UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Graduado en Ingeniería del Software

Herramienta para el modelado, simulación y análisis del impacto de las medidas preventivas frente a la COVID-19.

Tool for modeling, simulation and analysis of the impact of preventive measures against COVID-19.

Realizado por
Stela Ávila Muñoz

Tutorizado por
Eduardo Guzmán de los Riscos

Departamento
Lenguajes y Ciencias de la Computación
UNIVERSIDAD DE MÁLAGA

MÁLAGA, junio de 2021

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADO EN INGENIERÍA DEL SOFTWARE

Herramienta para el modelado, simulación y análisis del impacto de las medidas preventivas frente a la COVID-19.

Tool for modeling, simulation and analysis of the impact of preventive measures against COVID-19.

Realizado por
Stela Ávila Muñoz

Tutorizado por
Eduardo Guzmán de los Riscos

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, JUNIO DE 2021

Fecha defensa: Julio de 2021

Abstract

This TFG has been carried out in order to be able to get conclusions on the evolution of the virus that currently affects all countries in the world, such as COVID-19. From these conclusions it will be possible to obtain, for example, which restrictions are more effective and in what type of population this disease has less or more impact, but how the virus will evolve in the following weeks can also be predicted.

To do this, in this work a tool is developed to simulate geographic locations with multiple input parameters in order to adapt it to small and medium populations. In addition, this tool allows you to view the resulting data, compare different simulations carried out and even export them if necessary.

Keywords: **Agent-based modeling, MESA, COVID-19, Python**

Resumen

Este TFG ha sido realizado con el fin de poder extraer conclusiones de la evolución del virus que actualmente afectada todos los países del mundo, como es el COVID-19. De dichas conclusiones se podrá obtener, por ejemplo, qué restricciones son más efectivas y en qué tipo de población esta enfermedad tiene menos o más impacto, pero además se puede predecir cómo evolucionará el virus en las siguientes semanas.

Para realizar esto, en este trabajo se desarrolla una herramienta para simular lugares geográficos con múltiples parámetros de entrada para poder adaptarlo a pequeñas y medianas poblaciones. Además esta herramienta permite visualizar los datos resultantes, comparar distintas simulaciones realizadas e incluso exportarlas si fuera necesario.

Palabras clave: **Modelado basado en agentes, MESA, COVID-19, Python**

Índice

1 Introducción

1.1. Motivación

En la actualidad, la población mundial se está viendo afectada gravemente debido a la pandemia del COVID-19. Hay muchas personas que mueren cada día y aún más son las que se infectan. Todos los países se han visto obligados a tomar medidas que restringen las actividades de los ciudadanos para intentar frenar esta enfermedad. Estas han sido elegidas por expertos epidemiológicos, teniendo en cuenta la evolución que ha estado teniendo y la cantidad de datos que se pueden encontrar. Aunque se dispongan de todos estos medios para intentar predecir su evolución y por tanto para la toma de decisiones, a veces parece ser muy difícil o imposible. Por esto, resulta necesario el uso de la tecnología existente y aquí es donde surge la motivación de este Trabajo de Fin de Grado.

1.2. Objetivo

El objetivo de este trabajo es el desarrollo de un entorno que permita modelar y simular el comportamiento de los ciudadanos de una zona urbana en lo que a la expansión de la COVID se refiere. Este puede ser configurado con distintos parámetros referentes a las características de la población, es decir, número de habitantes, hogares, capacidad de los distintos lugares, etc. A este se pueden aplicar también distintas restricciones que afectarán a los habitantes y su rutina diaria. Todos estos se describen en profundidad más adelante. Posteriormente se puede realizar una simulación de esta población y así se observa cómo evolucionaría la enfermedad y el impacto que tendrían las medidas restrictivas aplicadas, de forma que ayude a la toma de decisiones.

Por otra parte se obtiene una aplicación que integra esta funcionalidad, pero también permite visualizar todas las simulaciones que hemos realizado, y que permite borrarlas, exportar-

las o compararlas facilitando el manejo de estas.

1.3. Metodología

La metodología que se ha seguido para llevar a cabo este trabajo ha sido Scrum. Es un marco de trabajo para desarrollo ágil de software. [1] Se basa en un desarrollo incremental e iterativo de las funcionalidades. Scrum permite una gran flexibilidad para adaptarse a los cambios que puedan ocurrir durante el proceso, produce un ambiente de trabajo más colaborativo y una mayor productividad. Además al ser iterativo el resultado se va refinando obteniendo como resultado así un producto de mayor calidad.
El proceso se divide en distintos Sprints o iteraciones donde se ordenan las tareas dependiendo de las distintas prioridades de las funcionalidades que se requieren. En cada uno de los Sprints se establecen las tareas a realizar en él definiendo bien los requisitos a implementar y sus prioridades. Durante el Sprint se realizará el desarrollo de estas y también las pruebas necesarias para verificar que el comportamiento es correcto. Cuando este acaba se realiza una entrega del resultado para que pueda ser validado y se plantea el siguiente definiendo las tareas a realizar. Así de forma iterativa e incrementando el crecimiento de la aplicación hasta que se obtiene el resultado final.

1.4. Estructura del documento

En primer lugar se realiza un estudio de las tecnologías que se van a utilizar y se realiza una pequeña descripción de cada una. A continuación se explican todas las fases de desarrollo propias de la ingeniería del software con las respectivas conclusiones y análisis del resultado tras el desarrollo de la aplicación.

2

Tecnologías y herramientas

Las tecnologías que se han utilizado son las que se describen a continuación.

2.1. Python

Es un lenguaje de programación creado a principios de los ochenta [2]. Es un lenguaje de código abierto y orientado a objetos que proporciona un código muy legible y que se ha convertido en uno de los más utilizados en los últimos años. Esto se debe al gran número de campos en la industria que cubre, como la Inteligencia Artificial, Big Data, Data Science, Desarrollo Web, etc. [3]

2.2. MESA

MESA es un framework para el modelado basado en agentes de python. Este framework permite modelar el entorno, analizar los resultados de estos modelos, simularlos con distintos parámetros y visualizar la simulación mediante una interfaz de JavaScript [4].

2.3. Flask

Es un framework de Python para el desarrollo de aplicaciones web básicas de forma rápida, siguiendo el patrón MVC (Modelo, Vista, Controlador).

2.4. React

React es una librería de Javascript de código abierto para el desarrollo de interfaces de usuario de forma sencilla. Permite crear una estructura basada en componentes que React

actualiza y renderiza de forma eficiente dando como resultado una interfaz compleja. Además estos componentes manejan su propio estado y sus propiedades [5].

2.4.1. Librerías

Hay múltiples librerías de React que ofrecen muchos componentes listos para usar mediante una API. Las utilizadas para este proyecto son las siguientes:

1. RSuite [6] y Material-UI [7]. Estas librerías contienen elementos básicos para una aplicación como botones, barras de navegación, etc..

2. Rechart. Tiene componentes para la creación y personalización de distintos tipos de gráficas [8].

2.5. MongoDB Atlas

MongoDB es un sistema de base de datos no relacional. MongoDB Atlas es el servidor de base de datos de Mongo en la nube, lo que nos permite administrar los usuarios que pueden conectarse a nuestras bases de datos, así como también nos facilita la conexión con la aplicación mediante una URL.

2.6. Git/ Git Hub

Git es un software para el control de versiones del código, obteniendo así un historial de cambios de los distintos archivos del proyecto. Git Hub es una plataforma en la nube para alojar los distintos proyectos en repositorios, donde se usa git para el control de versiones.

2.7. Herramienta de diseño

2.7.1. Draw.io

Esta es una herramienta para el diseño de diagramas de flujo. Permite importar y exportar fácilmente los diagramas creados a partir de Google Drive.

2.7.2. MagicDraw UML

Es una herramienta CASE que nos ofrece los elementos necesarios para realizar el diseño del software a desarrollar mediante una gran variedad de diagramas, por ejemplo, diagramas de clase, de flujo, casos de uso, etc.

2.7.3. Balsamiq Cloud

Es una herramienta online para hacer maquetas de una aplicación. Pueden realizarse bocetos tanto de aplicaciones web y móviles como de escritorio. Se trata de una herramienta versátil y con una gran variedad de componentes gráficos. También existe una versión de escritorio llamado Balsamiq Studio.

3

Modelado basado en agentes

3.1. Modelado basado en agentes

El modelado basado en agentes es una técnica que lleva utilizándose ya varios años para sistemas complejos. Estos sistemas tienen una estructura interna y están compuestos de interacciones entre individuos. Estos individuos o en este caso llamados 'agentes' tienen un comportamiento definido por unas reglas que pueden cambiar con el tiempo o no. Además el comportamiento de estos agentes se encuentra influenciado por las interacciones de estos con otros agentes y con su entorno.

La estructura de estos modelos se divide en tres elementos básicos. Encontramos el conjunto de agentes que se encuentran diferenciados por sus atributos y su comportamiento descrito en acciones. Las relaciones y métodos de interacción que se identifica como la forma en la que los agentes interactúan entre ellos que está relacionado con el último elemento que es el entorno, dependiendo de la topología de este los agentes se relacionarán de una forma o de otra con este y con lo demás agentes.

La simulación del modelo se basa en repetir de forma iterativa el comportamiento e interacciones de los agentes de forma que este comportamiento y la estructura del modelo va evolucionando.

Una de las principales características de los agentes es que son autónomos. Su comportamiento les permite tomar decisiones con la finalidad de conseguir sus objetivos. Hay muchas formas de establecer el comportamiento de los agentes, desde simples reglas realizadas mediante instrucciones condicionales hasta complejos comportamientos adaptativos modelados con técnicas de inteligencia artificial, de forma que su comportamiento se base en reglas básicas y además de reglas que puedan modificarse con el avance de la simulación.

Pero para que los agentes puedan tomar sus propias decisiones deben basarse en algunos parámetros o atributos. Los valores del conjunto de atributos en una situación determinada define su estado en este situación. En una simulación basada en agentes el estado es toda la información que se necesita para mover el sistema de un punto al siguiente. Los valores del estado de un agente y el comportamiento se encuentran influenciados en todo momento por el estado de los demás agentes y del entorno con el que interactúa. De esta forma obtenemos una división entre atributos estáticos que no cambian durante la simulación, y atributos dinámicos que evolucionan junto con el progreso de esta.

Los sistemas basados en agentes con sistemas descentralizados, es decir, no hay una entidad central que comunique la información global a todos los agentes ni controle su comportamiento. Los agentes interactúan con otros, pero no todos los agentes interactúan con todos, lo normal es que solo lo hagan con subconjunto de estos. Normalmente este subconjunto esta formado por sus vecinos y es de la interacción entre estos donde se obtiene la información.

Dependiendo de la topología del modelo la vecindad se definirá de una forma u otra. Por ejemplo si hablamos de una cuadrícula, los vecinos pueden referirse a los agentes que se encuentren en las celdas vecinas, si hablamos de una red, la vecindad se encuentra definida de una forma más general, estas relaciones puedes ser estáticas o dinámicas y se pueden predefinir. Sin embargo en los modelos aespaciales no importa la localización [9].

Se han realizado distintos modelos basados en agentes aplicados al COVID teniendo en cuenta distintos parámetros, uno de los modelos que se realizaron fueron en la situación de Argentina [10]. También se encuentran estudios como el de la Universidad de Guadalajara en México [11] o el que se realizó en Brasil [12].

