UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Extensión de un simulador de hogar digital para el etiquetado de datos de entrenamiento

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Puche Fernandez, Carlos Jesus

Tutor/a: Albert Albiol, Manuela

Cotutor/a externo: GIL PASCUAL, MIRIAM

CURSO ACADÉMICO: 2021/2022

Dedicatoria

Este trabajo lo dedico a mis padres, Antonio y Yolanda, que toda mi vida han sido apoyo, impulso y guía, me han hecho ser quien soy, y me han traído hasta aquí.

Se lo dedico también a Laura, la mejor compañera que puedo tener a mi lado, su inagotable alegría siempre me da el empuje en el momento adecuado.

Resum

En l'àmbit de la Inteligencia Ambiental es troben limitacions notables a l'hora de produir els data sets, el volums de dades que modelen les conductes de éssers humans, aquestes dades son arreplegades de manera automática per un sistema domòtic intel·ligent, però per a que aquest sistema puga arreplegar-les, un usuari ha de interactuar amb el sistema, i les seues accions han de ser registrades en manera que siguen mesurables i etiquetables. I en moltes ocasions quan es condueixen aquest tipus d'experiments, no está adequadament documentada la manera en què s'han obtingut les dades, ni la metodología empleada per mesurar, ni les característiques del sensors empleats, etc. En eixe context es troba la necessitat de conseguir data sets amb l'etiqueta del nivell d'obtrusivitat desitjat per l'usuari en un moment determinat, al voltant d'una activitat determinada. Per donar solució a aquest problema, el present document proposa una plataforma web, al voltant del simulador Open Smart Home Simulator, per tal de facilitar-ne l'ús del OpenSHS, i així mateix permetre un potencial ús multitudinari de la ferramenta que ha de servir per posar a disposició pública més data sets que modelen la interacción entre l'usuari i el sistema de un hogar intel·ligent. Els pròxims capítols detallen les qüestions principals de análisis del problema, de disseny, desenvolupament, i implementació i probes.

Paraules clau: reconeixement automàtic de conducta humana, data set, obtrusividad

Resumen

En el ámbito de la Inteligencia Ambiental se encuentran limitaciones notables a la hora de producir los data sets, los volúmenes de datos que modelan las conductas de seres humanos, éstas son recogidas de manera automática por un sistetma domótico inteligente, pero para que este sistema pueda recogerlas, un usuario tiene que interactuar con el sistema, y sus acciones tienen que ser registradas de manera que sean medibles y etiquetables. Y en muchas ocasiones cuando se conducen este tipo de experimentos, no está adecuadamente documentada la manera en la que se han obetnido los datos, ni la metodología empleada para medir, ni las características de los sensores empleados, etc. En ese contexto se encuentra la necesidad de conseguir data sets con la etiqueta del nivel de obtrusividad deseado por el usuario en un momento determinado, al rededor de una actividad determinada. Para dar solución a este problema, el presente documento propone una plataforma web en torno al simulador Open Smart Home Simulator, con tal de facilitar el uso del OpenSHS, y así mismo permitir un potencial uso multitudinario de la herramienta que tiene que servir para poner a disposición pública más data sets que modelen la interacción entre el usuario y el sistema de un hogar inteligente. Los próximos capítulos detallan las cuestiones principales del análisis del problema, de diseño, desarrollo, e implementación y pruebas.

Palabras clave: reconocimiento atuomático de conducta humana, data set, obtrusividad

Abstract

In the field of Ambient Intelligence there are notable limitations when it comes to generate data sets, the data used to model the behaviour of humans, this are automatically registered by an smart domothic system, but to be able to gather this data, the system needs a user that interacts with it, and user's actions must be registered in a way that can be seized and labelled. And in many times when an experiment like this is realised, the way the data was obtained is not properly informed, or the methodology for seizing, or the properties of the sensors used, ... In this context is framed the need of generating data

sets labelled with the desired obtrusivity level, in an specific moment, around an specific activity. To give a solution to this problem, this document proposes a web platform around the Open Smart Home Simulator, so to make OpenSHS' use easier, and also get a potentially extended use of this tool, that must allow to have at publical disposal, more data sets modelling the interaction between user and system in an smart home. The next chapters describe the main issues about problem analysis, design, development, implementation and tests.

Key words: human activity recognition, data set, obtrusivity

Índice general

Índice general v
Índice de figuras vii
Índice de tablas vii

1 Introducción 1
1.1 Motivación . . . 3
1.2 Objetivos . . . 3
1.3 Estructura de la memoria . . . 4
2 Estado del arte 7
2.1 Propuesta . . . 8
3 Análisis del problema 9
3.1 Análisis de una solución plausible. . . . 10
3.2 Solución propuesta . . . 12
3.3 Plan de trabajo . . . 12
4 Diseño de la solución 13
4.1 Arquitectura del sistema . . . 13
4.2 Diseño detallado . . . 15
4.3 Tecnología utilizada . . . 18
5 Desarrollo de la solución 21
6 Implantación y pruebas 25
7 Conclusiones 27
7.1 Relación del trabajo con los estudios cursados . . . 28
8 Trabajos futuros 31
Bibliografía 33

Apéndice
A Objetivos de Desarrollo Sostenible 35

Índice de figuras

4.1 Arquitectura del sistema 14
4.2 Patrón Contenedor/Presentador 16
4.3 Diseño de la base de datos 18

5.1 Captura de la ejecución del simulador 22

Índice de tablas

6.1 Tabla de pruebas 25

CAPÍTULO 1
Introducción

El desarrollo de las técnicas de aprendizaje automático y los sistemas inteligentes han permitido desarrollar los sistemas domóticos enormemente, impulsando líneas de investigación como la inteligencia ambiental, que en los últimos años están adquiriendo un fuerte protagonismo, a medida que la población dependiente de asistencia en las actividades cotidianas aumenta debido al envejecimiento poblacional. Además, el uso de éstas técnicas en espacios tanto públicos como domésticos es muy interesante para alcanzar nuevas cotas de eficiencia energética y optimización de recursos, cosa que se entronca con los objetivos de desarrollo sostenible marcados por la ONU.

Una problemática común cuando se trata de proyectos que emplean aprendizaje automático en domótica es la obtención de datos de entrenamiento y test; los corpus de datos, o data sets, que se necesitan para que el sistema aprenda lo que se requiere de él. Para obtener estos data sets en proyectos relacionados con el reconocimiento automático de la actividad humana, varios usuarios realizan actividades que son recogidas mediante sensores para modelar las conductas que el sistema debe aprender.

Hay diversas formas de registrar las actividades a reconocer por el sistema. Se pueden registrar mientras se realizan de foma natural, registrando las acciones de usuarios que se comportan sin condicionar sus conductas [1], de manera parcialmente condicionada, con un usuario que en determinados momentos realiza una o varias acciones que deben ser reconocidas por el sistema [2], o de manera totalmente estructurada por un guión con el fin de que los sensores de un laboratorio capten determinados movimientos o acciones [3]. Por otra parte, las metodologías empleadas en la generación de estos datos, deben estar claramente especificadas y seguir algún estándar para poner en contexto las conclusiones extraídas, y que éstas tengan validez formal; éste último punto dificulta hacer uso de data sets públicos ya que en muchos casos están pobremente documentadas. Otra aproximación a este problema, es usar la simulación para generar data sets. Simular situaciones, además de eliminar la necesidad de actores, permite experimentar usando periodos de tiempo amplios de manera ágil y poco costosa, y probar situaciones límite o de riesgo, o más intrusivas en la intimidad individual, sin los problemas de seguridad y la encrucijada ética que ello conllevaría [4].

El presente documento expone un proyecto para solucionar estos problemas, centrándose en un generador de data sets como es el simulador Open Smart Home Simulator[1] (OpenSHS). El OpenSHS constituye un entorno de experimentación centrado en un hogar digital que cuenta con distintas estancias, en las cuáles hay distribuidos varios sensores en puertas, electrodomésticos, mobiliario e interruptores, y que genera datos para distintos contextos y rangos horarios. El proyecto se centra en facilitar la documentación y etiquetado de data sets orientados a mejorar la interacción usuario-sistema, es decir,

[1] Se puede consultar la web del proyecto en https://openshs.github.io/openshs/

que un hogar inteligente, que use tecnologías de aprendizaje automático, aprenda el nivel de obtrusividad que debe emplear en la comunicación con el usuario. En este sentido, es necesario que el simulador posibilite al experimentador registrar el nivel de obtrusividad en diversos contextos y situaciones, y esto sea plasmado en el data set resultante.

Como ya se ha comentado, un factor importante del aprendizaje de estos sistemas es la variedad, pero siguiendo unos estándares, de los data sets empleados. Por eso la propuesta del proyecto incluye el desarrollo de una plataforma en línea que simplifica el uso del simulador, con un servidor propio para simular diversos contextos, y obtener los data sets resultantes ya etiquetados. La plataforma consta de una página web desarrollada en el framework de javascript *React*, un servidor Node para hacer posible la ejecución del simulador, y una base de datos de *Firebase*.

Introducción al Open Smart Home Simulator. Como se ha comentado arriba, el OpenSHS simula un hogar digital con varios sensores que recogen ciertas acciones de usuario. El escenario es una casa con un dormitorio, un despacho, cocina, baño y salón. Hay un sensor en la puerta de cada una de las estancias. Además recoge acciones como sentarse o levantarse de la silla, acostarse o levantarse de la cama, sentarse o levantarse del sofá, y apagar o encender electrodomésticos y luces en todas las estancias.

Las actividades que realiza el usuario se recogen en intervalos de tiempo, se pueden iniciar interactuando con ciertos elementos (p.e. si hago click en la mesa del despacho dará por iniciada la actividad "trabajar"), o desde la interfaz de usuario que tiene internamente el simulador, desde ésta se puede hacer que pase un cierto periodo de tiempo una vez iniciada la actividad, así se pueden simular periodos de horas, y concretar ciertos estados del simulador en el momento en concreto del día en que el usuario realiza una determinada acción. El estado de los sensores y las acciones realizadas, se representan por las columnas del data set, de modo que una fila del data set recoge el estado de los sensores de la casa y la acción realizada por el usuario en un momento en concreto, y la actividad en que estaba ocupado el usuario en el momento se etiqueta en la última columna de la fila. El proyecto pretende añadir otra etiqueta, justo antes de la etiqueta de actividad, que señale qué nivel de obtrusividad se desearía en esa situación, si el sistema tuviera que comunicar algo al usuario.

