ESCUELA DE INGENIERÍA DE FUENLABRADA

GRADO EN INGENIERÍA TELEMÁTICA

TRABAJO FIN DE GRADO

Versión Web Interactiva del Juego Social *Verdad o Reto*

Autor: Jose Luis Tamayo Díez

Tutor: Miguel Ángel Ortuño Pérez

Curso académico 2025/2026

Universidad Rey Juan Carlos

Resumen

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El objetivo de este capítulo es situar al lector en el contexto del proyecto, explicando el problema que da origen a la propuesta y las razones que motivan su desarrollo. Finalmente, se detalla el alcance del trabajo y la estructura general de la memoria.

1.1. Contexto y planteamiento del problema

En la actualidad, las aplicaciones web interactivas se han convertido en una herramienta habitual para el entretenimiento, la educación y la comunicación entre grupos de personas. Su facilidad de acceso desde cualquier dispositivo y la posibilidad de compartir experiencias en tiempo real las hace especialmente atractivas en entornos sociales y recreativos.

El clásico juego de *Verdad o Reto* es una de las dinámicas más populares en reuniones, fiestas o encuentros informales. Consiste en un juego en el que los participantes deben elegir entre responder sinceramente a una pregunta personal o realizar un desafío. Sin embargo, la mayoría de las versiones disponibles en línea presentan diversas limitaciones: no permiten personalizar las preguntas ni los retos, incluyen publicidad invasiva, requieren suscripción o no se adaptan correctamente a todos los tamaños de pantalla.

A partir de esta necesidad surge la idea de desarrollar una plataforma web sencilla, atractiva y completamente responsive que permita jugar a *Verdad o Reto* en cualquier dispositivo, sin necesidad de instalación, con la posibilidad de gestionar packs personalizados de preguntas y retos, y superando las limitaciones anteriormente mencionadas.

1.2. Motivación y objetivos del proyecto

La motivación principal para el desarrollo de este proyecto fue combinar el aprendizaje práctico de desarrollo web con el diseño de una aplicación interactiva completa, abarcando todas las fases del proyecto: desde su concepción inicial hasta su despliegue en producción.

El proyecto *Verdad o Reto* no solo busca ofrecer un producto funcional y entretenido,

sino también demostrar las competencias adquiridas en el Grado en Ingeniería, incluyendo programación web, diseño de interfaces, la gestión de bases de datos, la seguridad y el despliegue de aplicaciones en entornos reales.

A nivel personal, este trabajo representa una oportunidad para aplicar conocimientos de Python, Django, HTML, CSS y JavaScript en un entorno real, reforzando el enfoque *full stack* y fomentando el desarrollo de software orientado a la experiencia del usuario.

1.3. Alcance y público objetivo

La aplicación está diseñada para ser utilizada por grupos de amigos, estudiantes o familias que busquen una forma rápida y divertida de entretenerse desde un único dispositivo. Su interfaz es sencilla e intuitiva, y su diseño *responsive* permite que funcione correctamente en móviles, tablets y ordenadores.

El sistema también incluye un panel de administración desde el que los usuarios autenticados pueden crear y editar packs de preguntas y retos, gestionando su contenido de forma totalmente personalizada.

En cuanto al alcance técnico, el proyecto abarca el desarrollo del *frontend*, el *backend* y el despliegue en un servidor público, garantizando la correcta gestión de datos, la seguridad de acceso y una presentación visual coherente con la identidad del proyecto.

1.4. Estructura de la memoria

Esta memoria se estructura en varios capítulos que recogen todo el proceso de desarrollo del proyecto:

En este primer capítulo se introduce el contexto, explicando el alcance y los objetivos generales a lograr, así como la motivación que impulsó su desarrollo.

El segundo capítulo describe las tecnologías empleadas, tanto en el *backend* como en el *frontend*, incluyendo el entorno de desarrollo, los lenguajes utilizados y los mecanismos de seguridad y despliegue.

En el tercer capítulo se aborda el análisis y diseño del sistema, detallando los casos de uso, el modelo de datos y la estructura de la interfaz. Se muestra cómo hemos empleado estas tecnologías y herramientas utilizadas, justificando su elección.