4

Análisis y especificación de requisitos

4.1. Requisitos funcionales

A continuación se enumeran los requisitos funcionales.

RF01. Un usuario puede ejecutar la simulación.

RF02. Un usuario puede parar la simulación una vez haya sido iniciada.

RF03. Un usuario puede restaurar la simulación para aplicar los parámetros introducidos.

RF04. Un usuario puede ejecutar la simulación manualmente paso a paso.

RF05. Un usuario puede ajustar la velocidad de la simulación.

RF06. Un usuario puede establecer distintos parámetros sobre la población.

RF07. Un usuario puede aplicar distintas restricciones a la población.

RF08. Un usuario puede guardar una simulación.

RF09. Un usuario puede listar todas las simulaciones.

RF10. Un usuario puede visualizar una simulación mediante una gráfica.

RF11. Un usuario puede exportar a Excel una simulación.

RF12. Un usuario puede eliminar una simulación.

RF13. Un usuario puede comparar distintas simulaciones.

4.2. Requisitos no funcionales

A continuación se enumeran los requisitos no funcionales del sistema.

RNF01. La aplicación se encuentra codificada en Python 3.6.

RNF02. El sistema cuenta con un almacenamiento en una base de datos MongoDB en MongoDB Atlas.

RNF03. El idioma del sistema es el inglés.

4.3. Análisis de variables

Para determinar los requisitos de la aplicación que se ha realizado se deben analizar las variables que afectan a la incidencia del Covid-19. En primer lugar tenemos los parámetros que el usuario puede definir que son los parámetros de la población y posteriormente las restricciones que se aplicarán para reducir la incidencia de la enfermedad en la población, y por otro lado tenemos el análisis de los parámetros referentes a la enfermedad que son parámetros internos.

4.3.1. Parámetros de entrada

Son parámetros que define el usuario antes de comenzar la simulación.

Parámetros de la población Con la finalidad de que la herramienta resultante fuera muy versátil y el usuario pudiera representar una situación y un lugar geográfico específicos fácilmente se han definido los distintos parámetros.

- Número de personas.
- Número de hogares.
- Número de lugares de trabajo.
- Capacidad media de los lugares de trabajo. Este parámetro es utilizado para el cálculo del aforo permitido en estos lugares pero también para los centros educativos.

- Número de centros educativos.
- Número de lugares de ocio. Se refiere a lugares recreativos, parques, tiendas, museos, cines, etc.
- Capacidad media de los lugares de ocio. Este parámetro es utilizado para el cálculo del aforo permitido en estos lugares.
- Número de lugares de hostelería.
- Capacidad media de los lugares de hostelería. Este parámetro es utilizado para el cálculo del aforo permitido en estos lugares.
- Número de hospitales.
- Capacidad de camas de hospital. Es el número de camas ”normales”de los hospital.
- Capacidad de camas de UCI. Es el número de camas de UCI de los hospitales.
- Porcentaje de personas con patologías. Es el porcentaje de personas con patologías que pueden encontrarse más afectadas por la enfermedad.
- Porcentaje de gente sana. Es el porcentaje de la población que se encuentra sana al inicio de la simulación.

Se ha realizado la división entre lugares de ocio y lugares de hostelería debido a que algunas restricciones afectan en estos lugares de forma distinta, como es el caso del uso de la mascarilla.

Restricciones Son las restricciones que el usuario puede definir y que restringen las acciones de la población.

- Porcentaje de la capacidad permitida en lugares públicos. Es el porcentaje del aforo permitido en lugares públicos, es decir, en lugares de trabajo, centros escolares, lugares de ocio, hostelería, etc.
- Uso obligatorio de mascarilla.
- Número máximo de personas en reuniones.

- Hora de cierre de los comercios no esenciales.
- Hora del toque de queda.
- Confinamiento total.

4.3.2. Parámetros internos

Son parámetros que se han definido interiormente en el modelo y que corresponden a parámetros reales que han sido definidos por los expertos sanitarios.

Parámetros Covid-19 Tenemos distintas variables relativas al tiempo en el ciclo de infección de la enfermedad.

- Distancia mínima de contacto. Consideramos la distancia mínima que debe mantenerse entre dos personas para que haya cero riesgo de contraer la enfermedad. Según la OMS se debe guardar al menos un metro [13], pero esto no reduce el riesgo a cero, el Ministerio de Sanidad aconseja 1,5 metros [14]. Por lo que finalmente se ha definido una distancia de 2 metros.
- Tiempo de incubación. Es el tiempo que transcurre con la enfermedad hasta que esta presenta síntomas. Según el ministerio de sanidad se establece en 5 días [15]. Se establece este tiempo desde que una persona se infecta hasta que presenta síntomas y por tanto se contabiliza como positivo, pero además si este es asintomático y no presenta síntomas, tomamos los 5 días como el tiempo que tarda en ser avisado por su contacto directo. Según el Ministerio de Sanidad se establece en 5 días [15].
- Tiempo con la enfermedad. Es el tiempo desde que se contrae hasta que la vencemos. Según el Ministerio de Sanidad son 2 semanas de media si la persona presenta síntomas leves y de 3-6 semanas cuando ha sido grave o crítica [15].
- Tiempo de transmisión. Es el tiempo durante el cual somos capaces de contagiar y se detecta en nuestro organismo. La transmisión empieza de 1 a 2 días antes de dar síntomas [15].

5

Diseño de la aplicación

El diseño de la aplicación se encuentra dividido desde dos puntos de vista distintos. Por un lado podemos realizar una división desde el punto de vista de la interfaz de usuario y por otro lado, desde la propia arquitectura de la aplicación.

5.1. Diseño de la interfaz

Aquí realizamos una división en el diseño de la interfaz. Por un lado tenemos el diseño del servidor del framework MESA, que se encuentra limitado, y por otro, tenemos el diseño de la aplicación que integra la simulación.

5.1.1. Servidor de simulación MESA

Debido a que el framework MESA ya proporciona una interfaz propia del servidor web, el diseño de esta parte se encuentra restringido. Esta parte se ha basado en añadir los componentes para la entrada de datos y para la representación de los resultado. Además se han personalizado los elementos visuales que se encuentran en la simulación. Los bocetos se pueden ver en la Figura 1, Figura 2 y Figura 3 .

5.1.2. Diseño de la aplicación

El diseño de la aplicación se ha realizado con elementos de distintas librerías de React, mencionadas anteriormente.

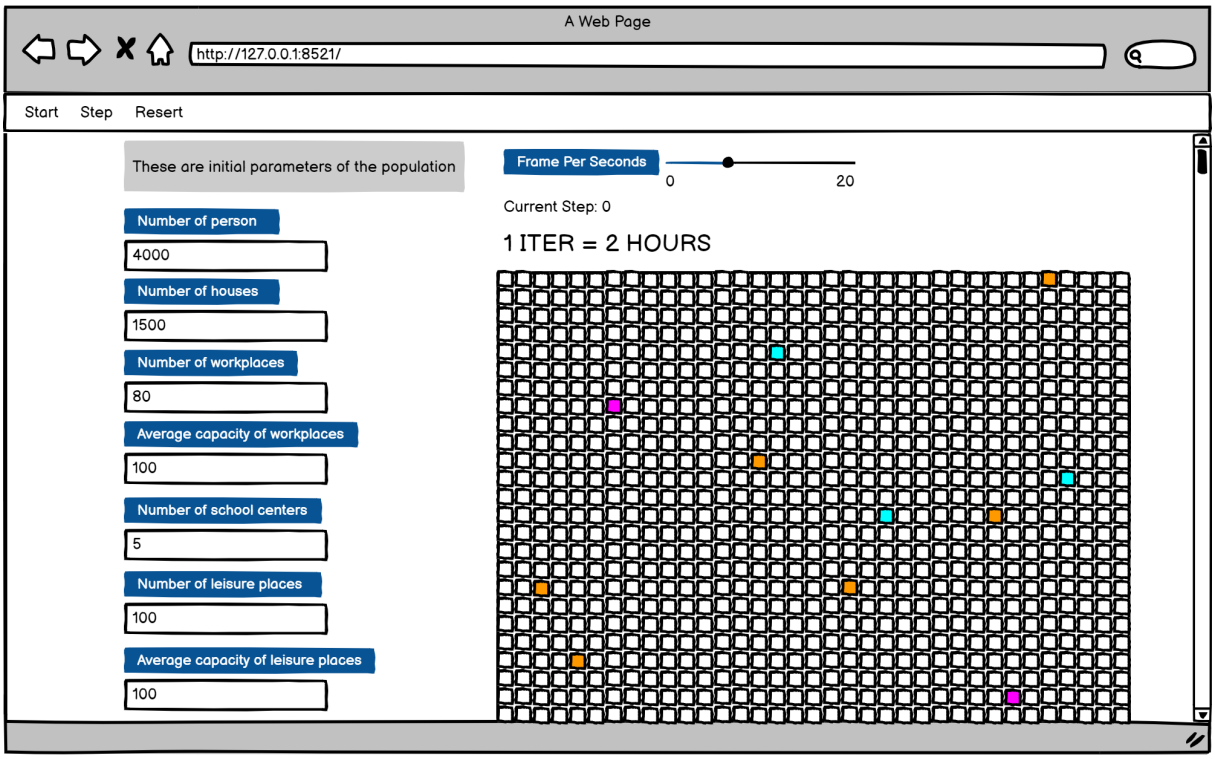

Figura 1: Servidor MESA

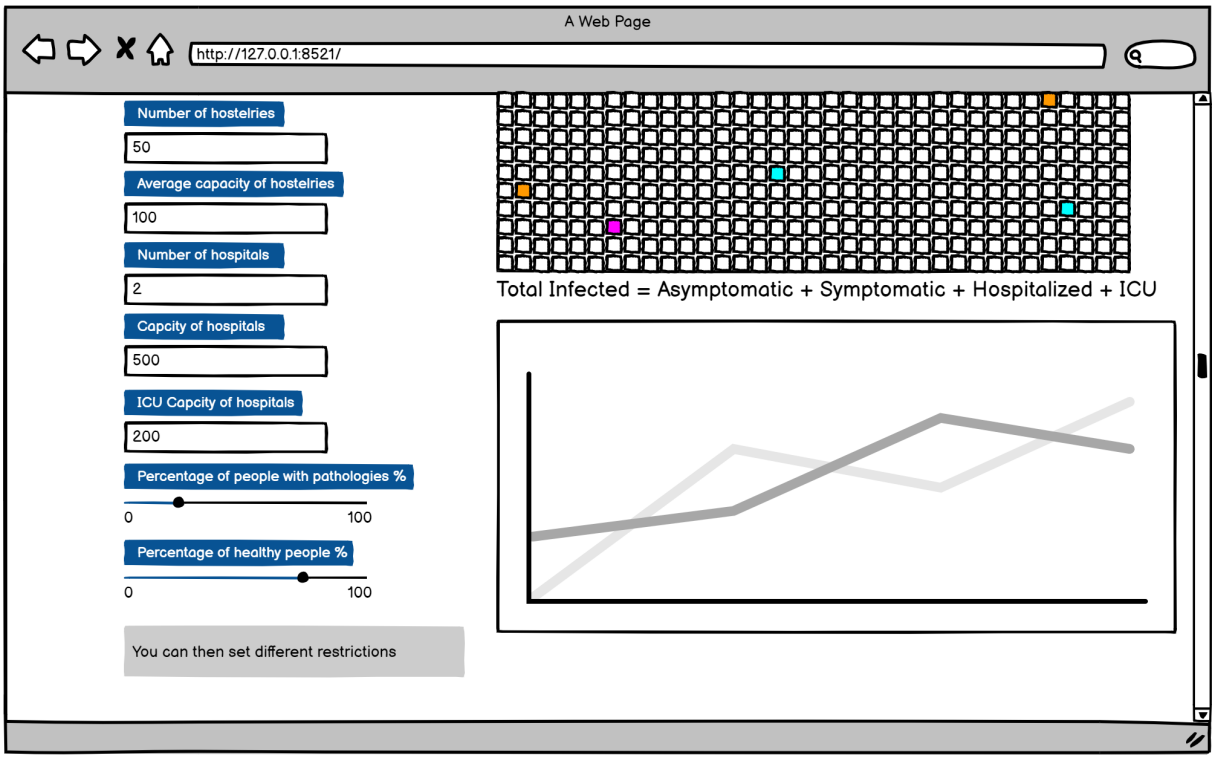

Figura 2: Servidor MESA

Figura 3: Servidor MESA

5.2. Formalización del modelo basado en agentes

Para realizar el diseño del modelo en el que se basa la simulación se han estudiado los distintos agentes.