Para generar un data set se deben simular el contexto de la mañana y el contexto de la tarde, lanzando cada uno de ellos para un día entre semana y un día de fin de semana. Cuando están los cuatro data sets 'temporales', se pueden agregar los datos en un data set final cuya información se muestra en función a dos parámetros: (i) El número de días que generamos, que define cuántas iteraciones hará el bucle generador de muestras, a partir de la información de los data sets temporales. Como el simulador calcula si una fecha corresponde a un día entre semana o fin de semana, generará muestras para cada uno de los n días que le hemos introducido, desde la fecha indicada cuando hemos lanzado la simulación, en función de si es un día entre semana o fin de semanana. Además podemos decirle en qué rangos de tiempo queremos agrupar las filas, por tanto si cada fila debe representar un rango de 5, 10, 20 min, o incluso más, entonces acotar el número de filas en que se representan las actividades.

Todo el simulador está implementado en Python, utilizando la Blender Python API para integrar los procesos de ejecución de blender con scripts de Python. Para lanzar una simulación hay que ejecutar por consola un comando que ejecuta el script "openshs.py" que lanza el contexto, mañana o tarde, que se le pasa como argumento; a continuación una línea de consola solicita al usuario una fecha y hora de inicio de ese contexto y arranca el simulador, se realizan las actividades, se simulan distintas situaciones y cuando se da por bueno se cierra el simulador, esto hace que se cree un archivo temporal para ese contexto y día. Los contextos son dos modulos de Blender separados, y por tanto

ejecuciones separadas, como se verá más adelante ésta forma de ejecutarse marcará las dificultades para integrarlo con la plataforma que se propone.

1.1 Motivación

Hay dos fenómenos de importancia mayor que si se combinan en los próximos años pueden ser la base de un gran salto en el campo de la inteligencia ambiental: (i) La baja tasa de natalidad en Europa, EE.UU y también algunos países asiáticos como Japón, ha propiciado un envejecimiento de la población y un aumento proporcional del número de personas que requieren cuidados asistenciales, por tanto las tecnologías domóticas y de supervisión y cuidados telemáticos tendrán presumiblemente un rol mayor en la vida cotidiana. (ii) Siguiendo los Objetivos de Desarrollo Sostenible de la ONU, gobiernos y tecnológicas de todo el planeta han puesto el foco en el coche eléctrico para reducir las emisiones de carbono, conforme se prepara esa revolución del sector del automóvil, se ha extendido el interés no sólo en el tipo del motor del vehículo, sino en la forma en que la energía es usada en todos sus aspectos, y particularmente en las ciudades, que concentran grandes masas de población, las *smart cities* han pasado por la primera plana de numerosos periódicos, y la inteligencia ambiental se plantea como una herramienta útil para lograr una mayor optimización energética en espacios domésticos y de uso público.

Por ello es un buen momento para proporcionar soluciones a los científicos y técnicos del ámbito, introducirse en la literatura relacionada, entender cuáles son los problemas comunes, las necesidades de la tecnología y de los usuarios que hacen uso de ella en el momento. En el apartado más técnico era realmente interesante la óptica de implementar los cambios en un software como el OpenSHS, entender el trabajo de otro ingeniero, su código, el funcionamiento de principio a fin y clase por clase, para asegurar que los cambios realizados no alterasen su correcto funcionamiento. Por otra parte, la propuesta de la plataforma y por tanto preparar el propio simulador para la integración con ella, implicaba producir una solución *end-to-end* cuya arquitectura, diseño y desarrollo recaían totalmente en decisiones propias. Un proyecto así es la oportunidad perfecta para probar los conocimientos adquiridos durante el grado, y tantear habilidades como la resolución de problemas, buscar soluciones alternativas, recabar información sobre tópicos que son poco familiares y asimilarlos, y otras competencias transversales.

1.2 Objetivos

El proyecto pretende desarrollar una tecnología que permita obtener de manera ágil data sets con la información sobre la conducta de un usuario, a partir de los sensores de un hogar digital simulado. Éstos data set deben estar etiquetados con el nivel de obtrusividad que éste desea cuando el sistema debe notificarle algo. Además es importante que esta tecnología sea ampliamente accesible y usada fácilmente por multitud de usuarios para lograr una mayor variedad de data sets, por tanto resulta interesante disponer de una interfaz sencilla, desde la que lanzar el simulador, y a ser posible se pueda acceder a ella en línea para ser usada por multitud de usuarios y experimentadores. Los objetivos que se proponen para satisfacer estas necesidades son:

- Poder seleccionar durante la simulación un nivel de obtrusividad y ejecutar OpenSHS en un servidor.
    - Implementar la funcionalidad para registrar tres niveles distintos de obtrusividad.

    - Reducir la interacción por consola en el arranque del simulador, y si es posible, eliminar el uso de consola.

- Poner en marcha una plataforma que sirva para lanzar ejecuciones parametrizadas del simulador, así como subir y ver data sets de otros usuarios.

    - Preparar una página web que sea la recepción del usuario y la interfaz de arranque del simulador.
    - Desarrollar un servidor que recoja las peticiones de la web, ejecute el simulador de acuerdo a los parámetros dados, y genere el data set.
    - Preparar una base de datos, accesible desde la web, que permita a los usuarios registrarse para subir y consultar data sets.

1.3 Estructura de la memoria

En el capítulo 2 el lector puede encontrar la descripción del estado del arte. Es un espacio de lectura destinado a repasar el contexto del uso de simuladores para trabajos de investigación y otros proyectos de la inteligencia ambiental, también se detalla el espacio que el proyecto busca llenar, incluyendo una descripción de la propuesta de proyecto.

En el capítulo 3 de la memoria se analiza en concreto el problema de la elaboración de data sets que modelen la interacción de un usuario con un hogar inteligente, conretamente en la comunicación entre los dos. También en ese capítulo se proyecta un plan de trabajo para cumplir los objetivos descritos en el capítulo anterior al actual; y cierra el capítulo una estimación del coste en horas y recursos económicos necesarios para hacer frente al proyecto.

El cuarto capítulo aborda el diseño de la solución. Se detallan y justifican las elecciones de diseño realizadas; también se detalla la arquitectura de todo el sistema que compone la solución, y la interacción entre las distintas partes que conforman el proyecto: la web, incluida la base de datos, el servidor y el simulador, y por último se enumeran las tecnologías empleadas para el desarrollo de las distintas partes.

En el captíulo 5 se abordan las varias fases del desarrollo; componente a componente del sistema, se describen los pasos necesarios para su desarrollo, también se comentan los problemas que han aparecido durante los desarrollos, y como las decisiones que se han tomado en orden al cumplimiento de los objetivos, o aproximarse a ellos todo lo posible, han concretizado el proyecto en una solución real.

Habiendo expuesto el desarrollo, el lector llega al capítulo 6 donde se presentan la implantación y las pruebas hechas para poner en marcha la solución, se abordan las necesidades para el servidor y las pruebas de ejecución del simulador, para ver que todo el funcionamiento del mismo está recogido en el producto final.

El capítulo 7 recoge las conclusiones que expongo como desarrollador del proyecto, es mi valoración como ingeniero de la consecución de los objetivos planteados al inicio de la memoria, además de los problemas y retos que he encontrado durante el desarrollo, también sopeso qué he aprendido, y qué conocimientos, fruto del grado, he aplicado para hacer frente a las distintas etapas del desarrollo. Finalmente, sigue a las conclusiones un breve capítulo que enumera algunos aspectos del desarrollo que no se han podido profundizar, pero que serían convenientes para dejar una solución perfectamente cualificada para estar en producción y usada a gran escala, así como aspectos del desarrollo que considero importante evitar en ampliaciones y mejoras. Marcan el cierre de la exposición las referencias bibliográficas empleadas.

Al final del documento el lector podrá el anexo que describe cómo el proyecto puede servir a los Objetivos de Desarrollo Sostenible.

CAPÍTULO 2
Estado del arte

Los avances en domótica, en sistemas multi-agente e inteligencia artificial han hecho posible interconectar un inmenso abanico de dispositivos, ofreciendo un salto en servicios y prestaciones a los usuarios de éstos. Esas nuevas prestaciones que han aparecido tienen su base en muchas ocasiones en las técnicas de aprendizaje automático, que a partir de los modelos de reconocimiento de la conducta humana, son capaces de entender los patrones en la actividad del usuario para asistirle en diversas tareas.

El punto crítico de estos sistemas de aprendizaje automático es la necesidad de los corpus de datos que modelan las conductas con las que entrenar su algoritmo y poder ser de ayuda, aquí es donde el campo de la Inteligencia Ambiental encuentra una problemática endémica como recogen en su investigación sobre los límites del reconocimiento de la actividad humana, Jeffrey W. Lockhart y Gary M. Weiss [1], pues existe un problema a la hora de disponer de data sets públicos por la falta de rigor en la metodología empleada para generarlos, la escasa documentación del contexto de la experimentación, o de los dispositivos hardware empleados, además de una escasez, en general, de data sets accesibles públicamente, debido en parte a lo costoso que es preparar estos laboratorios y entornos de experimentación, y disponer de usuarios suficientes para registrar cierta variedad en los datos.

Ante las dificultades de conseguir estos datos de manera natural, hace años que en multitud de líneas de investigación entorno a la conducta humana se usan simuladores para facilitar la recogida y el modelado de los datos. Los intereses de modelar el comportamiento humano en ámbitos muy dispares entre ellos ha ido aumentando conforme se desarrollaba la tríada de tecnologías enunciada en el primer párrafo, abarcando bajo la visión global de la inteligencia ambiental, proyectos que en lo concreto tienen fines muy diferentes, pero que todos tienen en el centro la conducta humana y el contexto en que ésta se desarrolla, y además un sistema informático que juega un rol más o menos activo, pero de una forma u otra está presente.

Un ámbito en que no sorprende mucho ver el uso de simuladores, es en el diseño de sistemas de seguridad automovilísticos, en ellos hace décadas que se usan simuladores para probar cómo se comporta el coche y los distintos elementos del tráfico en caso de accidentes, pero desde hace unos años se viene utilizando también la simulación para modelar la conducta humana y la interacción del conductor con los elementos del tráfico, y su peso de causalidad en los factores que conducen al accidente [6], y más desde que el coche autónomo se ha convertido en una realidad que va camino de extenderse y hacer más frecuente.

