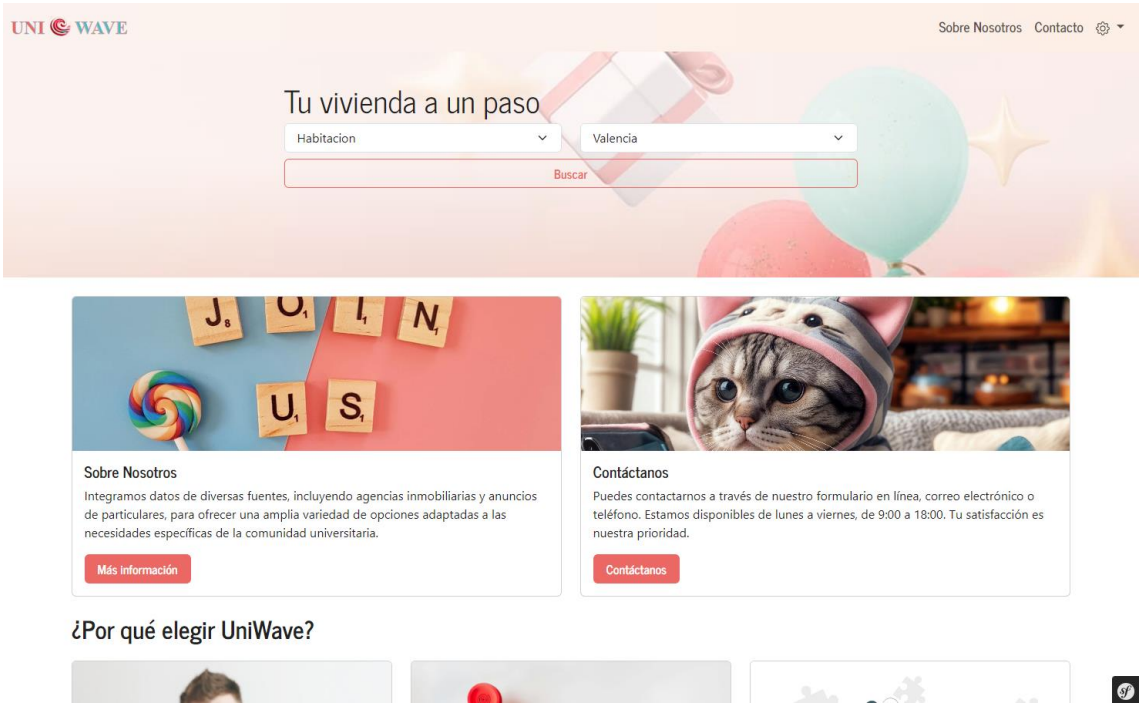

*Fuente: Elaboración propia*

## 6.2 Pruebas del diseño responsivo

En la actualidad, los usuarios acceden a plataformas web desde una amplia gama de dispositivos, como ordenadores, tabletas y teléfonos móviles. Para garantizar una experiencia de usuario óptima en el portal inmobiliario para estudiantes, se ha implementado un diseño responsivo que adapta el contenido a diferentes tamaños de pantalla y resoluciones.

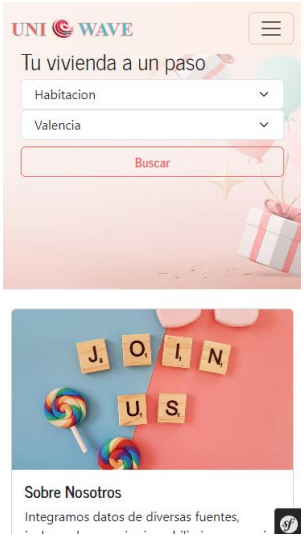
Este apartado presenta las pruebas realizadas para evaluar la efectividad del diseño responsivo del portal. El objetivo es asegurar una navegación intuitiva y accesible en cualquier dispositivo, proporcionando una experiencia de usuario coherente y satisfactoria.