5.2.1. Agentes

Persona Representa a las personas. Sus atributos son la posición, la edad, patologías, estado de infección, estado, fecha de infección, hospital, hogar, trabajo, amigos de ocio, amigos de hostelería y uso de mascarilla. Sus acciones son ir al trabajo, ir a casa, dormir, enfermar, morir, curarse y tener ocio.

Hogar Representa los hogares y sirven para clasificar las personas que son convivientes. Sus atributos son la posición y el conjunto de convivientes.

Lugares Representa los lugares en general donde asisten las personas. Sus atributos son la posición, el conjunto de personas que trabajan en ese lugar, la capacidad, el porcentaje de la

capacidad permitida y la capacidad permitida.

Negocios Representa los lugares de trabajo donde asisten las personas. Sus atributos son los de los lugares más el conjunto de agentes que van a trabajar de forma presencial. La única acción que tiene es seleccionar el conjunto de agentes que asistirán de forma presencial.

Centros escolares Representa los centros escolares donde asisten las personas menores de 18 años. Sus atributos y acciones son los mismos que los de los lugares de trabajo.

Lugares de ocio Representa los lugares de ocio como son los parques, lugares recreativos, cines, museos, etc.. Sus atributos son los mismos que los lugares más la hora de cierre.

Restauración y hostelería Representa los bares, restaurantes, discotecas, etc. Sus atributos son los mismos que los lugares de ocio.

Hospital Representa los hospitales. Sus atributos son los pacientes hospitalizados, los pacientes en UCI, la capacidad de camas normales de hospital y la capacidad de camas de UCI.

Se ha realizado la división entre lugares de ocio y lugares de hostelería debido a que algunas restricciones afectan en estos lugares de forma distinta, como es el caso del uso de la mascarilla.

5.2.2. Estados

Para definir el estado de una persona en una determinada iteración se definen dos tipos de estado. Por un lado tenemos el **estado físico** que se clasifica en dormido, en casa, en el trabajo, tiempo libre, en un lugar de ocio, en un lugar de restauración, en el hospital o fallecido. Y por otro lado tenemos el **estado de infección** que se clasifica en sano, asintomático, sintomático, sospechoso negativo, hospitalizado, en UCI, inmune y fallecido. Decimos que una persona es sospechosa negativa cuando ha estado en contacto directo con un positivo pero sin embargo, no ha contraído la enfermedad; en estos casos esta persona también debe guardar cuarentena.

5.2.3. Representación de los elementos en la simulación

Para representar los distintos agentes y los estados de una persona se han utilizado diferentes colores para poder distinguirlos, representados en la Figura 4.

Figura 4: Código de colores

5.2.4. Diagrama de clases

Para una mejor representación de las relaciones entre los agentes se ha diseñado un diagrama de clases. Véase la Figura 5.

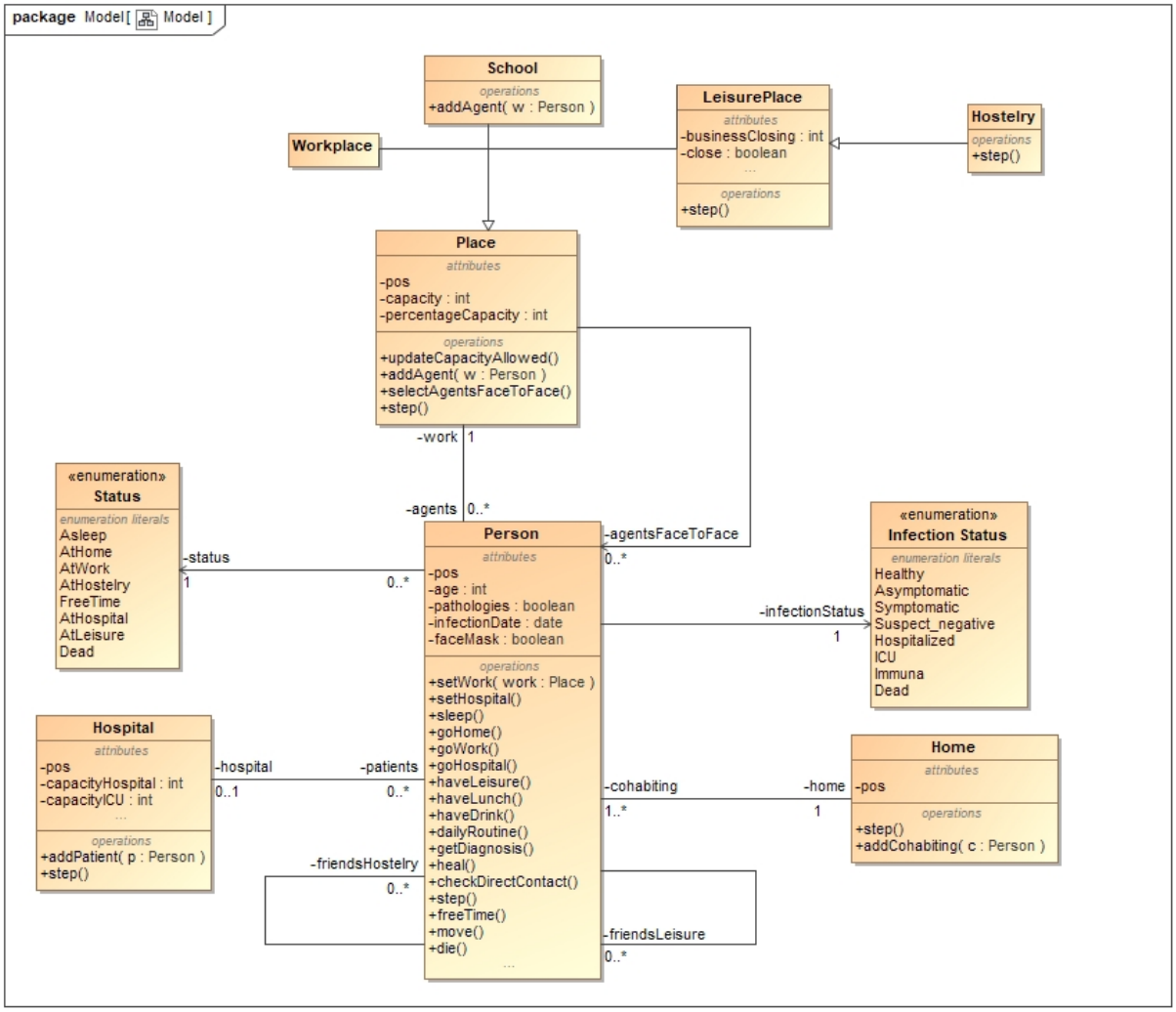

Figura 5: Diagrama de clases

5.2.5. Comportamiento de los agentes

En el modelado basado en agentes se define el comportamiento que deben seguir estos. Antes de definir este comportamiento es necesario definir cómo transcurre el tiempo mediante iteraciones, es decir qué cantidad de tiempo transcurre de una iteración a otra. En este caso se ha decidido que este tiempo es de 2 horas, debido a que la acción de una persona, que es el

agente principal, durará mínimo dos horas, ya puede ser dormir, estar en el trabajo o pasar el tiempo libre en un restaurante, o algún lugar de ocio.

Como hemos mencionado, el agente principal y el más importante en este modelo es el agente Persona. Para este se han definido distintos comportamientos dependiendo de su estado de salud pero también del punto en el que se encuentra la simulación, es decir, la hora del día y el día de la semana en la que se encuentra.

Se define una rutina diaria para este tipo de agentes debido a que el objetivo principal es representar una población real, por lo tanto se ha intentando aproximar lo máximo posible los comportamientos de estos agentes a uno real. Para esto hemos dividido las personas en tres grupos de edad: los menores de 18 años, los que se encuentran entre los 18 y los 65 años y los mayores de 65 años. La rutina de cada uno de este grupo queda definido por el Cuadro 1 y el Cuadro 2.

Otro comportamiento definido ha sido el de los lugares de trabajo y los centros educativos: cada día, de lunes a viernes a las 8:00 h, estos deciden el subgrupo de trabajadores o estudiantes que deben asistir al lugar de forma presencial.

5.2.6. Diagrama de flujo

Para representar el flujo de las distintas acciones de una personas en cada iteración se ha diseñado un diagrama de flujo, que se encuentra en la Figura 6.

5.3. Descripción de la arquitectura de la aplicación

La aplicación se divide en servidor o backend y cliente o frontend.

5.3.1. Servidor

En la parte del servidor encontramos las operaciones necesarias para interaccionar con la base de datos y proporcionar la información necesaria al cliente mediante respuestas HTTP a sus peticiones. Esta parte se ha realizado con el framework de Python, Flask.

La base de datos que se ha utilizado es MongoDB que se encuentra almacenada en MongoDB Atlas. Esta base de datos solo tiene una colección llamada simulación con la estructura de la Figura 7.

Nº iter % 12	Hora real	Acciones		
		Trabajadores (mayores de 18 y menores de 65)	**Estudiantes (entre 3 y 18 años)**	**Jubilados (mayor de 65)**
0	00:00	Dormir	Dormir	Dormir
1	02:00	Dormir	Dormir	Dormir
2	04:00	Dormir	Dormir	Dormir
3	06:00	Dormir	Dormir	Dormir
4	08:00	Levantarse e ir al trabajo	Levantarse e ir a la escuela	Tiempo libre
5	10:00	Trabajar	Estudiar	Tiempo libre
6	12:00	Trabajar	Estudiar	Tiempo libre
7	14:00	Almuerzo	Volver a casa y almuerzo	Almuerzo
8	16:00	Trabajar	Tiempo libre	Tiempo libre
9	18:00	Tiempo libre	Tiempo libre	Tiempo libre
10	20:00	Cena	Cena	Cena
11	22:00	Tiempo libre	Tiempo libre	Tiempo libre

Cuadro 1: Rutina de Lunes a Viernes.