[1] Para más detalle leer la investigación presentada por los autores en la *International Joint Conference on Pervasisve and Ubiquitous Computing* [5]

Otra línea donde se ha probado útil usar la simulación, es en el modelado del comportamiento humano ante situaciones de emergencia. Ya se ha comentado que modelar a partir de experimentos la conducta humana y medirla es complejo y bastante costoso en tiempo y recursos, un experimento que quiera modelar la conducta humana bajo situaciones de emergencia o crisis es esencialmente imposible, la respuesta humana ante una crisis está marcada por los impulsos de supervivencia, el estrés y la ansiedad, ningún actor podría representar la conducta real de una situación similar, sin arriesgar realmente su vida [7].

Además de esos ámbitos el uso de simuladores en el diseño de edificios se ha convertido en una parte fundamental para recoger no sólo necesidades arquitectónicas, sino también mejorar la seguridad antes emergencias, los rangos de eficiencia energética y consumo de recursos. Entender cómo usan los usuarios el edificio, qué zonas ocupan por periodos más largos de tiempo, qué zonas tienen más afluencia de gente, es útil para diseñar planes de evacuación ante incendios, unido a la dimensión del modelado psicológico que se comentaba en el apartado anterior [8], y por otra parte, y atendiendo al consumo de energía y el uso de recursos, para ir en línea con los Objetivos de Desarrollo Sostenible, la inteligencia ambiental puede tener un papel importante a la hora de economizar, y no sólo el gasto doméstico, si no la propia construcción, Seung Wan Hong (et al., 2020) plantea un concepto de sostenibilidad social tomando como premisa que la relación de las personas con el edificio que habitan o usan frecuentemente, es mejor cuando éste satisface adecuadamente sus necesidades, pues los usuarios estarán contentos y esto se traduciría, según el autor, en una mejor conservación del edificio y el entorno. [9]

2.1 Propuesta

La propuesta de solución consiste en desarrollar una herramienta para elaborar data sets etiquetados por los usuarios con la obtrusividad deseada cuando el sistema deba notificarles algo, en un momento dado, para los varios contextos posibles, y así evitar el etiquedado manual por parte del usuario o el experimentador. Se propone también, desarrollar una aplicación o plataforma que sirva de lanzadera para el simulador, para que un mayor número de usuarios puedan producir data sets de manera fácil, así como subir a la plataforma los suyos y poder descargar los de otros, con el fin de ampliar el número de data sets a disposición pública, para investigaciones y modelos que quieran modelar la comunicación usuario-sistema.

CAPÍTULO 3
Análisis del problema

El proyecto hace a la vez frente a una necesidad concreta y una problemática endémica del ámbito de la inteligencia ambiental. Por un lado, se requiere de un generador de data sets que, en el marco de un hogar digital, permita al experimentador o al usuario etiquetar para diversos contextos o situaciones, y mientras se realizan varias actividades, el nivel de obtrusividad deseado para las notificaciones del sistema hacia el usuario. Como se ha adelantado en la introducción, el OpenSHS tiene un problema con su forma de ejecutarse, que tiene implicaciones a nivel de diseño de la solución. Éste está constituido por dos bloques ejecutables para cada franja del día, uno para el contexto de la mañana y otro para el de la tarde, y ya que no es posible constituir un único archivo ejecutable, la carpeta entera del simulador debe ser accesible para ejecutarlo, ésta circunstancia, afecta a la manera en que se integrará el simulador con cualquier plataforma o aplicación con el fin de agilizar su ejecución.

La necesidad expuesta arriba, se entronca con el problema que se ha comentado en el estado del arte, cuando se trata de disponer de data sets variados, estandarizados y bien documentados en su metodología de elaboración; por tanto la herramienta también tiene que facilitar el uso del simulador y poder ser utilizada por gran número de usuarios, y un perfil variado de los mismos, para poner a disposición pública una mayor cantidad de data sets construídos bajo el mismo método. A continuación, se enumeran los requisitos que deben cumplir el simulador y la plataforma, empleando un estándar para definir criterios de aceptación de ingeniería del software:

Requisitos del simulador.

- Como usuario del OpenSHS, quiero poder etiquetar con el nivel de obtrusividad deseado distintas situaciones, durante la simulación de los varios contextos posibles
    - Ejecuto el OpenSHS
    - Accedo a la interfaz de selección de actividades y tiempo
    - Verifico que aparecen 3 botones más, con los respectivos niveles de obtrusividad definidos
    - Selecciono una actividad
    - En un momento dado, selecciono un nivel de obtrusividad
    - Sigo realizando actividades, para todos los contextos
    - Agrego los data sets
    - Verifico que en el data set resultante cada actividad lleva asociada una etiqueta con el nivel de obtrusivad para ese momento

Requisitos de la plataforma.

- Como usuario del OpenSHS, quiero disponer de una plataforma con una interfaz para ejecutar el simulador, y subir y consultar data sets
  - Accedo a la plataforma
  - Verifico que hay una sección para la ejecución del simulador
  - Accedo a la sección
  - Verifico que aparece un campo para definir cuantos días quiero abarcar en la agregación de datos
  - Introduzco un número de días
  - Verifico que aparece un campo para definir el espacio de tiempo en que quiero fraccionar las actividades
  - Introduzco el tiempo en que quiero que se condensen las actividades
  - Verifico que hay un campo para introducir la fecha de inicio y la hora
  - Introduzco una fecha y una hora
  - Verifico que hay un botón "PLAY"para arrancar la simulación
  - Hago click
  - Verifico que arranca el simulador con la hora y la fecha introducidas
  - Simulo los dos contextos y los dos tipos de día de la semana
  - Verifico que al terminar, me aparece un enlace para descargar el data set resultante con los datos agregados, acorde al número de días y rangos de tiempo definidos

3.1 Análisis de una solución plausible.

Los cambios en el simulador suponen un desarrollo muy concreto dado que éste sólo dispone de una interfaz. No obstante, para abordar el problema de la ejecución del simulador, facilitar su uso a un amplio número de usuarios a través de una aplicación, y para la propia integración del simulador con la misma, no existe una única alternativa.

Una potencial aproximación al problema sería preparar una aplicación de escritorio conectada a un repositorio en internet:

La aplicación se ejecutaría localmente en la máquina del usuario, se configuraría la ruta en que el usuario tiene el simulador, y la herramienta constaría de una interfaz gráfica para definir los diversos parámetros de ejecución, elegir el contexto, lanzar la simulación, y posteriormente agregar los datos de la simulación en el data set resultante. Ésta aplicación podría estar conectada con un repositorio online al que subir los data sets, y también descargarlos.

- **Ventajas y desventajas de la solución.**
  - Ventajas:
    - Si la ejecución es local, la aplicación sólo requeriría de una sencilla interfaz gráfica que lanzase por comandos de sistema, de manera transparente al usuario, la simulación que ha definido el usuario con los parámetros pasados por la interfaz.

  - La instalación del propio OpenSHS, y la eficiencia de su ejecución recaen en la configuración de la máquina del usuario y su potencia, eso ahorra desarrollos y configuraciones que podrían ser necesarios para la aplicación o el servidor.
  - El usuario puede producir data sets siempre que lo desee y disponer de ellos inmediatamente tras la ejecución, sin estar conectado a internet, y subirlos y compartirlos cuando le sea conveniente.
- Desventajas:
  - Dado que el agregado de los datos para generar el data set puede parametrizarse y convertirse en un proceso computacionalmente pesado, si el equipo del usuario carece de RAM y/o capacidad de procesamiento suficientes, podría incurrir en cuelgues, pantallazos azules o interrupciones del proceso provocadas por el propio usuario por necesitar hacer uso del equipo para otras tareas.
  - Un usuario que posee los conocimientos técnicos para modificar el código del simulador, podría afectar a sus data sets generados, si después de sus modificaciones el usuario subiese accidentalmente estos resultados al repositorio público, otros usuarios que hicieran uso de ellos podrían pasar por alto las diferencias si éstas son sutiles y conducirles a conclusiones erróneas en su investigación, con una fuente de error que podría ser muy difícil de trazar para poder corregirla.

- **Justificación del rechazo de la solución.**

  Aunque la aproximación sobre papel tiene más pros que contras, hay que considerarlos con una visión global del problema. Los puntos a favor son facilidades que pueden ahorrar ciertos desarrollos y configuraciones que podrían ser tediosos. Por otra parte, ejecutar la aplicación localmente y dar al usuario disponibilidad para usarla allí donde esté, sin necesidad de internet, es una opción interesante, pero enfocada con óptica, no está muy en línea con el problema de disponer de muchos data sets públicamente, el usuario podría no subirlos, o limitarse a dejarlos en su máquina, e incluso a usarlos y eliminarlos para ahorrar espacio, además, un usuario o equipo que pretende usar una herramienta así para un proyecto de investigación, contará presumiblemente con conexión a internet sin complicaciones, así que no supone una ventaja remarcable.

  Los contras por su parte pesan notablemente en relación con los objetivos planteados. Si el usuario no dispone de un equipo potente y topa con situaciones bloqueantes agregando ciertas cantidades de datos, sería un problema, probablemente no tendría más opción que limitar las situaciones que simula o la cantidad de datos que produce, para que su equipo pueda hacer frente al agregado de los datos, entonces la propia herramienta está limitando las posibilidades de un usuario, sólo porque no dispone de una máquina potente.

  Si cada usuario que usa la herramienta emplea su propia instancia del simulador, no habría garantías de que los data sets compartidos en la plataforma expresan el mismo modelo, y ni si quiera añadir un método de validación en el momento de la subida podría servir de garantía, aunque se recorran celda a celda los data sets, algo computacionalmente muy costoso, los datos de cada celda podrían ser resultado de un número indefinido de operaciones que el usuario ha introducido en su simulador, para sus propias consideraciones, por tanto data sets aparentemente válidos, podrían contener datos que modelan cosas totalmente distintas a otros, llevando a quién los use a conclusiones erróneas, y ésto iría en contra de lo que se propone la herramienta.