*Figura 68. La página inicial de la aplicación en dispositivos grandes*



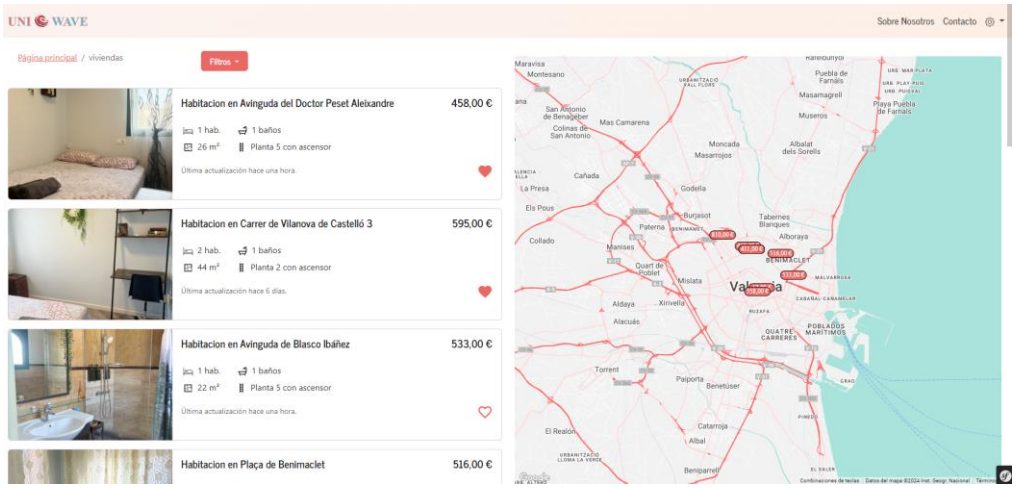

*Fuente: Elaboración propia*

*Figura 69. La página inicial de la aplicación en dispositivos móviles*




*Fuente: Elaboración propia*

*Figura 70. Página principal para visualizar viviendas en dispositivos grandes*

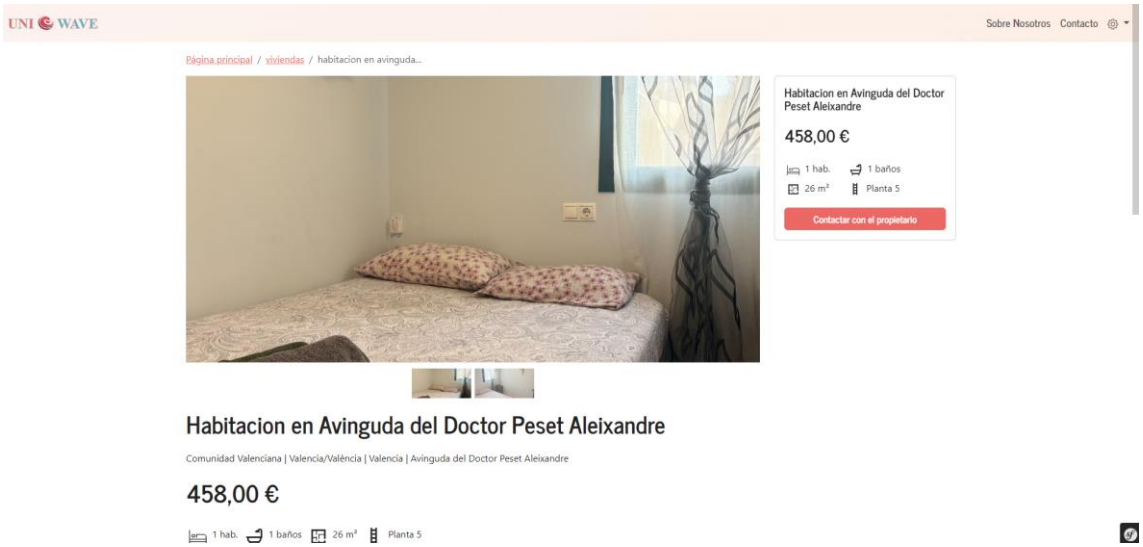



*Fuente: Elaboración propia*

*Figura 71. Página principal para visualizar viviendas en dispositivos dispositivos pequeños*