Debido a que esta parte se ha creado en Flask, se definen distintas rutas"para realizar peticiones a este, que nos permiten obtener las simulaciones almacenadas, guardar nuevas simulaciones, eliminarlas y exportarlas a excel.

5.3.2. Cliente

Esta es la parte estética de la aplicación y por tanto la que el usuario va a visualizar. Se encarga de realizar las peticiones HTTP al servidor para obtener la información necesaria, procesarla y mostrarla al usuario. Esta parte se ha realizado con la tecnología React.

Corresponde por tanto a las funcionalidades que ofrece la aplicación, es decir, listar las simulaciones, visualizarlas y compararlas. En la Figura 8, Figura 9, Figura 10, Figura 11, Figura 12, Figura 13 y Figura 14, se muestran los diseños de las interfaces de usuario.

Nº iter % 12	**Hora real**	**Acciones**		
		Trabajadores (mayores de 18 y menores de 65)	**Estudiantes (entre 3 y 18 años)**	**Jubilados (mayor de 65)**
0	00:00	Tiempo libre	Tiempo libre	Dormir
1	02:00	Dormir	Dormir	Dormir
2	04:00	Dormir	Dormir	Dormir
3	06:00	Dormir	Dormir	Dormir
4	08:00	Tiempo libre	Tiempo libre	Tiempo libre
5	10:00	Tiempo libre	Tiempo libre	Tiempo libre
6	12:00	Tiempo libre	Tiempo libre	Tiempo libre
7	14:00	Almuerzo	Almuerzo	Almuerzo
8	16:00	Tiempo libre	Tiempo libre	Tiempo libre
9	18:00	Tiempo libre	Tiempo libre	Tiempo libre
10	20:00	Cena	Cena	Cena
11	22:00	Tiempo libre	Tiempo libre	Tiempo libre

Cuadro 2: Rutina de Sábado y Domingo.

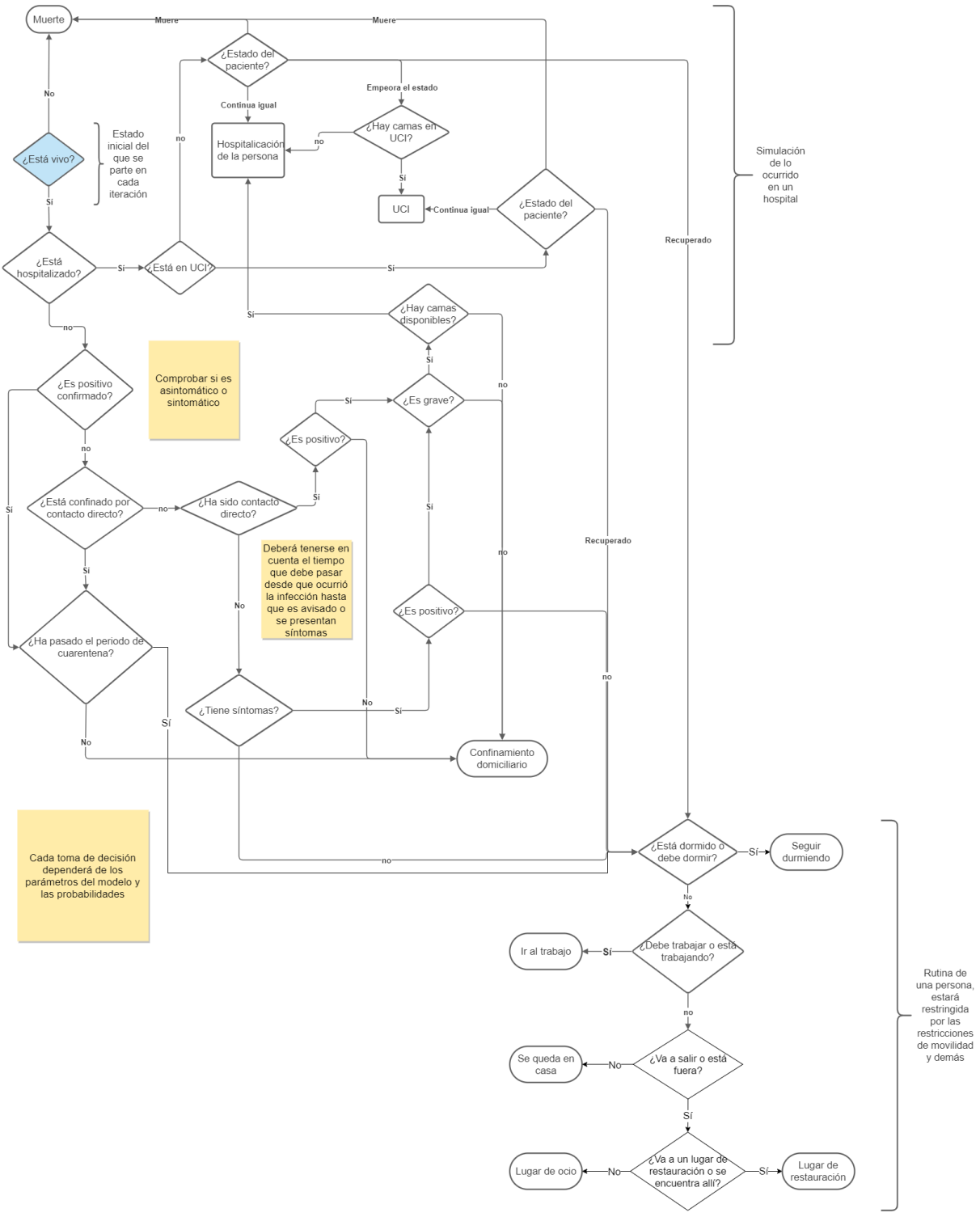

Figura 6: Diagrama de flujo

```
{
    "_id": {
        "$oid": "60c112ed3bd335a9ed9506a5"

},
    "creation_date": "Sat May 29 17:25:00 2021",

"parameters": {
        "num_persons": 4000,

"num_houses": 1500,
        "num_work": 80,

"capacity_work": 100,
        "num_school": 5,

"num_leisure_places": 50,
        "capacity_places": 100,

"num_hostelry": 50,
        "capacity_hostelry": 100,

"num_hospital": 2,
        "capacity_hospital": 500,

"capacity_ICU": 200,
        "per_path": 20,

"per_healthy": 85
    },
    "restrictions": {

"capacity_work_school": 100,
        "face_mask": false,

"num_friends": 5,
        "business_closing": 4,

"curfew_time": 4,
        "total_isolation": false
    },

"iters": "10",
    "iterations": {
        "Healthy":
[{ 0: 2309 }, ...],
        "Immune": [{ 0: 1092 }, ...],
        "Suspect
negative": [{ 0: 1125 }, ...],
        "Asymptomatic": [{ 0: 175 }, ...],

   "Symptomatic": [{ 0: 139 }, ...],
        "Total_Infected": [{ 0: 599
}, ...],
        "Hospitalized": [{ 0: 147 }, ...],
        "ICU": [{ 0:
138 }, ...],
        "Dead": [{ 0: 0 }, ...]
    }
}
```