3.2 Solución propuesta

La propuesta a desarrollar será una plataforma web con un servidor para la ejecución del simulador porque cumple mejor todos los requisitos enumerados. Ésta aproximación permite que un número potencialmente muy amplio de usuarios hagan uso del simulador, sin importar la potencia de sus equipos. Por otra parte, al estar el simulador instalado en la máquina del servidor, todos los usuarios que generen data sets, y hagan uso de los data sets subidos a la plataforma, sabrán que son formalmente iguales en contenido, porque todos pasarán por el mismo proceso de agregado. La plataforma web contará con una base de datos, que permite a los usuarios registrarse y subir sus data sets, para bajarlos de nuevo en el futuro, y no tener que conservarlos siempre en su máquina, así como disponer de los data sets subidos por otros usuarios. El coste que añade usar un servidor para la ejecución se limita a tener que preparar la máquina que lo ejecute con un entorno Python, e instalar Blender para que el servidor pueda lanzar el simulador.

3.3 Plan de trabajo

Para desarrollarla se propone un plan de desarrollo por fases y en cascada.

- Fase 0. Introducción al ciclo de ejecución del OpenSHS y corrección de errores. Antes de empezar cualquier desarrollo era importante empaparse del funcionamiento del simulador, no sólo como herramienta en sí, también de su funcionamiento a nivel de código. Entender la forma en que está construido, cómo funciona el ciclo de ejecución del simulador, qué clases contienen qué funcionalidad, así como comprobar si hay errores no detectados por los creadores.

- Fase 1. Diseño de la nueva herramienta. Puesto a punto el simulador, y conociendo en detalle cómo funciona, se procede a diseñar la nueva herramienta. Entender el ciclo de ejecución de las simulaciones permite plantear varias opciones para abordar la integración con la futura plataforma, y anticipar qué características debe tener el servidor que lo ejecute.

- Fase 2. Desarrollo de los cambios para el OpenSHS, prepararlo para la nueva plataforma, y creación de la misma. Esta fase ocupa todos los desarrollos necesarios para materializar la solución. Pimero los cambios necesarios en el simulador para etiquetar los data sets con niveles de obtrusividad. A continuación la plataforma web, que contiene la interfaz para lanzar la simulación y una sección destinada a consultar, y descargar, los data sets propios y de otros usuarios subidos a la base de datos no-sql de *Firebase*, se incluye un pequeño apartado explicando cómo se configuró la base de datos. Y por último se describe el desarrollo del servidor, haciendo uso de los cambios introducidos en el simulador para ejecutarlo más fácilmente y preparando una API con Express para recibir peticiones desde la página web.

- Fase 3. Implantación y pruebas. Para poner en producción la herramienta habrá que prepara una máquina con las instalaciones y configuraciones de sistema necesarias para alojar el servidor, y que éste pueda, a su vez, ejecutar el simulador. Si no se desea dedicar parte de la capacidad de la máquina a sostener un servidor para la aplicación web, se pueden buscar alternativas para la puesta en producción de la web. Para probarlo se seguirán los estándares de los tests de aceptación de usuario y así verificar que la plataforma recoge toda la funcionalidad requerida.

CAPÍTULO 4
Diseño de la solución

Este capítulo aborda en detalle los principios de diseño del sistema y sus componentes. Viene analizado en los varios niveles de profundidad, desde cómo está construido el sistema sobre sus tres componentes principales, que son el servidor, la página web y la base de datos, hasta qué arquitectura y diseño emplea cada componente internamente para garantizar que cumple con los requisitos de funcionalidad.

4.1 Arquitectura del sistema

El funcionamiento del sistema se asienta sobre los tres componentes que son los pilares de la solución. El componente central es el servidor, que procesa las peticiones recibidas desde la plataforma web para lanzar simulaciones y retorna una respuesta cuando concluye la ejecución. Por eso se ha pensado en una distribución similiar a la arquitectura MERN[1] que en la capa de datos usa una alternativa llamada *Firebase* y que está alojada en la nube de Google *Firecloud*. La figura 4.1 muestra el esquema de un sistema MERN al uso, y la variación empleada en el proyecto. Dado que *Firebase* es un servicio de Google Cloud, la comunicación entre la página web y la base de datos es independiente del servidor, lo que permite simplicar su desarrollo y concentrar sus recursos en las peticiones para ejecutar simulaciones.

- **Aplicación web**
  - Es el componente frontal de la solución. Sirve para recibir a los usuarios de la herramienta. Se compone de una página principal con información sobre el proyecto, una sección para lanzar el simulador según parámetros introducidos por formulario y una sección para consultar y descargar data sets propios y de otros usuarios. Se comunica con el servidor enviando las peticiones json para lanzar las simulaciones. En el caso de la solución, a diferencia de otros sistemas MERN, es la página web la que interactua directamente con la base de datos, sin pasar por el servidor.
- **Servidor**
  - Es el núcleo de la ejecución del simulador, contiene todo lo necesario para lanzarlo, y está constituido por una única clase que recibe las peticiones de la plataforma a través de la API de Express, procesa la petición para extraer los parámetros de simulación y agregado de los datos, y ejecuta la simulación

[1] Las siglas se refieren a MongoDB Express React Node. Para más información consultar: https://www.mongodb.com/mern-stack

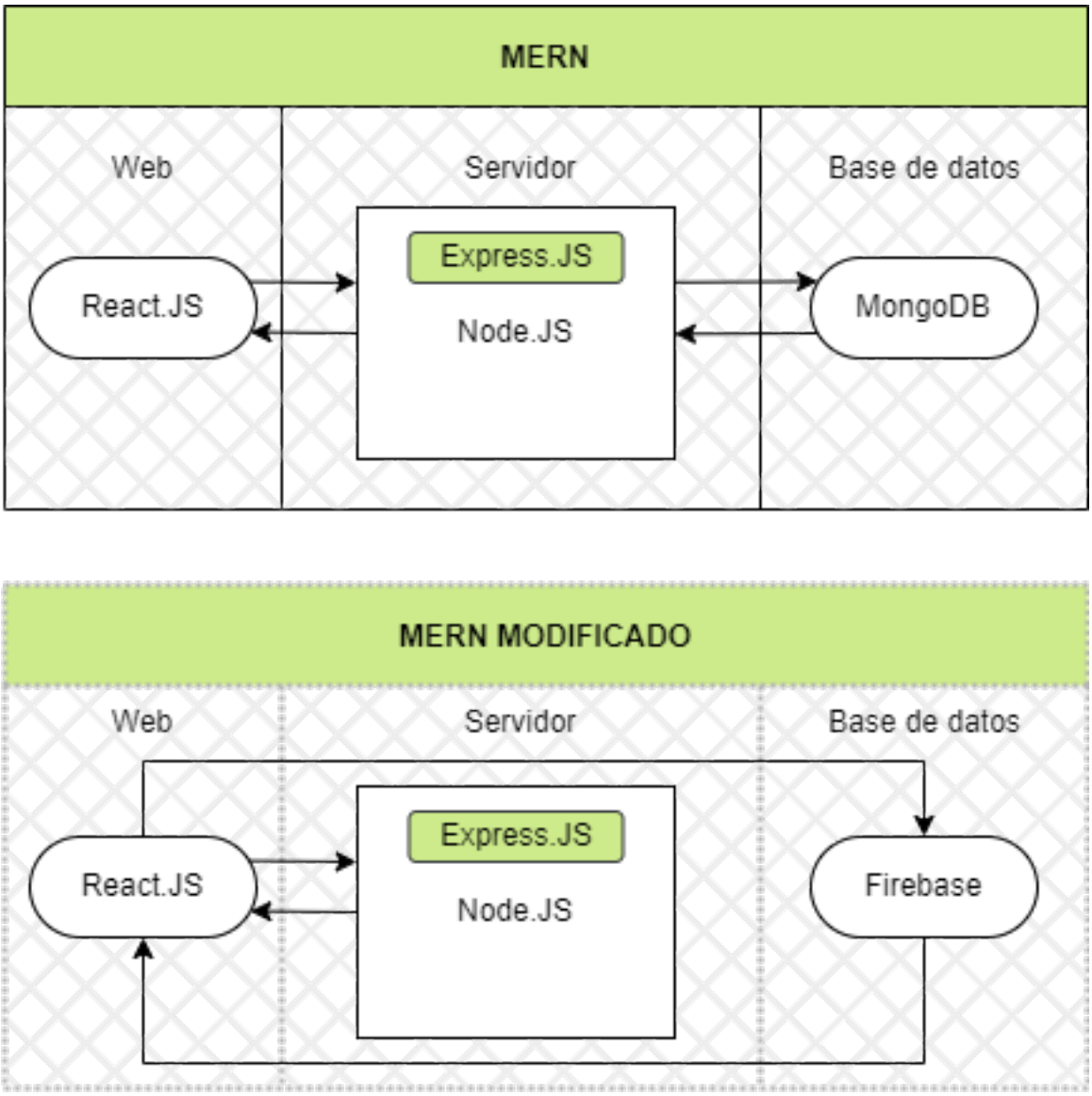

Figura 4.1: Arriba, esquema de un sistema con arquitectura MERN. Abajo: Esquema usado en el proyecto que reemplaza MongoDB con Firebase

en función a ellos, cuando el usuario ha completado la simulación de todos los contextos entonces agrega los datos según se ha recibido en la petición. Para este proyecto la interfaz Express sólo interactúa con la aplicación web, distinguiéndose del patrón MERN típico en que también comunica con la base de datos.

- **Base de datos**
  - Como se ha mencionado *Firebase* es parte de los servicios de Google *Firecloud*. Es una base de datos no-sql, con un funcionamiento similiar a MongoDB en cuanto a la forma de organizar la información, pero centrado en lo que llama "colecciones" que agrupan las distintas entidades que son manejadas por el sistema. Tiene su propia API para interactuar con ella y hacer las operaciones CRUD básicas.

4.2 Diseño detallado

- **Estructura y relaciones de la aplicación web.**

  Para desarrollar la aplicación web se han utilizado varios patrones y arquitecturas extendidos en el desarrollo de aplicaciones web. Las clases para gestionar el flujo de información originado por las acciones de usuario a través de la interfaz, usan una arquitectura que sigue el patrón Contenedor/Presentador (por legibilidad emplearemos *C-P*). Éste patrón sigue una lógica parecida al patrón Modelo Vista Presentador (MVP) de ingeniería del software; la diferencia es que en C-P no se presta tanta atención al modelo, que sólo sirve para 'caracterizar' la información entre componentes en tiempo de ejecución, mientras que en MVP todo gira entorno a los eventos que modifican el modelo.