El cuarto capítulo se centra en la implementación de las distintas funcionalidades.

El quinto describe el proceso de integración, configuración del entorno y despliegue en el servidor.

En el sexto capítulo se exponen las pruebas realizadas y el rendimiento de la aplicación.

El capítulo siete presenta las conclusiones, competencias alcanzadas y los objetivos cumplidos.

Para finalizar, el último capítulo recoge las conclusiones y plantea posibles líneas de mejora y trabajo futuro.

Tras los capítulos principales, tenemos una serie de anexos que contienen guías de instalación, manuales y documentación técnica adicional.

Capítulo 2

Estado del arte y tecnologías

2.1. Python y Django

2.1.1. *Python*

En la actualidad existen numerosos lenguajes de programación que permiten la posibilidad de desarrollar aplicaciones web y de escritorio. Entre ellos, *Python*[1] destaca por su sencillez, legibilidad y potencia, convirtiéndose en uno de los lenguajes más populares y versátiles en la industria del software.

Python es un lenguaje de programación de alto nivel, interpretado y de tipado dinámico. Su sintaxis clara y cercana al lenguaje natural facilita la comprensión del código, reduciendo la complejidad del desarrollo y el mantenimiento de proyectos. Gracias a estas características, se ha consolidado como una herramienta de referencia en ámbitos académicos, científicos y empresariales.

Entre las cualidades que hacen de *Python* una herramienta adecuada para este proyecto destacan su sintaxis clara y legible, que facilita el desarrollo y mantenimiento del código, y la gran cantidad de librerías disponibles para tareas como el desarrollo web, la gestión de datos o la automatización. Además, su compatibilidad con distintos sistemas operativos y el respaldo de una comunidad activa que mantiene el lenguaje actualizado y bien documentado lo convierten en una opción segura y eficiente para proyectos de cualquier tamaño.

En el contexto de este proyecto, *Python* se emplea principalmente como lenguaje base para el framework *Django*, encargado de gestionar la lógica del servidor, las peticiones HTTP, la seguridad y la conexión con la base de datos. Su elección se debe a su equilibrio entre simplicidad, rendimiento y robustez, además de su idoneidad para el desarrollo rápido de aplicaciones web.

[1] P. S. Foundation. «Python Official Documentation.» Consultado en octubre de 2025. dirección: https://www.python.org/.

2.1.2. Django

Sobre la base del lenguaje *Python*, este proyecto utiliza el framework *Django*[2], una herramienta de código abierto que facilita el desarrollo de aplicaciones web, diseñada para simplificar la creación de aplicaciones seguras, escalables y mantenibles. *Django* se fundamenta en el patrón de arquitectura *MVT (Model–View–Template)*, que separa claramente la lógica de negocio, la presentación y el acceso a datos, facilitando el mantenimiento y la evolución del sistema.

Asimismo, incorpora medidas de seguridad integradas frente a ataques comunes como *XSS*, *CSRF* o inyecciones SQL, y genera automáticamente un panel de administración que facilita la gestión de los modelos de datos. A ello se suma su sistema de plantillas, que permite separar el diseño del contenido dinámico, simplificando el mantenimiento de la interfaz y garantizando una experiencia de usuario coherente.

En el contexto del proyecto *Verdad o Reto*, *Django* actúa como el núcleo del sistema, gestionando las vistas, los modelos y la autenticación de usuarios, y organizando la comunicación entre las distintas capas del patrón MVT. Esta estructura garantiza una base sólida, modular y fácil de ampliar. Además, *Django* simplifica el despliegue en entornos productivos gracias a su compatibilidad con servidores como Gunicorn y su integración con bases de datos como SQLite o PostgreSQL, lo que permite adaptar la aplicación a distintos escenarios de uso.

2.2. Patrón MVT

El framework *Django* se fundamenta en el patrón de arquitectura *MVT (Model–View–Template)*, una adaptación del modelo clásico *MVC (Model–View–Controller)* orientada al desarrollo web. Este patrón separa de forma clara la lógica de negocio, la gestión de los datos y la presentación visual, lo que permite un desarrollo estructurado, mantenible y escalable.