Figura 7: Esquema MongoDB

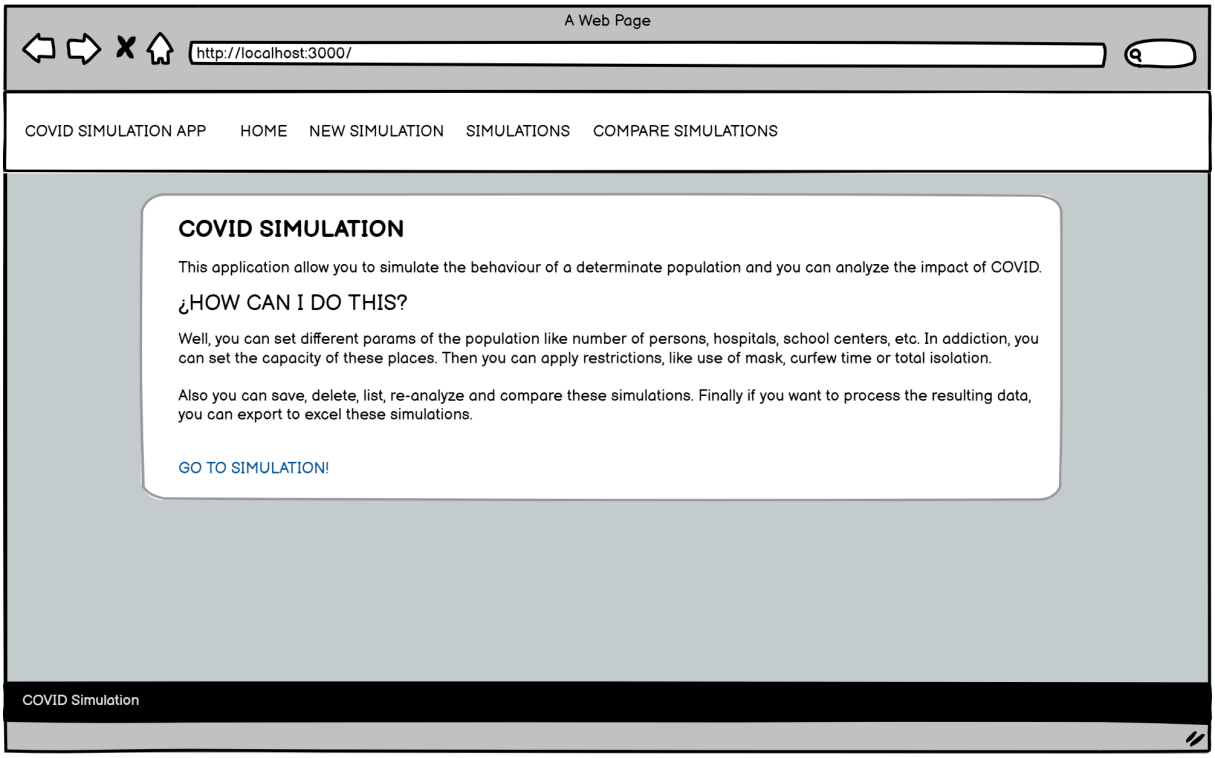

Figura 8: Página de inicio

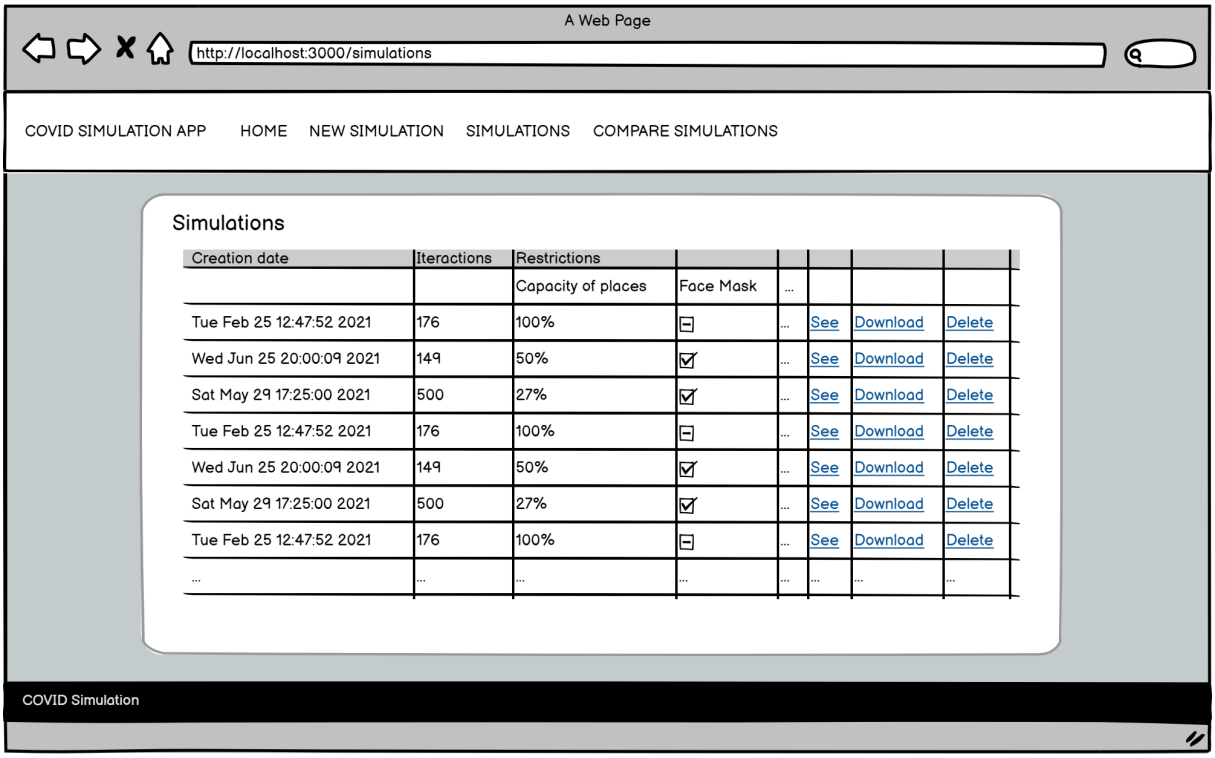

Figura 9: Simulaciones

Figura 10: Visualizar simulación

Figura 11: Visualizar simulación

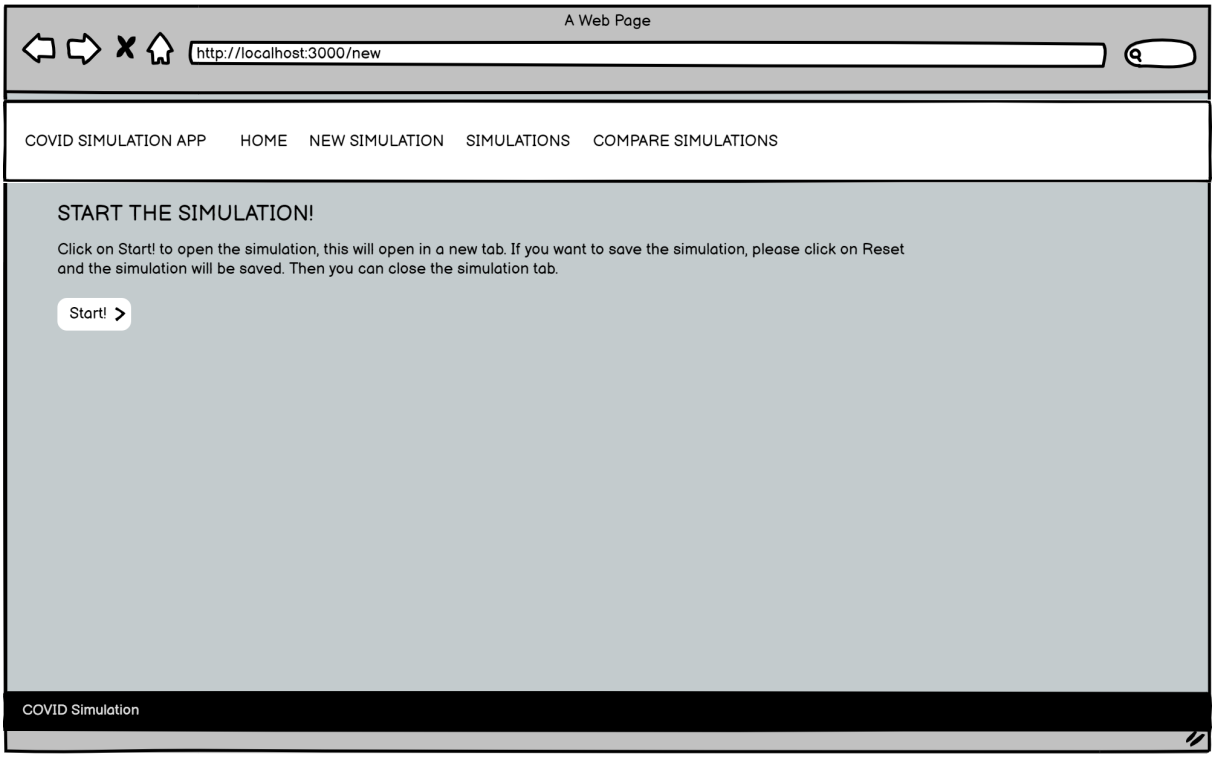

Figura 12: Nueva simulación

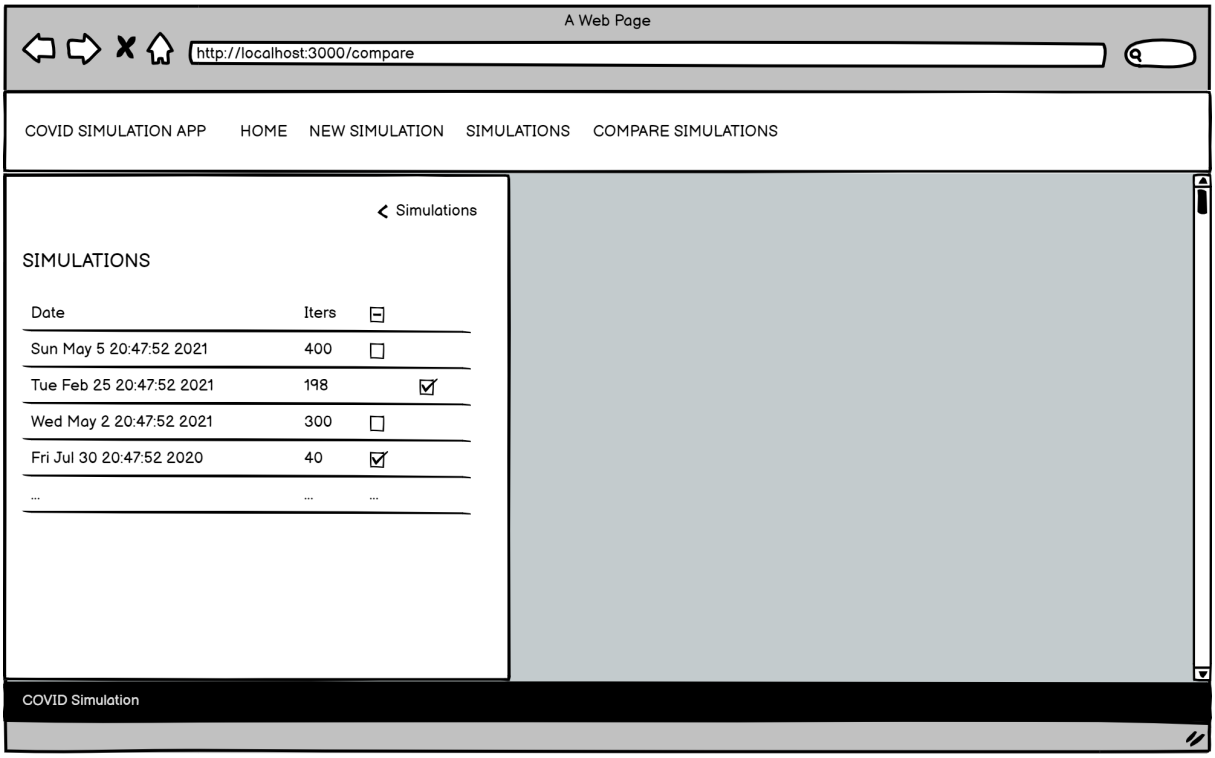

Figura 13: Comparar simulaciones

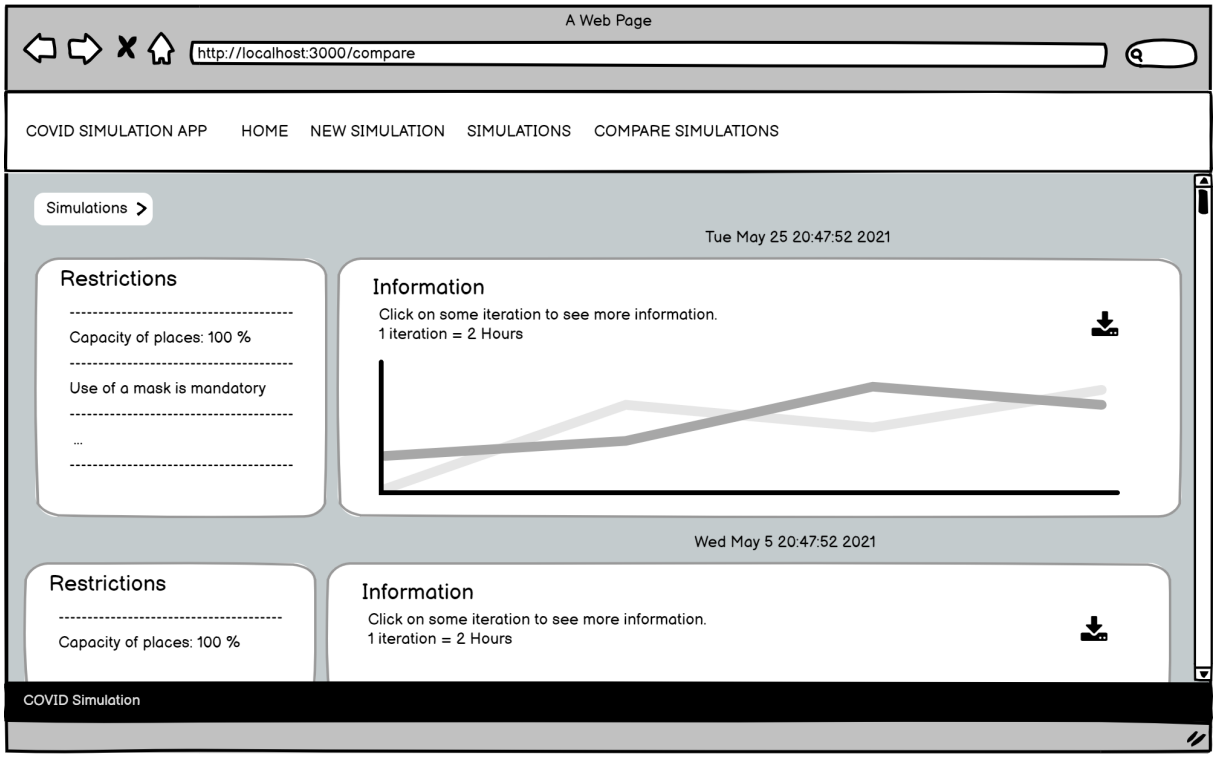

Figura 14: Comparar simulaciones

6

Implementación de la aplicación

6.1. Implementación del modelo basado en agentes

A continuación se describen las distintas fases en las que ha consistido el desarrollo de la herramienta.