  En la práctica ambos patrones de diseño gestionan igual la información, el Contenedor en C-P tiene la lógica de negocio que tendría el Presentador en MVP, es decir, decide **qué** datos deben mostrarse, y el Presentador en C-P al igual que la Vista en MVP decide **cómo** deben mostrarse[2]. La figura 4.2 muestra un esquema de los dos tipos de componentes que dan forma a este patrón de diseño.

  A continuación se listan las clases que tienen alguna responsabilidad en los casos de uso de la aplicación. Además de los dos tipos de componentes arriba expuestos, se verán algunos señalados con un asterisco, éstos son proveedores de contexto, que vienen definidos y listados tras los contenedores y presentadores.

  Componentes de la interfaz para la configuración y ejecución del simulador.

  - SimulatorDataContainer.tsx - Contenedor de SimulatorView.tsx. Administra los datos introducidos por el usuario en el formulario para parametrizar la ejecución, y cuando le llega el evento parra lanzarla, los proporciona a SimulatorRequestContextProvider.tsx*, se lanza la petición al servidor, y administra la respuesta del mismo para hacerla llegar hacia el usuario con la información pertinente.
  - SimulatorView.tsx - Presentador de la página donde el usuario introduce los ajustes de la simulación y la lanza. Permite la comunicación de los eventos de usuario desde los componentes más sencillos de la interfaz hasta SimulatorDataContainer y viceversa. Su componente hijo RunSettingsForm.tsx envuelve el siguiente nivel de contenedores y presentadores.

[2]Para más información sobre las características del patrón Contenedor/Presentador en aplicaciones web consultar: https://www.patterns.dev/posts/presentational-container-pattern

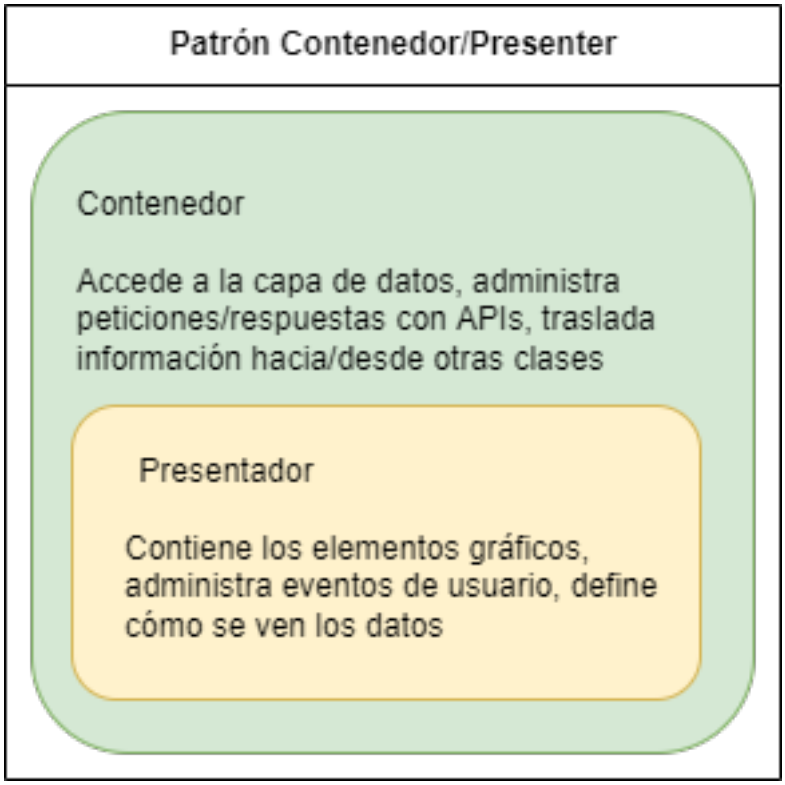

Figura 4.2: Esquema de la arquitectura Contenedor/Presentador

- RunSettingsFormDataContainer.tsx - Contenedor de RunSettingsFormView.tsx. Gestiona a nivel práctico los eventos de usuario más sencillos capturados por los componentes básicos de su presentador y permite que se escalen hacia componentes superiores una vez procesados.
- RunSettingsFormView.tsx - Presentador del formulario RunSettingsFormInput.tsx, y el panel de ayuda RunSettingsFormHelper.tsx. Permite al usuario cambiar de la vista del formulario a la vista del panel de ayuda; el primero recibe las eventos de introducción de datos para los ajustes de la simulación, el segundo tiene información sobre la implicación de cada campo en la simulación. Asciende los eventos cuando cambian los datos del formulario a RunSettingsDataContainer para que éste a su vez los siga escalando, y cambia la información que se ha de mostrar cuando viene propagada desde componentes superiores, p.e cuando el data set es devuelto por el servidor al terminar la simulación.

  Componentes de la interfaz para consulta de data sets subidos a la plataforma.

- DatasetListDataContainer.tsx - Contenedor de DataSetListView.tsx. Administra la comunicación con la base de datos a través de FirebaseRequestContextProvider.tsx*. Si el usuario está logueado pide los datos a Firebase y proporciona el elenco de data sets del usuario y de los data sets públicos a su presentador DatasetListView
- DatasetListView.tsx - Presentador de la vista de data sets. Se compone de una sencilla interfaz con dos bloques, uno para el elenco de data sets de usuarios y otro para el de data sets públicos. Cada data set viene presentado en una línea con un botón para descargarse, este evento es capturado y escalado a DatasetListDataContainer para que lo solicite a *Firebase*.

Componentes de la autenticación de usuario.[3]

- LoginDataContainer.tsx - Contenedor de LoginView.tsx. Recibe la información escalada por su componente hijo cuando se lanza la acción de login, ésta información la proporciona a LoginRequestContextProvider.tsx* para crear el usuario en base de datos.
- LoginView.tsx - Presentador del formulario de inicio de sesión. Recibe los datos introducidos por el usuario, cuando se lanza la acción de login, escala el evento para que sea enviada la petición.

  *** Componentes proveedores de contexto.**

  Para no introducir confusión sobre las relaciones entre los componentes esenciales de la aplicación, se ha omitido la definición de estos componentes durante el listado. Su función es la de agrupar parte de la lógica de negocio empleada por los Contenedores; lo particular de los proveedores de contexto es que permiten hacer uso de la lógica que en ellos se define sin pasar ninguna propiedad explícitamente por código, a cualquier componente sin importar su nivel de anidamiento, siempre que sean componentes hijos del proveedor.

  Éste tipo de componentes suele acompañar el patrón Contenedor/Presentador ya que permite deslocalizar lógica del contenedor a otras clases y mantener fácilmente su código cuando éstos crecen, o cuándo la complejidad del anidamiento de la página aumenta demasiado y se convierte en un problema hacer llegar esta lógica a los componentes anidados al final. Se define su lógica individualmente como sigue:

- SimulatorRequestContextProvider.tsx - Provee a los componentes hijos del SimulatorDataContainer de las funciones necesarias para la petición al servidor para lanzar la simulación. SimulatorDataContainer lo inicializa para que RunSettingsDataContainer, pueda disoner de éstas funciones y ejecutar la petición. Para la comunicación con el servidor utiliza ISimulatorRepository.ts, una interfaz interna del repositorio ApiSimulatorRepository.ts que tiene exclusivamente el método asíncrono que lanza la petición HTTP al servidor de Node.Js, de ésa manera sólo ésta clase conoce los detalles de conexión.
- FirebaseRequestContextProvider.tsx - Provee el contexto para las peticiones a la base de datos de *Firebase* a los hijos de DatasetListDataContainer. Proporciona todo lo necesario para realizar las operaciones CRUD básicas para la colección de data sets. Al igual que su homólogo para la comunicación con el servidor, utiliza una interfaz interna, en este caso IFirebaseRepository.ts, del repositorio ApiFirebaseRepository.ts para hacer la lectura y creación de data sets, asociados a usuarios.
- LoginRequestContextProvider.tsx - Este contexto es compartido por los tres contenedores de los módulos de autenticación, proporciona la lógica necesaria para hacer peticiones de alta de usuario, de inicio de sesión y restablecer la contraseña a partir de un correo. También emplea una interfaz y un repositorio: IAuthenticationRepository.ts y ApiAuthenticationRepository.ts para lanzar las peticiones a la base de datos.

  Es preciso añadir que la decisión de separar entre estos dos últimos proveedores de contexto, asi como sus repositorios de comunicación con*Firebase* es para seguir cumpliendo con el principio de responsabilidad única, de modo que

[3]La aplicación permite registrarse, iniciar sesión con una cuenta ya creada, y actualizar las credenciales si se pierden. Para no resultar redundantes en la exposición, se omiten SignupDataContainer y su presentador, y RecoveryDataContainer y su presentador, y se exponen sólo el contenedor LoginDataContainer y su presentador LoginView por la mención en el punto anterior, ya que los otros dos módulos son casi idénticos.

hay un repositorio, y entonces un contexto, sólo para el CRUD de usuarios, y otro repositorio y otro contexto que lo usa sólo para el CRUD de data sets.

- Diseño de la base de datos
  Firebase organiza sus bases de datos entorno a colecciones. Las dos entidades que constituyen conlecciones en la solución son: Usuarios y Data sets. Un usuario puede tener muchos data sets, pero cada data set pertenece a un único usuario. Ésta relación uno a muchos viene materializada por el ID del data set: cada Usuario posee una lista de ID's de los data sets que ha generado. El diagrama de clases de la figura 4.3 muestra la relación entre las dos entidades.

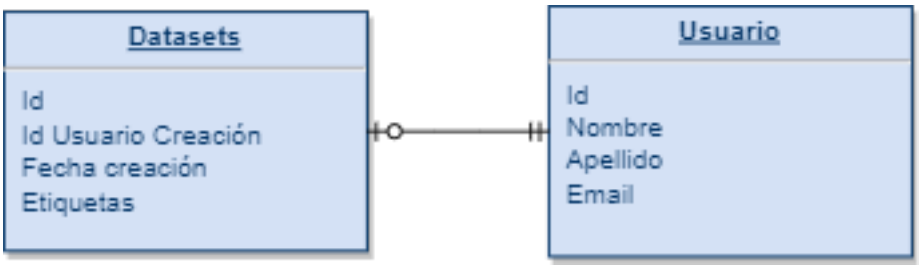

Figura 4.3: Diagrama de clases de la base de datos en *Firebase*

- Diseño del servidor
  El servidor de Node.Js tiene una estructura muy sencilla: consiste en una única clase server.js que declara, utilizando los métodos de la API de Express, una función que actúa de *end-point* para recibir las peticiones desde la plataforma web. Procesa la petición para definir el arranque del simulador y ejecuta secuencialmente una simulación para cada contexto, mañana y tarde, y para cada tipo de día entre semana o fin de semana, cuando el usuario ha concluido todas las simulaciones, agreda los datos en el data set resultante y lo envía a la plataforma como parte de la respuesta.