El patrón MVT está compuesto por tres elementos principales que colaboran de manera coordinada:

El *Modelo (Model)* define la estructura de los datos y sus relaciones mediante clases de *Python*, que se traducen automáticamente en tablas dentro de la base de datos a través del sistema ORM (Object–Relational Mapper) de *Django*. Este enfoque permite manipular la información de forma sencilla sin necesidad de escribir consultas SQL, reduciendo errores y aumentando la productividad.

La *Vista (View)* actúa como intermediaria entre el modelo y la plantilla. Su función es procesar las solicitudes del usuario, aplicar la lógica de negocio necesaria, recuperar los datos requeridos y devolver una respuesta adecuada. De esta forma, las vistas controlan el flujo de información dentro de la aplicación y determinan qué contenido debe mostrarse en cada momento.

Por su parte, la *Plantilla (Template)* constituye la capa de presentación del sistema. Está

[2]D. S. Foundation. «Django Documentation.» Consultado en octubre de 2025. dirección: https://docs.djangoproject.com/en/stable/.

formada por archivos HTML que utilizan el lenguaje de plantillas de *Django*, el cual permite insertar variables y estructuras de control para generar contenido dinámico. Gracias a este mecanismo, el aspecto visual de la aplicación se mantiene separado de la lógica, facilitando la modificación del diseño sin afectar al funcionamiento interno.

El flujo de trabajo del patrón MVT sigue un proceso sencillo y bien definido: el usuario realiza una petición HTTP, *Django* identifica la vista correspondiente mediante el archivo `urls.py`, la vista obtiene y procesa los datos necesarios a través de los modelos, y finalmente se renderiza una plantilla que genera la respuesta HTML mostrada en el navegador.

En el proyecto *Verdad o Reto*, este patrón se aplica de forma completa. Los modelos representan las entidades principales del sistema (`Pack` y `Accion`), las vistas gestionan la lógica del juego y la interacción con el usuario, y las plantillas definen la interfaz visible desde el navegador. Gracias a esta estructura, el código de la aplicación se mantiene limpio, modular y fácilmente ampliable. Además, el uso del patrón MVT ha permitido desarrollar el proyecto de manera iterativa, incorporando mejoras progresivas —como el sistema de autenticación, la gestión de packs o la personalización visual— sin comprometer la estabilidad general de la aplicación.

2.3. Tecnologías web

El desarrollo del *frontend* de la aplicación *Verdad o Reto* ha sido desarrollado utilizando las tecnologías estándar de la web: *HTML*, *CSS* y *JavaScript*. Estas herramientas permiten definir la estructura, el estilo y la interactividad de la interfaz de usuario. El objetivo principal en el diseño del *frontend* ha sido crear una experiencia visual atractiva, intuitiva y coherente con el carácter lúdico del juego, garantizando al mismo tiempo una correcta visualización en todo tipo de dispositivos.

2.3.1. HTML

La estructura base de cada página se ha desarrollado con *HTML5*, siguiendo un enfoque semántico que mejora la accesibilidad y facilita la comprensión del código. Las plantillas de *Django*, ubicadas en la carpeta `templates/`, emplean etiquetas como {`% block content %`} y {`% endblock %`} para definir secciones dinámicas dentro de una plantilla principal, `base.html`, que actúa como marco común para el resto de las páginas.

Este enfoque modular permite mantener una estructura ordenada y coherente: los elementos compartidos (cabecera, pie de página, estilos globales o scripts) se definen una única vez en la plantilla base y se heredan en todas las demás páginas. De esta manera, se facilita el mantenimiento y la homogeneidad visual en toda la aplicación.

2.3.2. CSS

El lenguaje *CSS3* (*Cascading Style Sheets*) ha sido utilizado para definir la apariencia visual de la aplicación. *CSS* permite separar la presentación del contenido estructural, lo que favorece la reutilización del código y el mantenimiento a largo plazo. Además, proporciona herramientas avanzadas de diseño, como rejillas (*CSS Grid*), cajas flexibles (*Flexbox*) y variables personalizadas, que facilitan la creación de interfaces consistentes y adaptables.