*Fuente: Elaboración propia*

*Figura 72. Detalles de la vivienda en dispositivos grandes.*




*Fuente: Elaboración propia*

*Figura 73. Detalles de la vivienda en dispositivos grandes.*


*Fuente: Elaboración propia*

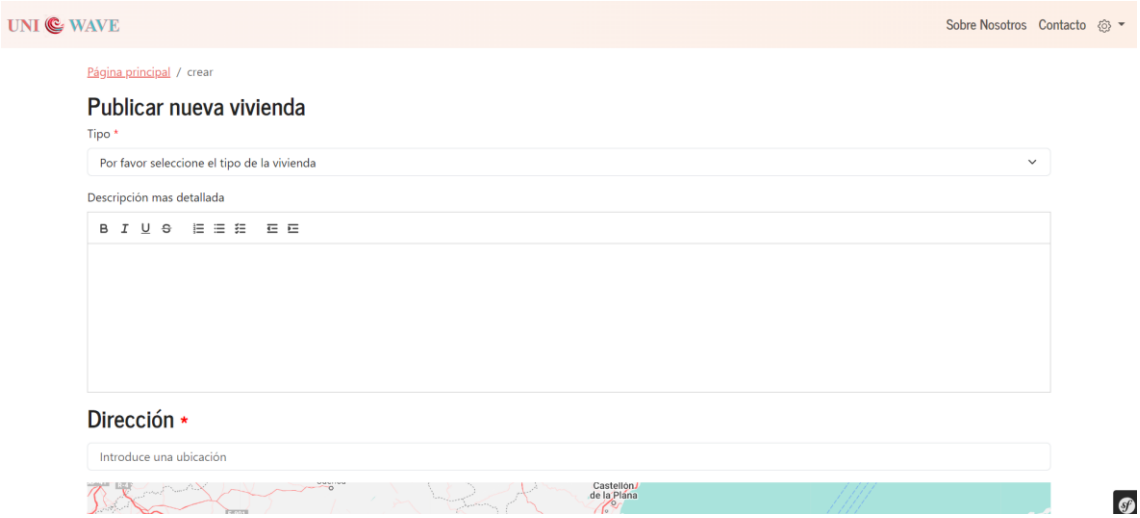
*Figura 74. Editar perfil en dispositivos grandes*


*Fuente: Elaboración propia*

*Figura 75. Editar perfil en dispositivos pequeños*

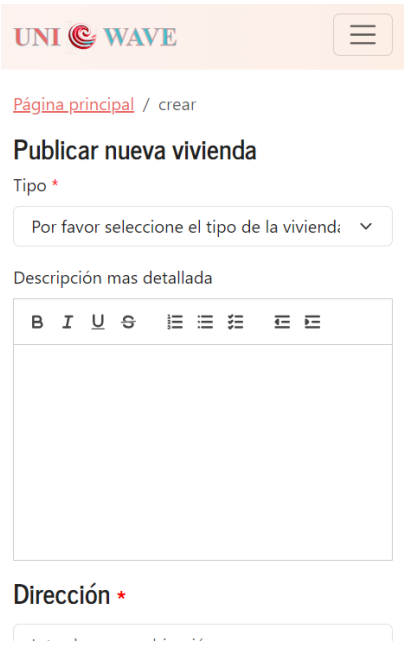

*Fuente: Elaboración propia*

*Figura 76. Publicar vivienda con dispositivo grande*




*Fuente: Elaboración propia*

*Figura 77. Publicar vivienda con dispositivos móviles*




*Fuente: Elaboración propia*

# 7. Conclusiones

En el desarrollo de este Trabajo Fin de Grado, se han alcanzado los objetivos planteados inicialmente, proporcionando una solución práctica para la búsqueda de vivienda dirigida a la comunidad universitaria. Se ha creado una plataforma web que permite crear las ofertas de alquiler específicamente dirigidas a estudiantes y personal universitario.

La plataforma ha sido diseñada con una interfaz de usuario intuitiva y amigable, adaptable a diferentes dispositivos, desde ordenadores de escritorio hasta teléfonos móviles. Este diseño responsivo asegura una experiencia de usuario coherente y satisfactoria, independientemente del dispositivo utilizado para acceder a la plataforma.

Además, se han implementado herramientas de búsqueda avanzada y filtros personalizados que permiten a los usuarios afinar sus búsquedas según múltiples criterios, como la ubicación, el precio, el tipo de propiedad y otros factores relevantes. Estas funcionalidades mejoran significativamente la eficiencia y efectividad del proceso de búsqueda. Además de las funcionalidades básicas de búsqueda se ha añadido soporte multilingüe.

Las pruebas de funcionalidad, rendimiento y usabilidad han demostrado que la plataforma cumple con las expectativas y necesidades de los usuarios. Los usuarios han podido navegar y utilizar la plataforma de manera efectiva, encontrando alojamientos adecuados y gestionando sus propiedades con facilidad.