4.3 Tecnología utilizada

- **Lenguajes de programación.**
  - Python - Lenguaje en que está escrito todo el OpenSHS y los archivos de configuración de Blender que éste utiliza, todo los cambios sobre su interfaz y los cambioas necesarios para agilizar la ejecución del mismo se han desarrollado en python.
  - TypeScript - Lenguaje de programación construido sobre JavaScript usado para desarrollar la plataforma, su uso es interesante porque permite clases estáticas y objetos para modelar la información compartida entre los componentes y acotar posibles errores y malas prácticas al desarrollar aplicaciones y mejorar la mantenibilidad de las mismas. En particular en la aplicación ha servido para modelar la comunicación de las peticiones entre la plataforma y el servidor, y entre la plataforma y la base de datos.
  - JavaScript 'vanilla' - El servidor que ejecuta el simulador está programado en Node.js, utilizando JavaScript sin ninguna capa corriendo por encima.
- **Frameworks y librerías.**
  - React.js - Librería de JavaScript empleada para desarrollar la aplicación web por las facilidades que da en el desarrollo de interfaces de usuario.

  - Express - Framework de Node.js empleado para desarrollar fácilmente una API de comunicación con la aplicación web, introducir una capa de características durante el procesado de las peticiones para gestionar su información, así como configurar los ajustes de la comunicación con la plataforma (puerto, IP, protocolos, etc).

- **API's**

  - Blender Python API - Interfaz para integrar en la ejecución de Blender, modulos y scripts de Python. Para hacer posible la ejecución del simulador por parte del servidor se ha escrito un script python que usaba métodos proporcionados por la API y reducía el uso de comandos por consola.
  - Node Powershell API - Interfaz que permite a un servidor Node ejecutar comandos de Powershell. Es necesaria para lanzar el script python que arranca todo el simulador al completo.

CAPÍTULO 5
Desarrollo de la solución

Como se mencionara en el plan de trabajo del capítulo sobre el análisis del problema, la solución ha seguido un desarrollo por fases y en cascada. Durante la fase previa de toma de contacto con el OpenSHS, se encontraron algunos errores que era preciso subsanar. Cuando se generaban los data sets temporales, el simulador creaba un archivo .csv con un nombre que indicaba qué día es, qué contexto y la fecha y la hora de inicio la cuál venía en formato HH:MM:SS, pero el uso de los dos puntos en nombres de archivo en sistemas windows provocaba un error del sistema, y rechazaba sobreescribir el archivo temporal que se escribe con los datos de la simulación, haciendo que si se pasara por alto se pudiera pisar el resultado de una simulación con el de la siguiente y perder los datos, así que se corrigío el nombre usado para el archivo temporal que se emite tras la simulación utilizando sólo guiones bajos como separadores.

Una de las cosas a probar en esta toma de contacto era el funcionamiento del simulador a través de Blender. Para iniciar el simulador había que ejecutar el script "openshs", introduciendo una instrucción por termina, pero eso no ejecuta el simulador en sí, si no que abre el entorno de Blender, cargando el modelado del escenario, y el motor de juego y su programación, además de algunos scripts. Desde ahí el usuario lanza manualmente el simulador bien por la consola interna de Blender-Python, bien usando la interfaz gráfica del entorno de Blender. La figura 5.1 muestra la ejecución del simulador lanzando el script desde PowerShell y la interfaz gráfica de Blender que se le muestra al usuario.

Para buscar una forma más fácil de ejecutar el simulador, y posibilitar ejecutarlo desde un servidor, era preciso exportar cada escenario, el de la mañana y el de la tarde como archivos ejecutables que no requiriesen de la intervención del usuario con varios comandos por consola, y tampoco se debería abrir el editor de Blender. Sin embargo al usar las herramientas de compilación y exportación de Blender, todo parecía ir correctamente, pero por algún error durante la exportación que el entorno no reportaba, al arrancar la simulación con el ejecutable resultante, todas las interacciones programadas en el escenario (abrir puertas, encender luces, tumbarse en la cama, ...) dejaban de funcionar, incluso al apretar la tecla que sirve para abrir la interfaz interna del simulador y elegir actividades no respondía.

Ésta circunstancia limitaba mucho las posibilidades de la herramienta y ponía en cuestión incluso su viabilidad si el proyecto giraba entorno al OpenSHS, porque tratar de integrar un software así en un servidor estaba descartado. Para hacer posible compilar los entornos en ejecutables se barajó traer el simulador a versiones más nuevas de Blender, pero OpenSHS funciona solamente con la versión 2.7 porque utiliza el motor de ejecución *Blender Game Engine*, y la v2.7 fue la última versión de editor que lo soportaba, a partir de ahí el motor se dejó de mantener por los creadores del editor por lo que ac-

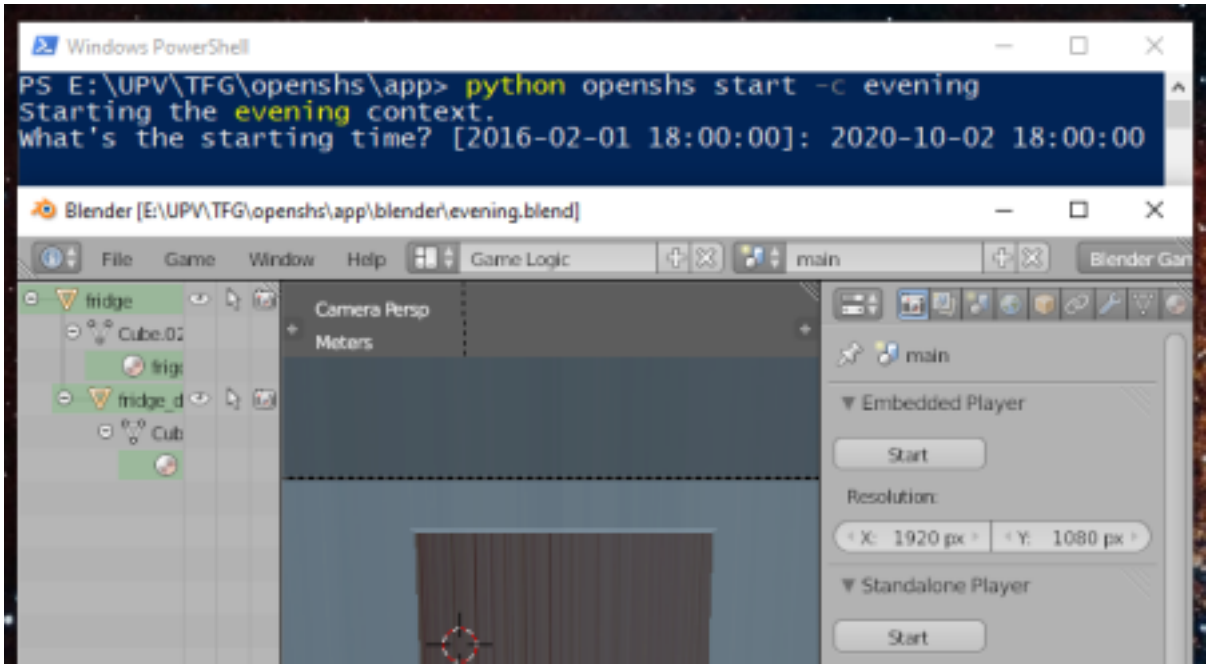

Figura 5.1: En la captura se puede ver la ejecución del script por consola y el entorno de Blender

tualizar el simulador implicaría refactorizarlo casi por completo, modelado 3D incluido, y eso se salía de las posibilidades del proyecto.

Por tanto, si la opción del ejecutable no era posible, habría que buscar la manera de ejecutar línea a línea todo lo necesario para lanzar una simulación. La documentación de la Blender Python API ofrece una equivalencia entre la instrucción que se usaría en una terminal para realizar una acción del editor de blender, y su traducción a una llamada a la API para realizar esa misma acción pero dentro de un script de Python. Durante el reconocimiento del código del simulador, se ha podido observar que el script principal del simulador que ejecuta la carga del escenario y abre el entorno Blender, usa una función que ejecuta secuencialmente las instrucciones PowerShell que se le pasen como cadenas de carácteres, inicialmente a esa función se pasa la instrucción de PowerShell que ejecuta blender precargando un escenario, y como la documentación de la API reporta, a ésta función se le puede pasar como argumento un script Python para lanzarlo inmediatamente después, añadiendo lo siguiente:

```
--python <Script.py>
```

Así que se escribió un sencillo script de python con las llamadas para lanzar un escenario precargado y para cerrar el entorno de blender, y se añadió a los argumentos de la función como se ha indicado.

Quedaba resuelto el problema de pasar por el editor de Blender para iniciar el simulador, pero aún había que interactuar por consola para introducir fecha y hora, entonces se añadieron dos argumentos, para que admitiera por línea de comandos al ejecutarse el script "openshs" un espacio para la fecha y un espacio para la hora:

```
@click.option('--date', '-dt', help='Which day to start.')
@click.option('--hour', '-h', help='Which hour to start.')
```

Con estas líneas, se puede quitar el diálogo que solicita fecha y hora, y combinado con los cambios descritos hasta ahora, sólo es necesario utilizar una línea de terminal para lanzar una ejecución del simulador parametrizada.

El siguiente paso en el desarrollo era implementar los tres nuevos botones que deben etiquetar el nivel de obtrusividad deseado. Toda la lógica que gestiona la interfaz interna se encuentra en el script "ui.py", en ella se añadieron las líneas necesarias para visualizar cada botón nuevo, y el controlador del evento para añadiese la etiqueta correspondiente al data set, además hubo que añadir al archivo de configuración del motor de juego de blender "init.py" la nueva columna de obtrusividad para que al generar el data set la etiqueta se mostrara correctamente, de lo contrario una excepción no controlada interrumpía el proceso. Así el simulador queda listo para etiquetar niveles de obtrusividad, y su ejecución se ha simplificado a una línea de terminal.