El uso de *CSS* ha permitido definir un estilo visual coherente con la identidad del proyecto, aplicar animaciones ligeras y garantizar una presentación uniforme en distintos navegadores. La modularización de las hojas de estilo —organizadas por componentes y secciones— contribuye a la claridad del código y a su fácil actualización.

2.3.3. Diseño responsivo

El diseño *responsive* o adaptable constituye un aspecto esencial del desarrollo web actual. Permite que una misma interfaz se visualice correctamente en dispositivos con diferentes resoluciones y orientaciones, sin necesidad de mantener versiones separadas del sitio. Para lograr esta flexibilidad, se han utilizado unidades relativas (`vw`, `vh`, `em`) y consultas de medios (`@media`) que ajustan automáticamente las proporciones, márgenes y tipografías según el tamaño de la pantalla.

En el caso de *Verdad o Reto*, el diseño responsivo garantiza que la aplicación sea plenamente funcional tanto en dispositivos móviles como en ordenadores, manteniendo la legibilidad y la usabilidad del contenido. De este modo, se cumple uno de los objetivos fundamentales del proyecto: ofrecer una experiencia fluida e inclusiva en cualquier entorno.

2.3.4. Interactividad y usabilidad

El lenguaje *JavaScript* complementa las funcionalidades del *frontend*, permitiendo dotar de dinamismo e interactividad a la aplicación sin recargar el procesamiento del servidor. En este proyecto, se utiliza de manera controlada para mejorar la experiencia de usuario mediante pequeñas animaciones, validaciones y manipulación puntual del contenido del DOM (*Document Object Model*).

Asimismo, durante el desarrollo se han tenido en cuenta principios básicos de **usabilidad y accesibilidad**, orientados a garantizar una navegación intuitiva: contraste adecuado entre texto y fondo, elementos interactivos claramente identificables y una disposición visual que prioriza la legibilidad y la simplicidad. Estos aspectos contribuyen a una experiencia de usuario más agradable y accesible, en línea con los estándares actuales del diseño web.

2.4. Autenticación, CSRF y seguridad web

2.5. Persistencia de datos (SQLite/PostgreSQL)

2.6. Opciones de despliegue y comparación

Capítulo 3

Análisis y diseño

3.1. Casos de uso

3.2. Modelo de datos

3.3. Arquitectura del sistema

3.4. Diseño de la interfaz y navegación

3.5. Usabilidad y accesibilidad

Capítulo 4

Implementación

4.1. Estructura del proyecto Django

4.2. Gestión de packs, acciones (verdad/reto) y tokens

4.3. Vista pública: selección aleatoria y flujo de uso

4.4. Panel de gestión / dashboard

4.5. Autenticación y autorización

4.6. Gestión de errores 404/500 y logging

4.7. Internacionalización (si aplica)

Capítulo 5

Integración y despliegue

5.1. Entorno de desarrollo

5.2. Configuración de `settings` (`DEBUG`, `ALLOWED_HOSTS`, `STATIC_FILES`

5.3. Archivos estáticos y `collectstatic`

5.4. Dominio y DNS

5.5. HTTPS y certificados

5.6. Proceso de despliegue

5.7. Monitorización y copias de seguridad

Capítulo 6

Pruebas y validación

6.1. Plan de pruebas

6.2. Pruebas unitarias y de integración

6.3. Pruebas de usabilidad (móvil y escritorio)

6.4. Rendimiento

Capítulo 7

Resultados y discusión

7.1. Cumplimiento de objetivos

7.2. Limitaciones

7.3. Lecciones aprendidas

Capítulo 8

Conclusiones y trabajos futuros

8.1. Conclusiones

8.2. Trabajos futuros

Competencias

Bibliografía

[1] D. S. Foundation. «Django Documentation.» Consultado en octubre de 2025. dirección: https://docs.djangoproject.com/en/stable/.

[2] P. S. Foundation. «Python Official Documentation.» Consultado en octubre de 2025. dirección: https://www.python.org/.

Apéndice A

Guía de instalación y ejecución

Apéndice B

Manual de usuario

Apéndice C

Licencias y avisos legales