En conclusión, este TFG ha logrado desarrollar una solución integral para la búsqueda de vivienda universitaria, abordando los desafíos iniciales y proporcionando un recurso valioso para estudiantes y personal universitario. El aprendizaje derivado de este proyecto abarca tanto aspectos técnicos como de gestión de proyectos, sentando una base sólida para futuros desarrollos y mejoras en el ámbito de plataformas inmobiliarias digitales.

# 8. Bibliografía

Bootstrap. (02 de 07 de 2024). *Bootstrap*. Obtenido de https://getbootstrap.com/docs/4.0/about/history/

Instituto Geográfico Nacional. (s.f.). *educativo.ign.es*. Recuperado el 28 de 07 de 2024, de https://educativo.ign.es/atlas-didactico/organizacion-territorial-bach/organizacin_territorial_espaola.html

MariaDB Foundation. (02 de 07 de 2024). *MariaDB en resumen*. Obtenido de https://mariadb.org/es/

PHP. (02 de 07 de 2024). *Historia de PHP*. Obtenido de https://www.php.net/manual/es/history.php.php

Stackoverflow. (2023). *Encuesta anual de Stackoverflow*. Obtenido de https://survey.stackoverflow.co/2023/#most-popular-technologies-language

*Symfony*. (02 de 07 de 2024). Obtenido de https://symfony.com/projects

# 9. Anexos

## 9.1 ODS (Objetivos de desarrollo sostenible)

*Tabla 1. Tabla 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 | |

**Fin de la pobreza:** los estudiantes que enfrentan dificultades financieras a menudo luchan para encontrar un alojamiento adecuado dentro de su presupuesto. Al facilitar la búsqueda de opciones de bajo costo, la app puede contribuir a reducir el estrés financiero y permitir que los estudiantes inviertan sus recursos en otras necesidades esenciales, como la alimentación y los materiales educativos.

**Salud y bienestar:** El acceso a una vivienda adecuada está estrechamente vinculado al bienestar físico y mental de los estudiantes. Una vivienda segura, limpia y bien ubicada puede reducir el estrés y la ansiedad, que son factores comunes en la vida universitaria. Además, al evitar condiciones de vida precarias la aplicación puede contribuir a mejorar la salud general de los

estudiantes, reduciendo riesgos como enfermedades o problemas de salud mental relacionados con el estrés por la vivienda.

**Educación de calidad:** la disponibilidad de un alojamiento adecuado y asequible es crucial para el éxito educativo de los estudiantes universitarios. La proximidad a la universidad, la tranquilidad del entorno y la estabilidad de la vivienda son factores que influyen directamente en el rendimiento académico. Al facilitar el acceso a viviendas que satisfagan estas necesidades, la aplicación apoya de manera significativa la educación de calidad, permitiendo que los estudiantes se concentren más en sus estudios y menos en las preocupaciones relacionadas con la vivienda.

**Trabajo decente y crecimiento económico:** una vivienda adecuada es un pilar fundamental para que los estudiantes completen su educación, lo que a su vez aumenta sus oportunidades de conseguir empleos decentes en el futuro. Además, al apoyar la movilidad estudiantil y la educación continua, la aplicación contribuye al crecimiento económico a largo plazo, ya que los estudiantes bien educados están mejor preparados para ingresar al mercado laboral y contribuir de manera significativa a la economía.

**Ciudades y comunidades sostenibles:** la aplicación apoya la creación de ciudades y comunidades más sostenibles al fomentar la ubicación de viviendas cerca de las universidades, reduciendo así la necesidad de desplazamientos largos y promoviendo el uso eficiente del espacio urbano. Además, al facilitar el acceso a viviendas bien ubicadas y asequibles, la app contribuye a la inclusión social y a la creación de comunidades estudiantiles más cohesionadas, lo que es esencial para el desarrollo de entornos urbanos más sostenibles y habitables.

**Alianzas para lograr los objetivos:** aunque la relación es menor en comparación con otros ODS, la aplicación tiene el potencial de fomentar alianzas estratégicas entre universidades, propietarios de viviendas y gobiernos locales. Estas colaboraciones pueden fortalecer las redes que apoyan a los estudiantes y promover la creación de políticas y programas que mejoren el acceso a la vivienda estudiantil. Al facilitar estas alianzas, la aplicación ayuda a

movilizar recursos y conocimientos para alcanzar otros objetivos de desarrollo sostenible.