6.1.1. Entorno de simulación - MESA

En primer lugar se realizó la implementación del entorno en el que se va a realizar la simulación. Como se ha mencionado anteriormente el framework utilizado para ello ha sido MESA. Este nos proporciona distintas clases básicas para implementar la simulación como son Model, Agent, RandomActivation, etc. Las clases implementadas heredarán de estas.

La clase **Model** es la clase central del modelo, una instancia de esta clase corresponde a una simulación determinada. Esta clase es la que gestiona los agentes y almacena la información básica del modelo como son los parámetros de entrada y los atributos propios del modelo.

La clase **Agent** representa a los elementos que contiene la simulación. Estos modelos tienen una serie de atributos que definen su estado y, en función de esto, y de las interacciones con los demás agentes, tomarán decisiones y actuarán, también conforme a su comportamiento descrito.

Para hacer avanzar el modelo y la simulación utilizamos un objeto planificador o 'scheduler'. Los modelos basados en agentes se mueven en pasos (steps) o tics. Este tipo de clase se encarga de activar a los agentes en cada paso. En este caso hemos utilizado la clase RandomActivation que activa de forma aleatoria a todos los agentes haciendo que 'den un paso más' en la simulación y estos actuarán conforme a su comportamiento definido e interactuando con

su entorno y los demás agentes.

Además MESA tiene una clase definida llamada **DataCollector** que se usa para recoger la información que necesitemos en cada iteración de la simulación y, de esta forma, poder almacenar los resultados intermedios del modelo.

Por último, implementamos la parte visual de la simulación, por lo que se debe iniciar un servidor. Para esto se ha utilizado la clase **ModularServer** que proporciona MESA. Para inicializar esta clase solo se necesita el modelo, los componentes visuales y los parámetros de entrada que queremos que el usuario introduzca; estos últimos son los descritos en los requisitos y para implementarlos se ha utilizado la clase **UserParam**.

Para la parte visual este framework proporciona muchos componentes, pero además se pueden crear módulos personalizados con Javascript e incluirlos. En primer lugar añadimos el componente principal y el más importante: el espacio sobre el cual se moverán los distintos agentes. MESA proporciona dos tipos de espacios generales, la cuadrícula (grid) que como su nombre indica se trata de una cuadrícula y el continuo que se trata de un espacio en blanco. En este caso utilizamos una cuadrícula, en concreto se usa **CanvasGrid** ya que es más ordenado para situar toda la variedad de agentes que tenemos. En la simulación implementada en una celda puede haber más de una persona, considerándose así para poder realizar la simulación con un mayor número de personas.

Además del espacio se han añadido tres gráficas que nos muestran para cada estado de salud que se ha definido, la cantidad de personas que se encuentran en él durante la simulación. Los dos primeras gráficas son instancias de la clase **ChartModule**, y se tratan de dos gráficas de líneas. En la primera se representan las personas sanas, inmunes, el total de infectados y las muertes. En la segunda tenemos el número total de personas infectadas, las hospitalizadas y las que se encuentran en UCI. La tercera gráfica se trata de un histograma donde se representa la cantidad de personas que hay en cada estado de salud en la iteración actual.

Cuando hablamos del total de infectados a la hora de representar los resultados se engloban a las personas asintomáticas, sintomáticas, hospitalizadas y en UCI.

6.1.2. El modelo

El modelo se encarga de instanciar los distintos agentes y de inicializar la simulación. Para ello se crearán los hogares, personas, colegios, lugares de trabajo y hospitales dependiendo del

número de estos que defina el usuario. Después se asignará un hogar a cada persona de forma aleatoria y un lugar de trabajo si es mayor de 18 años o escuela si es menor.

La capacidad de las escuelas se calculará una vez que a todos los menores se les haya asignado una, ya que tienen derecho a tener una plaza en algún colegio.

La capacidad de los lugares de trabajo será definida por los usuarios. Posteriormente se asignará un lugar de trabajo a cada persona hasta que se complete la capacidad de cada uno. Cuando no queden lugares de trabajo, las personas que no tengan ninguno se considerará como desempleado.

Si una persona inicialmente se encuentra hospitalizado, o en UCI, se le asignará un hospital. Se buscará un hospital donde haya espacio adecuado para el paciente. Si por ejemplo solo está hospitalizado y todas las camas están ocupadas pero hay camas de UCI libres, se le asignará una de estas. Si es paciente de UCI pero no hay camas de UCI, se le asignará una normal. Si no hay camas de ningún, tipo el paciente permanecerá en su casa.

6.1.3. Los agentes

El siguiente paso en la implementación de la herramienta es definir los atributos y comportamientos de los agentes. Estos ya han sido explicados anteriormente por lo que este apartado se centrará únicamente en describir en profundidad el comportamiento del agente Persona.

Los individuos como agentes. El diseño del comportamiento de este tipo de agentes se ha descrito anteriormente, pero se explica con mayor detalle a continuación.

Para cada iteración del modelo, si la persona no ha fallecido pero se encuentra hospitalizada, entendemos como hospitalizada una persona que necesita cuidados especiales, y que pasado el tiempo medio establecido con la enfermedad, se curará o morirá. Para tomar esta decisión se ha considerado que si una persona no ha podido ser hospitalizada morirá, debido a que no ha recibido la atención necesaria. En otro caso la mayoría de las personas vivirán. Si no ha pasado el tiempo medio, pero no se encuentra en un hospital se buscará una cama libre para esa persona.

Por otro lado, si una persona se ha infectado y ha pasado el tiempo de incubación, **se obtendrá un diagnóstico** y se pondrá en cuarentena. Si no ha pasado el tiempo de incubación, **seguirá con su rutina** diaria y, si se ha infectado y ha pasado el tiempo medio de cuarentena

establecido, se curará y pasará a ser inmune.

Por último, si una persona no es inmune, no está en cuarentena, ni se ha infectado, se comprobará si en ese momento es **contacto directo** de alguna persona infectada.

Obtener un diagnóstico. Para obtener el diagnóstico de una persona, es decir el grado de gravedad de su infección, se ha tenido en cuenta su edad y sus patologías, además del factor aleatorio.

- Si la persona es mayor de 60 años y tiene patologías, esta persona será hospitalizada o necesitará cuidados intensivos (UCI).
- Si una persona tiene patologías o no es mayor de 40 años, entonces será sintomático, deberá ser hospitalizado o necesitará cuidados intensivos.
- En otro caso podrá ser asintomático, sintomático, hospitalizado o sospechoso negativo, es decir no ha contraído la enfermedad.

Ser contacto directo. Consideramos que una persona es contacto directo de otra si está en la misma celda que otra infectada y no se encuentra en un lugar de ocio, si hay algún infectado en el grupo de personas con las que convive o si en el grupo de personas con el que ha asistido a algún lugar de ocio hay algún infectado. La gestión de lugares de ocio se explicará más adelante.

Rutina diaria. La rutina diaria ha sido descrita también anteriormente en el diseño. Aparte del desglose de las actividades de la Tabla 2, en las franjas de tiempo libre se ha considerado que la decisión de acudir a un sitio de ocio, a uno de restauración o quedarse en casa se ha realizado de forma aleatoria. Por otro lado, en las franjas que corresponden con la cena o el almuerzo, la decisión de asistir a un restaurante o quedarse en casa también se realiza de forma aleatoria.

Los lugares como agentes. Los lugares definidos también tienen un comportamiento aunque de forma más pasiva.

Con respecto a los **lugares de ocio** y la **hostelería**, en cada paso del modelo se dividirán las personas de cada lugar en subgrupos dependiendo del número de personas permitidas en reuniones de modo que, si alguien está infectado, solo afectará a los que están en ese grupo.

6.1.4. Medidas restrictivas

Por último tenemos la implementación de las medidas restrictivas. La introducción de estas ha modificado el comportamiento de los agentes anteriores.

Aforo. Se introducirá el aforo permitido en los espacios cerrados. En el caso de lugares de trabajo y escuelas se decidirá todos los días qué personas asistirán de forma presencial. En el caso de los lugares de ocio y hostelería cuando una persona quiera ir a uno de estos se buscará uno libre, si no se encuentra ninguno esa persona irá a casa si es hora de comer (almuerzo, cena) o caminará libremente.

Uso obligatorio de mascarilla. Si se activa esta opción, todas las personas llevarán mascarilla en todo momento excepto en sus hogares y en los lugares de restauración. Solo se podrán contagiar si no la llevan.

Cierre de comercios no esenciales. Inicialmente aparece desactivado con un valor fijado en las 4 horas de la madrugada. Cuando este valor se modifique, por ejemplo a las 20 (las 8 de la tarde), a partir de esa hora las personas no podrán asistir a ningún lugar de ocio, únicamente podrán ir a su hogar u moverse libremente.

Toque de queda. Inicialmente aparece desactivado con un valor fijado en las 4 horas de la madrugada. Cuando este valor se modifique, por ejemplo a las 23 (las 11 de la noche), a partir de esa hora las personas no podrán salir de su hogar.

Número de personas en reuniones. Con respecto a los lugares de ocio y la hostelería, en cada paso del modelo se dividirán las personas de cada lugar en subgrupos dependiendo del número de personas permitidas en reuniones de modo que si alguien está infectado solo afectará a los que están en ese grupo.

Confinamiento total. Cuando esta opción esté activa, las personas únicamente saldrán para ir al trabajo o a la escuela y después volverán a casa cuando acaben su jornada.

Debido a la introducción de las restricciones se ha creado una clase para la planificación de la ejecución de los pasos en cada iteración. De esta forma, los lugares de trabajo y los centros escolares ejecutan sus acciones antes para poder avisar a los agentes que deben asistir de forma presencial, después irán las personas y por último los lugares de ocio y la hostelería que organizarán a las personas que hayan asistido en grupos de personas según las restricciones.

6.2. Implementación de la aplicación

6.2.1. Servidor

El código de la simulación, el cual ha sido descrito anteriormente, se encuentra integrado en el servidor de la aplicación.

Como se ha mencionado anteriormente en el diseño, la estructura de este se divide en rutas, como se puede ver en el Cuadro 6.2.1.

Ruta	**Método**	**Implementación**
/simulations	GET	Obtiene todas las simulaciones
/simulations/<id>	GET	Obtiene la simulación con el id indicado
/simulations/<id>	DELETE	Elimina la simulación con el id indicado
/export/<id>	POST	Se crea un archivo excel con la simulación con el id indicado
/simulation	GET	Inicia el servidor de la simulación

Para exportar una simulación concreta se ha utilizado la librería de Pandas para crear el fichero excel y de esta forma enviarlo al cliente para su descarga.

Base de datos. Para todas las rutas anteriores se hace uso de la base de datos definida en MongoDB, ya sea para obtener todas la simulaciones, obtener una en concreto, eliminarla o guardarla. Dicha base de datos llamada *covid_simulation* se ha configurado mediante la URL que proporciona MongoDB Atlas para conectar bases de datos con un lenguaje y versión específicos, en este caso Python 3.6. Usando esta dirección con el conector *PyMongo* que incluye

flask se obtiene un objeto de este tipo, que nos permite obtener la base de datos y realizar las acciones necesarias para guardar simulaciones, borrarlas u obtenerlas.

6.2.2. Cliente

La parte del cliente ha sido implementada en React por lo que la implementación se ha dividido en distintos componentes (clases) de React.