Para desarrollar la plataforma se ha utilizado como se comentaba en la sección de diseño, una arquitectura de componentes Contenedor/Presentador, para administrar como navegan los eventos de usuario y la información entre componentes. Todos los elementos usados en la página son componentes creados personalmente, no se han usado de paquetes de terceros. En el desarrollo de todos los componentes se ha puesto mucha atención a que sean autocontenidos, de manera que las vistas de la web, crecen siempre en un anidamiento que es fácilmente mantenible, cada capa debería ser desconocedora de lo que tienen las de arriba, más allá de lo estrictamente necesario para su funcionamiento, y también de lo que usan las de abajo, más allá de lo que le pidan a ella directamente. Utilizando *Higher Order Components* (HOC)[1], en combinación con la arquitectura descrita con anterioridad se ha logrado ese acople por capas, minimizando dependencias, y eso ha permitido reciclar componentes, minimizar desarrollos y sentar la base de una mantenibilidad futura más sencilla incluso para quién no conociese el proyecto en detalle. Para la comunicación con servicios externos a la web, tanto el servidor como la base de datos, se han usado API's internas para que los componentes proveedores de contexto no conociesen nada del envío HTTP, salvo la función proporcionada por la interfaz. Estas API's estaban compuestas por repositorios y sus interfaces, distinguidos entre ellos por el tipo de petición que realizaban.

El desarrollo del servidor se inició cuando la web tenía lista la estructura suficiente para rellenar la petición HTTP con los parámetros para simulación. Cuando se completó la funcionalidad para recibir las peticiones de la app todo funcionó en el primer intento, pero al añadir el envío de datos desde la web, entonces el servidor empezó a bloquear las solicitudes de la web por error de configuracón en las cabeceras de la misma, porque Express usa una configuración en comunicaciones sin envío de datos, y para la comunicación creada inicialmente para probar el funcionamiento del *end-point* dentro del servidor funcionaba, pero cuando desde la web se intentaban enviar datos, la petición hacía saltar los mecanismos de seguridad en el Intercambio de Recursos de Origen Cruzado (CORS, sus siglas en inglés). Pese a ello, el problema se solventó con unas cuantas líneas de configuración, explicitando el tipo de recursos que se podían esperar desde la web, y no existieron más complicaciones en la parte del servidor.

[1] Más información en https://www.patterns.dev/posts/hoc-pattern/

CAPÍTULO 6
Implantación y pruebas

Es preciso introducir el capítulo señalando que no ha sido posible una puesta en producción del proyecto por falta de tiempo, no obstante, si estaban proyectados los pasos ideados para hacerlo posible.

Para poner en marcha el servidor se había proyectado usar Docker y una imagen de Windows Server 2016. Es importante la versión porque será necesario instalar, además del entorno Python, Blender v2.7, para que pueda funcionar el simulador, esa versión es de 2016 por eso se elige esa imagen de Windows para evitar problemas de compatibilidad.

En cuanto a levantar la plataforma hay varias opciones. Para empezar se había considerado GitHub Pages, al ser un servicio gratuito de *hosting* de páginas web, muy sencillo de configurar, y al ser parte de la plataforma donde estaría el código sería más fácil desplegarla. Pero GitHub sólo mantiene sitios estáticos, que no estén sujetos a peticiones sobre un servidor para cambiar la información que muestran en pantalla. Por otra parte está Heroku, que ofrece opciones interesantes para páginas web, aunque hay que pagar para muchas funcionalidades también. Por supuesto otra opción es levantar un servidor dedicado a la página web, podría hacerse en la misma máquina de Windows Server que se utilice para levantar el servidor del simulador.

Como la aplicación sí que está desarollada aunque no esté en producción, se han podido hacer pruebas para comprobar que los requisitos se cumplen, y también algún control para probar cómo se comporta la solución en situaciones que podría conducir a errores o caídas.

Tabla 6.1: Tabla de las pruebas de criterios de aceptación realizadas.

Prueba de funcionalidad	Entrada	Salida esperada	Salida obtenida
Lanzar el simulador	Entrada vacía	Data set correcto	Data set correcto
Lanzar el simulador	Entrada de usuario	Data set correcto	Data set correcto
Etiquetar obtrusividad	Entrada de usuario	Data set etiquetado	Data set etiquetado
Descargar data set	Acción de usuario	Archivo descargado	Archivo descargado
5 peticiones simult.	5 entradas	5 data sets	5 data sets
10 peticiones simult.	10 entradas	10 data sets	10 data sets
25 peticiones simult	25 entradas	25 data sets	25 data sets

Como expone la tabla, se ha probado toda la funcionalidad planteada en los objetivos. Las pruebas con varias peticiones tenían como fin provocar algún corte de conexión en los procesos que llevaran más tiempo esperando, la idea de esa prueba de "estrés" viene porque haciendo pruebas de la navegación y la funcionalidad, se hizo evidente que no se estaba controlando cómo se administraba el acceso al propio simulador como un recurso

crítico, si las peticiones se encolaban podría darse que hubiese problemas y errores en las últimas en llegar, cuando se acumulaban demasiadas peticiones, y entonces se hizo necesario probar qué pasaba con las peticiones que tardaban más en servirse. Durante las pruebas no hubo ningún *time out* ni interrupción del servicio, teniendo siempre presente que ha sido en un entorno local, y que por tanto éste tipo de mediciones no son totalmente fieles a la realidad, pero en cualquier caso de las pruebas se puede extraer rápidamente que haya o no cortes de servicio, desde el punto de vista del usuario es visto como una pérdida de tiempo tener que esperar indefinidamente a poder usar el simulador, éste es uno de los puntos que queda pendiente, para completarla como una herramienta funcional en producción.

CAPÍTULO 7
Conclusiones

La valoración del proyecto en orden a los objetivos enunciados al principio es positiva, si vemos cómo cumple la herramienta respecto al problema enunciado en el capítulo 3:

- Permite generar data sets etiquetados con el nivel de obtrusividad.
- Permite un acceso más sencillo a la ejecución del simulador a través de una interfaz.
- Puede hacer más accesible públicamente data sets variados.
- Da garantías sobre el contenido y el método de obtención de los data sets.

Y si nos ceñimos a los requisitos de funcionalidad propuestos para el simulador y la plataforma en ese mismo capítulo:

- Como usuario del OpenSHS, quiero poder etiquetar con el nivel de obtrusividad deseado distintas situaciones, durante la simulación de los varios contextos posibles
- Como usuario del OpenSHS, quiero disponer de una plataforma con una interfaz para ejecutar el simulador, y subir y consultar data sets

La herramienta que ha resultado del desarrollo da solución a los problemas y cumple ambos criterios de aceptación como ha quedado recogido en las pruebas, siguiendo esos criterios se puede decir que cumple todos los objetivos propuestos. Si bien es cierto que aún se tiene que poner en producción, y que antes de permitir que sea escalada a un acceso multitudinario, es imprescindible abordar el problema de las ejecuciones concurrentes, respecto a lo cuál una posible solución sería aprovechar las posibilidades que da javascript para administrar varios hilos de ejecución, y estudiar la manera en que se pudiera crear un hilo por cada petición y que accedieran a distintas estancias del simulador, para así evitar las colas.

El balance que se puede hacer del desarrollo es que ha sido bastante desequilibrado en complejidad. Los desarrollos menos extensos como eran aquellos sobre el simulador, han entrañado más problemas por ser sobre una tecnología dada por obsoleta por sus creadores, como es el motor gráfico de Blender. Además el simulador no tenía mucha actividad en su propia página de GitHub, con lo que no ha sido posible contar con la ayuda de otros usuarios en momentos puntuales. De todas las tecnologías empleadas en el proyecto, aquellas que tenían que ver con el simulador me eran las más desconocidas. Es cierto que Python se ve en el grado en Estructura de Datos y Algoritmos en segundo, y más adelante en Algorítmica y en Sistemas Inteligentes (para los alumnos de mención

en computación como es mi caso), pero Python usado para interfaces por consola, y para programar el motor de un simulador como éste, era algo nuevo para mí. No conocía muchas de las librerías que se usaban para hacer funcionar el código, y ha sido realmente difícil trazar el error en algunos momentos en que uno de mis cambios provocaba fallos en el simulador, en éste sentido he aprendido mucho de Python, de sus posibilidades más allá del procesado de grandes volúmenes de datos, o su potencia para escribir algoritmos, era una faceta del lenguaje que desconocía, y además me ha servido para aprender sobre la API que integra instrucciones de Blender con Python, algo que desconocía totalmente antes empezar el desarrollo.

Desde el punto de vista tecnológico el proyecto usa multitud de tecnologías, algunas vistas en el grado también, pero he tenido necesidad de refrescarlas, por ejemplo en los pasos a seguir para montar el servidor de Node.Js y conectarlo con la plataforma web, el problema con las cabeceras de orígenes cruzados fue muy interesante de afrontar porque profesionalmente estoy acostumbrado a que ésto ocurra en ocasiones en el despliegue de proyectos nuevos, cuando se levanta el *front-end* y no se ha configurado adecuadamente el servidor, pero nunca soy yo quién entra al servidor a configurar esas cabeceras, es mi responsable de equipo, y hay que añadir que nunca había progamado usando Express para definir una API o un *end-point*, por eso fue un momento de ponerme en la piel de un administrador y buscar las razones de que mi configuración no estuviese funcionando, rastrear qué estaba fallando y documentarme bien sobre las tecnologías en acción en el momento para encontrar una solución.

Node.Js y Express son las dos tecnologías más enriquecedoras profesionalmente de lo aprendido en el pryoecto, porque en mi puesto de trabajo actual me estoy perfilando más hacia la arquitectura de software, y concretamente la arquitectura de soluciones basadas en aplicaciones web, por eso me fijé en la arquitectura MERN para el sistema, y por eso elegí Express como capa de características para administrar la comunicación con la web. Tras haber hecho posible levantar un servidor como el que se ha desarrollado en este trabajo ahora puedo decir que no soy nuevo en el campo, que ya he tocado un servidor Node con API de Express que era parte de una solución más grande, y he encontrado y superado un problema que es punto de fallo común en este tipo de despliegues. El proyecto también me ha servido de terreno de pruebas de varias ideas y conceptos de desarrollo web, que en mi trabajo no siempre tengo tiempo de desarrollar y profundizar por los tiempos de los sprints, y que llevaba tiempo queriendo utilizar y experimentar con ellos para comprobar cómo facilitaban el desarrollo, que opciones daban a mejorar la mantenibilidad, qué posibilidades dan a nivel de arquitectura.

7.1 Relación del trabajo con los estudios cursados

En el trabajo desarrollado hay muchos items y conocimientos que vienen sacados directamente del grado. De lo aprendido en el grado he podido aplicar una buena metodología de diseño a nivel de arquitectura, son los conocimientos de asignaturas como Ingeniería del Software, los que me han permitido saber cómo tenía que enfocar el análisis del problema, qué diagramas podía usar para representar según qué aspecto de la solución, del diseño, del modelo, etc.

Luego hay que tener en cuenta que para el servidor y la aplicación se ha empleado JavaScript, las bases del lenguaje se adquieren en la asignatura de Tecnologías de la Información y la Comunicación, es ahí donde se hacen las primeras prácticas con JavaScript, y también de esa asignatura tenía los conocimientos, al menos los más básicos, para levantar el servidor de Node.Js, aunque luego haya tenido que buscar guías para configurarlo con Express y hacer comunicación mediante API en lugar de sockets, pero también de

los conocimientos puestos en práctica con esa asignatura he podido proyectar un posible despligue del servidor usando Docker y una imagen de Windows.

En general, como intenta plasmar la memoria, el producto final es resultado de la combinación de varios ámbitos de conocimiento de la informática, y varias tecnologías y conceptos adquiridos durante el grado, algunos con más peso porque los pude profundizar más a partir de tercero, otros porque los he adquirido más tarde como profesional, pero en cualquier caso en la base de todo ello están los aprendizajes hechos durante mis estudios, que produndizados en mayor o menor grado, me dan herramientas para enfocar adecuadamente los problemas que me puedo encontrar como ingeniero.

CAPÍTULO 8
Trabajos futuros

Por su puesto a raíz del desarrollo realizado se pueden imaginar nuevos desarrollos:

- Permitir hacer modificaciones sobre la salida del simulador: número de columnas, sensores usados, nuevas etiquetas a registrar.
- Añadir nuevas funcionalidades al simulador: medición de niveles de consumo energético y de agua, o cantidad de alimentos dentro del frigorífico, etc.
- Implementar nuevos electrodomésticos y aparatos de uso que registren actividad del usuario.

Éstas son posibles mejoras que se podrían hacer para aumentar el valor de la herramienta. También sería interesante, añadir trazas internamente al simulador para seguir mejor el flujo de información, y trazar posibles errores si se implementan algunas mejoras, aunque respecto a las mejoras sí que es necesario decir que antes de abordar cualquier cambio sobre el simulador, sopesaría hacer antes la refactorización que sea necesaria para poder actualizar el software que utiliza a las versiones más recientes de Blender, y posteriormente resolver el problema de la ejecución de las ejecuciones simultáneas del servidor.

Vista toda la solución en su conjunto, es fácil imaginar que con el auge de los sistemas que usan inteligencia artificial, y el auge del internet de las cosas, habrá en los próximos años varios ámbitos que encuentren los problemas que encuentra el campo de la Inteligencia Ambiental, y por tanto una línea de desarrollo interesante sería extender las posibilidades de la herramienta a la generación de data sets para modelar otro tipo de sistemas, o fenómenos o procesos, y que se documente que métodos se usan para generarlos y sean consultables, e incluso que la herramienta sirva para compartir resultados de aplicar ciertas técnicas de aprendizaje automático sobre determinados data sets que se pueden encontrar en ella, todas esas opciones abren nuevas líneas no para mejorar la herramienta sino para extender su alcance a otros ámbitos.

Bibliografía

[1] Zhang, H., Liu, Y., Wang, C., Fu, R., Sun, Q., & Li, Z. Research on a pedestrian crossing intention recognition model based on natural observation data. *Sensors*, vol. 20-6, febrero, 2020. https://doi.org/10.3390/s20061776

[2] Giuseppe Amato, Davide Bacciu, Stefano Chessa, Mauro Dragone, Claudio Gallicchio, Claudio Gennaro, Hector Lozano, Alessio Micheli, Gregory M.P. O'Hare, Arantxa Renteria and Claudio Vairo. A Benchmark Dataset for Human Activity Recognition and Ambient Assisted Living. *Ambient Intelligence- Software and Applications – 7th International Symposium on Ambient Intelligence (ISAmI 2016). Advances in Intelligent Systems and Computing*, vol. 476, Springer, Cham, pp 1–9, mayo, 2016.

[3] Semwal, V.B., Gupta, A.& Lalwani, P. An optimized hybrid deep learning model using ensemble learning approach for human walking activities recognition. *J Supercomput*, vol. 77, pp.12256–12279, noviembre, 2021. https://doi.org/10.1007/s11227-021-03768-7

[4] Azkune, G.; Almeida, A.; López-de-Ipiña, D.; Chen, L. Combining Users' Activity Survey and Simulators to Evaluate Human Activity Recognition Systems. *Sensors* ,vol. 15, pp.8192-8213, abril, 2015 https://doi.org/10.3390/s150408192

[5] Jeffrey W. Lockhart and Gary M. Weiss. Limitations with activity recognition methodology & data sets. *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication (UbiComp '14 Adjunct)*, pp.747–756, septiembre, 2014

[6] William Young, Amir Sobhani, Michael G. Lenné, Majid Sarvi. Simulation of safety: A review of the state of the art in road safety simulation modellings. *Accident Analysis & Prevention*, vol. 66, pp.89-103, enero, 2014 https://doi.org/10.1016/j.aap.2014.01.008

[7] Belhaj, Mouna & Kebair, Fahem & Ben Said, Lamjed. Modelling and simulation of human behavioural and emotional dynamics during emergencies: A review of the state-of-the-art. *International Journal of Emergency Management*, 2015

[8] Hong, Seung Wan, and Yun Gil Lee. The Effects of Human Behavior Simulation on Architecture Major Students' Fire Egress Planning. *Journal of Asian Architecture and Building Engineering*, vol. 17, no. 1, pp.125-132, octubre, 2018 https://doi.org/10.3130/jaabe.17.125

[9] Hong, Seung Wan, et al. Effects of Human Behavior Simulation on Usability Factors of Social Sustainability in Architectural Design Education. *Sustainability (Basel, Switzerland)*, vol. 12, no. 17, pp. 7111–, 2020 https://doi.org/10.3390/su12177111

APÉNDICE A
Objetivos de Desarrollo Sostenible

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

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

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

El TFG presentado guarda notable relación con los ODS dado el momento en que nos encontramos hisóricamente desde un punto de vista social, económico y tecnológico. Como se comenta en la motivación de la memoria, en Europa, EE.UU y algunos países asiáticos, se está viviendo un declive demográfico como consecuencia de la baja natalidad. Los factores que han influido para llevar a ese punto son varios, la combinación de dos salarios en las parejas jóvenes desde la introducción de la mujer al mercado laboral, que se combina sin embargo con un gran sentimiento de incertidumbre entre las personas en edad de formar pareja y tener hijos.

La media de hijos por mujer en Europa ha llegado a bajar hasta 1.2, mínimos de tiempo de la post-guerra en España, y muy por debajo del mínimo para asegurar la continuidad biológica, sin entrar en la vista a muy largo plazo y las fatídicas consecuencias que ésto tiene, las primeras ya se están haciendo notar, la proporción de personas mayores de 65 ha aumentado mucho en proporción a otros rangos de edad, el Levante de junio de este año reportaba que casi un tercio de las personas mayores de 65 años viven solas, uno de los factores que señala el cotidiano son las viviendas no adaptadas y las barreras arquitectónicas, que aumentan el factor de soledad por las dificultades que añade en la vida cotidiana a personas de esa edad, o más mayores. Éste es el ámbito en que la Inteligencia Ambiental, y por tanto proyectos como el presentado en el TFG pueden marcar la diferencia, añadir herramientas para mejorar los desarrollos de sistemas de reconocimiento de la conducta humana permitirá aportar más rápidamente soluciones domóticas que cuiden mejor la **salud y el bienestar** de esas personas, como agentes físicos asistenciales, con toma de decisiones basada en el aprendizaje automático, o sencillamente a través de sistemas de monitorización del individuo, de la manera menos intrusiva posible, pero que permite a profesionales de la salud poder hacerse cargo de emergencias rápidamente.

Otro gran fenómeno que se está acelerando es la revolución del sector del automóvil. Algunos medios han dejado caer un tercio del PIB mundial se podría ver afectado, y es que no sólo se han puesto en el punto de mira los motores a combustión, como se introducía también en la motivación, se proyectan gigantescas reestructuraciones del transporte urbano, del sector energético, de varios sectores de la industria y hasta de la arquitectura en sí misma, ya hoy si uno presta un poco de atención a las obras que pueda haber alrededor de su barrio, o de su ciudad, puede observar fácilmente proyectos de casas "autogestionadas", es decir diseñadas de tal manera, desde los materiales empleados, la orientación de la fachada y las ventanas, las alturas de los techos, etc que logran reutilizar una grán cantidad de los propios recursos que la casa consume, y minimizar gastos. En un entorno así, un sistema inteligente que aprenda las costumbres que tienen en particular los habitantes de la casa, podrá ayudar economizar aún más muchos aspectos, ayudando a hacer un **consumo responsable**, mediante recomendaciones al usuario o controlando cosas como el apagado o encendido de los aparatos de aclimatación de estancias del hogar, por ejemplo.

Además hay que considerar que una casa así se enmarcaría en el contexto de una ciudad inteligente, con un uso de los recursos más **sostenible**, y donde la energía se podría redirigir a lugares concretos donde sea necesaria y reducir la que se destina a sitios donde no se está consumiendo tanto, reduciendo la base de la producción de electricidad que se tiene que hacer de manera continua. Evidentemente será preciso superar muchas barreras tecnológicas en el camino, también hacer frente a dilemas y éticos sobre el uso de sistemas inteligentes en hogares, o estudiar la conducta humana en ciertos contextos, entre otros, pero por ello las instituciones y gobiernos que han empujado en el recorrido de

la Agenda 2030 deben poner los límites a quienes querrán aprovecharse de la situación para que no permitan que el esfuerzo se desmerezca y acabe en nada, y si lo permiten entonces deberemos ser el resto quienes exijan cuentas porque lo que ocurra con el bienestar de los que hoy son mayores, y el trato que le demos al entorno en que hoy vivimos, es la herencia que dejaremos a quienes nos sigan.