En primer lugar tenemos el componente *Navbar* para el menú horizontal de la aplicación. Posteriormente se divide la implementación en las distintas funcionalidades o vistas que se pueden encontrar.

La primera de estas es la clase *Home*, que corresponde con la vista inicial de la aplicación donde se resume de qué funcionalidades está provista esta.

Después podemos encontrar las demás funcionalidades accesibles a través del menú superior. La siguiente funcionalidad es la que aparece como *New Simulation* que corresponde con la clase *Simulation* y que mediante una petición al servidor se inicia el servidor de MESA donde se puede ejecutar la simulación.

La parte de *Simulations* corresponde con la funcionalidad de visualizar todas las simulaciones que se encuentran en la base de datos, mediante una petición GET a nuestro servidor. Posteriormente en esta tabla podemos realizar distintas acciones. Se puede exportar a excel una simulación y se puede eliminar una simulación mediante las peticiones correspondientes al servidor, pero también se puede visualizar una simulación. Esta última funcionalidad corresponde con la clase *ShowSimulation* y que permite visualizar los resultados que se han obtenido durante la simulación en una gráfica lineal, y pulsando en alguna iteración de esta nos aparece los valores para esa iteración, además aparecen las restricciones de dicha simulación y los parámetros.

Por último tenemos la clase *CompareCharts* que corresponde con la funcionalidad de comparar simulaciones. Esta consiste en seleccionar de una tabla las simulaciones que deseemos comparar y las gráficas de estas aparecerán una detrás de otra junto con sus restricciones de forma que se puedan comparar fácilmente.

Todo esto al igual que el servidor también se divide en rutas. Véase el Cuadro 6.2.2

El uso de estas funcionalidades aparece de forma más detallada en el manual de usuario.

Ruta	Vista	Componente	Funcionalidad
/	Home	Home	Página de inicio.
/new	New Simulation	Simulation	Página de inicio de la simulación.
/simulations	Simulations	Simulations	Listado de todas las simulaciones, permite ir a la página de visualización de una simulación, exportarla o eliminarla.
/simulation/<id>	Simulation	ShowSimulation	Visualización de una simulación.
/compare	Compare Simulations	CompareSimulations	Visualización de varias simulaciones.

7

Verificación y pruebas

7.1. Verificación y pruebas

Se han realizado distintas pruebas para verificar el correcto funcionamiento de la aplicación. Para describir dichas pruebas se ha utilizado el lenguaje Gherkin [16].

7.1.1. Un usuario puede ejecutar la simulación.

Feature: Ejecución de la simulación

Given Me encuentro en la página de inicio de la aplicación

When Pulso en la opción *New Simulation* del menú

And Pulso en el botón *Start*

And Pulso el botón *Start* de la página de simulación

Then Empieza la simulación aumentando las iteraciones y moviéndose los agentes.

7.1.2. Un usuario puede parar la simulación una vez haya sido iniciada.

Feature: Parar una simulación

Given Me encuentro en la página de la simulación

And La simulación se encuentra iniciada

When Pulso en el botón *Stop*

Then La simulación se detiene

7.1.3. Un usuario puede restaurar la simulación para aplicar los parámetros introducidos.

Feature: Restaurar la simulación

Given Me encuentro en la página de la simulación

When Pulso en el botón *Reset*

Then Se han cambiado las posiciones y los elementos de la cuadrícula

7.1.4. Un usuario puede ejecutar la simulación manualmente paso a paso.

Feature: Aumentar manualmente la simulación paso a paso

Given Me encuentro en la página de la simulación

When Pulso en el botón *Step*

Then Aumenta el número de iteración de la simulación en uno

7.1.5. Un usuario puede ajustar la velocidad de la simulación.

Feature: Aumentar manualmente la simulación paso a paso

Given Me encuentro en la página de la simulación

And La simulación se encuentra iniciada

When Muevo la barra con la etiqueta *Frames Per Second* hacia la derecha

Then Aumenta la velocidad con la que aumentan las iteraciones

7.1.6. Un usuario puede establecer distintos parámetros sobre la población.

Feature: Introducir el número de colegios para la simulación

Given Me encuentro en la página de la simulación

When Cambio el número de colegios a 6

And Pulso el botón de *Reset*

Then Aparece en la cuadrícula 6 colegios

7.1.7. Un usuario puede aplicar distintas restricciones a la población.

Feature: Aplicar distintas restricciones

Given Me encuentro en la página de la simulación

When Aplicar las restricciones que desees

And Pulso el botón de *Reset*

Then Las restricciones se han aplicado

7.1.8. Un usuario puede guardar una simulación.

Feature: Guardar una simulación en ejecución

Given Me encuentro en la página de la simulación

And La simulación se encuentra iniciada

When Pulso el botón de *Reset*

Then La simulación se ha guardado

Feature: Guardar una simulación parada.

Given Me encuentro en la página de la simulación

And La simulación se encuentra parada

And La simulación tiene más de una iteración

When Pulso el botón de *Reset*

Then La simulación se ha guardado

7.1.9. Un usuario puede listar todas las simulaciones.

Feature: Listar todas las simulaciones.

Given Me encuentro en la página de inicio de la aplicación

When Pulso en la opción *Simulations* del menú

Then Aparece una tabla con todas las simulaciones

7.1.10. Un usuario puede visualizar una simulación mediante una gráfica.

Feature: Visualizar una simulación mediante una gráfica.

Given Me encuentro en la página de inicio de la aplicación

When Pulso en la opción *Simulations* del menú

And Pulso el botón de ver de la primera simulación que aparece

Then Aparece una gráfica lineal junto con la lista de restricciones y parámetros de la ismulación.

Feature: Visualizar una simulación mediante una gráfica.

Given Me encuentro en la página de inicio de la aplicación

When Pulso en la opción *Simulations* del menú

And Pulso el botón de ver de la primera simulación que aparece

And Pulso un punto de la gráfica principal

Then Aparece un gráfico de barras sobre la información de la iteración seleccionada.

7.1.11. Un usuario puede exportar a Excel una simulación.

Feature: Exportar una simulación a excel.

Given Me encuentro en la página de inicio de la aplicación

When Pulso en la opción *Simulations* del menú

And Pulso el botón de exportar de la segunda simulación que aparece

Then Se descarga el fichero excel.

7.1.12. Un usuario puede eliminar una simulación.

Feature: Eliminar una simulación.

Given Me encuentro en la página de inicio de la aplicación

When Pulso en la opción *Simulations* del menú

And Pulso el botón de eliminar de la segunda simulación que aparece

And Pulso el botón *Delete* para confirmar la eliminación

Then La simulación se elimina y aparece un mensaje de éxito.

Feature: Cancelar la eliminación de una simulación.

Given Me encuentro en la página de inicio de la aplicación

When Pulso en la opción *Simulations* del menú

And Pulso el botón de eliminar de la segunda simulación que aparece

And Pulso el botón *Cancel* para confirmar la eliminación

Then La simulación no se elimina.

7.1.13. Un usuario puede comparar distintas simulaciones.

Feature: Comparar dos simulaciones.

Given Me encuentro en la página de inicio de la aplicación

When Pulso en la opción *Compare simulations* del menú

And Pulso el botón *Simulations*

And Selecciono la primera y tercera simulación

Then Aparecen las dos simulaciones una debajo de otra.

Feature: Comparar todas las simulaciones.

Given Me encuentro en la página de inicio de la aplicación

When Pulso en la opción *Compare simulations* del menú

And Pulso el botón *Simulations*

And Selecciono todas las simulaciones

Then Aparecen todas las simulaciones una debajo de otra.

8

Conclusions and Futures Lines of Research

8.1. Conclusions

As a result of this work, an application has been obtained that allows simulating the behavior and evolution of COVID-19 in a given population. The definition of the population on which the simulation is to be carried out is established according to some input parameters that the user enters, as well as the restrictions to be applied to the population. In addition, the application allows the management of the different simulations created, so that they can be listed, viewed, deleted, exported to Excel and the results can be compared between them.

After carrying out this work successfully, a tool has been obtained that adapts with great flexibility to any real population, and that by applying different restrictions, quite successful results are obtained.

In the development of this work, different difficulties have been encountered that have been solved. The first, more than a difficulty, was to find the closest way to reality to simulate the behavior of a person. For this, the articles that have been written on which aspects of daily life most influence the incidence of the virus have been of great help [12]. Later, once the idea was materialized and implemented in the MESA server, the next difficulty was to introduce this in the application with Flask and React. Finally, it was decided to open a separate page, that is, to launch the MESA server itself from the application's Flask server. And finally, the least difficult was learning to develop the application in React, but since it is very intuitive and there are multiple libraries that help to make an application pleasant and easy, it did not take long to learn the technology.

With the development of technologies in recent decades, the application of these for the detection and prevention of emergencies and decision-making in these is a great success because real situations can be simulated and from these we can put ourselves before and avoid many catastrophes .

8.2. Future lines of Research

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

9

Conclusiones y Líneas Futuras

9.1. Conclusiones

Como resultado de este trabajo, se ha obtenido una aplicación que permite simular el comportamiento y evolución de la COVID-19 en una población determinada. La definición de la población sobre la que se va a realizar dicha simulación, se establece acorde con unos parámetros de entrada que el usuario introduce, así como las restricciones que se quiere aplicar a dicha población. Además la aplicación permite la gestión de las distintas simulaciones creadas, de modo que se pueden listar, visualizar, eliminar, exportar a Excel y comparar los resultados entre ellas.

Tras la realización de este trabajo con éxito, se ha obtenido una herramienta que se adapta con una gran flexibilidad a cualquier población real, y que mediante la aplicación de distintas restricciones se obtienen unos resultados bastantes acertados.

En el desarrollo de este trabajo se han encontrado distintas dificultades que han ido siendo solventadas a medida que avanzaba el proceso. La primera, más que una dificultad, era encontrar la forma más aproximada a la realidad de simular el comportamiento de una persona. Para esto ha sido de gran ayuda los artículos que se han realizado sobre qué aspectos de la vida cotidiana influyen más en la incidencia del virus [12]. Posteriormente una vez materializada la idea e implementada en el servidor MESA, la siguiente dificultad era introducir esto en la aplicación con Flask y React. Finalmente se optó por abrir una página aparte, es decir lanzar el propio servidor de MESA desde el servidor Flask de la aplicación. Y por último, lo menos difícil o la menor dificultad era aprender a desarrollar la aplicación en React, pero debido a que es muy intuitivo y hay múltiples librerías que ayudan a realizar una aplicación agradable

y de forma fácil, tampoco llevo mucho tiempo el aprendizaje de la tecnología.

Con el desarrollo de las tecnologías en las últimas décadas, la aplicación de estas para la detección y prevención de emergencias y la toma de decisiones en estas es un gran acierto ya que se pueden simular situaciones reales y a partir de estas poder anteponernos y evitar muchas catástrofes.

9.2. Líneas Futuras

Del trabajo expuesto aquí pueden desarrollarse varias líneas futuras. Una de ellas sería la introducción de la Inteligencia Artificial, que mediante la introducción de datos y algoritmos se pueda obtener un resultado de las simulaciones aún más exacto dando como resultado una herramienta más fiable y ajustada a la realidad.

Además, también puede partirse de esta herramienta para ver cómo evolucionaría la vacunación. Desarrollando dentro de esta un sistema de vacunación que podría tener distintos parámetros de entrada y así poder adaptarlo a una situación real. A partir de ahí se podría observar la situación en la que ahora mismo nos encontramos, la herramienta podría predecir cómo avanzaría el sistema de vacunación con unos recursos y una situación sanitaria específica y a la vez cómo avanza la expansión del virus que aún se encuentra presente en la población.

Bibliografía

[1] Wikipedia. https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software). 24 de Marzo de 2021.

[2] Wikipedia. https://es.wikipedia.org/wiki/Python. 11 de Marzo de 2021.

[3] Akademus. https://www.akademus.es/blog/programacion/principales-usos-python/. 11 de Marzo de 2021.

[4] MESA. https://mesa.readthedocs.io/en/master/overview.html. 11 de Marzo de 2021.

[5] React. https://es.reactjs.org/. 18 de Marzo de 2021.

[6] Rsuite. https://rsuitejs.com/.

[7] Material- UI. https://material-ui.com/.

[8] Rechart. https://recharts.org/en-US/.

[9] Operational Research Society. https://link.springer.com/article/10.1057/jos.2010.3.

[10] Instituto de Investigaciones Geográficas, Universidad Nacional de Luján. https://ri.conicet.gov.ar/bitstream/handle/11336/107719/CONICET_Digital_Nro.35741ff1-2717-42c6-8a86-4be3970462be_A.pdf?sequence=2&isAllowed=y.

[11] Universidad de Guadalajara, Mexico. https://www.sciencedirect.com/science/article/abs/pii/S001048252030192X.

[12] Petrônio C.L.Silva, Paulo V.C.Batista, Hélder S.LimabMarcos A.Alves, Frederico G.Guimarães, Rodrigo C.P.Silva. https://www.sciencedirect.com/science/article/abs/pii/S0960077920304859.

[13] Organización Mundial de la Salud. https://www.who.int/es/emergencies/diseases/novel-coronavirus-2019/advice-for-public. 3 de Junio de 2021.

[14] Ministerio de Sanidad. https://www.mscbs.gob.es/profesionales/saludPublica/ccayes/alertasActual/nCov/img/Infografia_6M.png. 3 de Junio de 2021.

[15] Ministerio de Sanidad. https://www.mscbs.gob.es/profesionales/saludPublica/ccayes/alertasActual/nCov/documentos/ITCoronavirus.pdf. 3 de Junio de 2021.

[16] Cucumber. https://cucumber.io/docs/gherkin/reference/.

Apéndice A
Manual de Instalación

En la siguiente sección se explica cómo se puede desplegar la aplicación de forma local en su equipo.

En primer lugar se abre el proyecto en cualquier IDE válido para los lenguajes utilizados, para el desarrollo de la aplicación se ha utilizado PyCharm y Visual Studio Code. Después se deben abrir dos terminales y acceder a la parte del servidor mediante líneas de comando y a la parte del cliente. Posteriormente el despliegue se divide en dos partes.

Por un lado tenemos la parte del servidor, para poder ejecutar esta parte es necesario tener la versión de Python 3.6. Este se encuentra realizado en Flask y que tiene distintas dependencias. Para instalar dichas dependencias ejecutamos *pip install -r requirements.txt*. Una vez que la instalación haya terminado debemos ejecutar *python app.py* y nuestro servidor se deberá iniciar correctamente en la ruta *http://127.0.0.1:5000/*, este no cuenta con interfaz por lo que no podemos acceder a él directamente.

Por otra lado tenemos la parte del cliente, donde podremos acceder a la aplicación. Para desplegar únicamente debemos ejecutar *npm start* y la aplicación se arrancará automáticamente en el navegador por defecto en la dirección *http://localhost:3000/*. Puede que haga falta instalar alguna dependecias, en ese caso usando la isntrucción *npm install* más la dependencia necesaria sería suficiente.

Apéndice B
Manual de Usuario

Este manual se encuentra dividido conforme a las funcionalidades que proporciona la aplicación. Cuando esta se inicia aparece la página de la Figura 15.

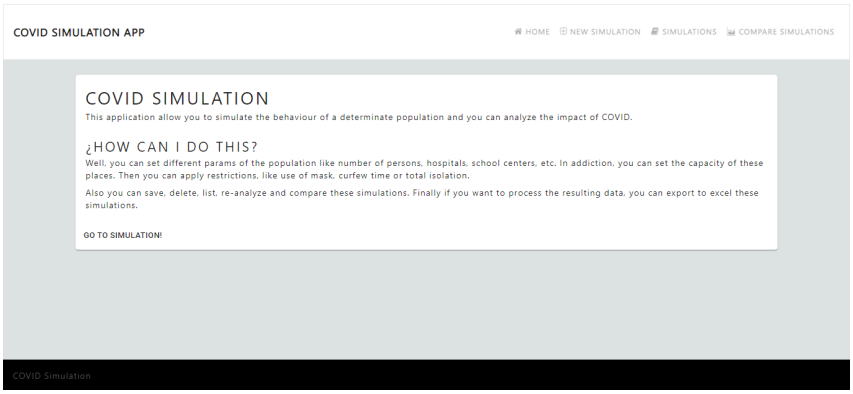

Figura 15: Página de inicio

B.1. Nueva simulación

Para iniciar una nueva simulación, haga clic en la opción *NEW SIMULATION* (1) o pulse el enlace *GO TO SIMULATION!* (2). Véase la Figura 16.

Posteriormente aparecerá la página de la Figura 17, haga clic en el botón *START*, para abrir el servidor de la simulación.

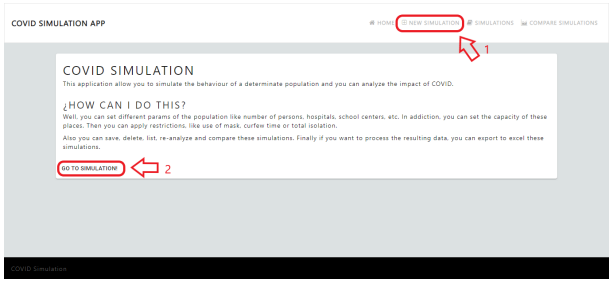

Figura 16: Página de inicio - Nueva simulación

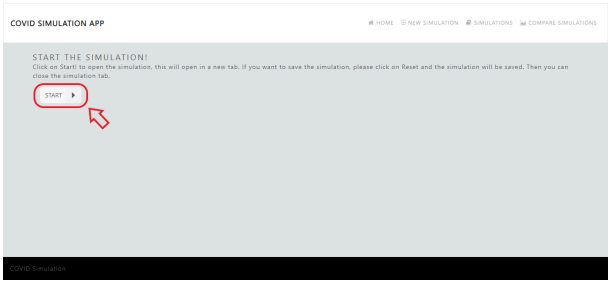

Figura 17: Nueva simulación

Se iniciará el servidor y aparecerá la interfaz de la simulación (Figura 18, Figura 19 y Figura 20). Aquí puede aplicar distintos parámetros y restricciones (1) para la simulación, además de establecer la velocidad de la simulación (2).

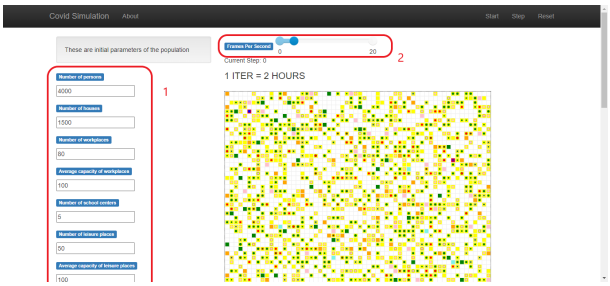

Figura 18: Servidor MESA

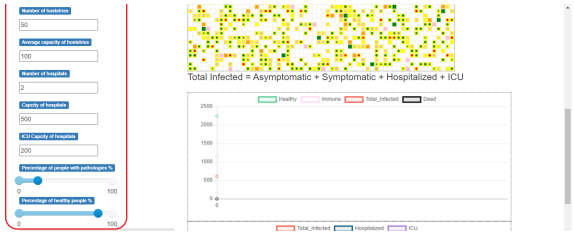

Figura 19: Servidor MESA

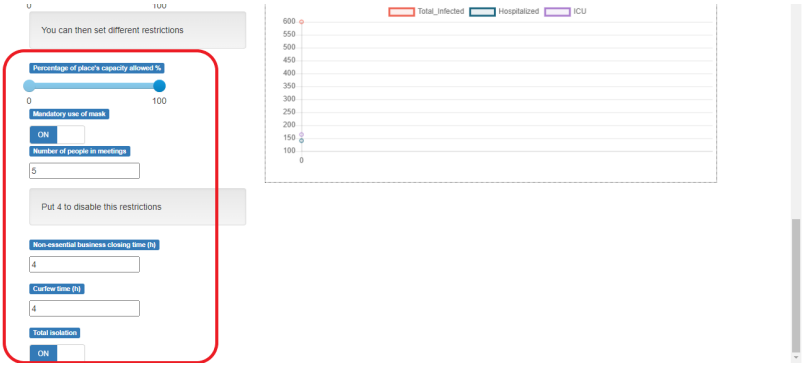

Figura 20: Servidor MESA

Una vez que se introduzcan los parámetros deseados haga clic en el botón *Reset* (1), para aplicar los cambios que se han introducido. Después haga clic en *Start* (2) para iniciar la simulación de forma automática. Por otro lado, si desea avanzar de forma manual las iteraciones de la simulación, haga clic en *Step* (3) para pasar a la siguiente iteración. Una vez que la simulación haya sido iniciada y tenga más de una iteración, podrá guardarla haciendo clic en el botón *Reset* (1). Véase la Figura 21.

Para detener la simulación solo tiene que hacer clic en el botón *Stop* (Figura 22).

Figura 21: Servidor MESA - Start

Figura 22: Servidor MESA - Stop

B.2. Listar simulaciones

Haga clic en la opción *SIMULATIONS* del menú para listar las simulaciones (Figura 23) y aparecerá una tabla con todas ellas (Figura 24).

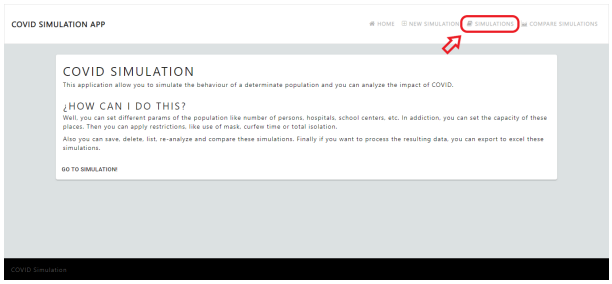

Figura 23: Página de inicio - Simulaciones

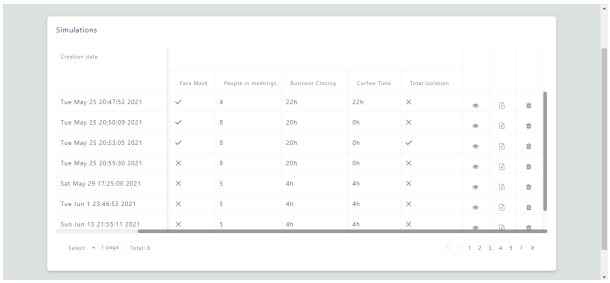

Figura 24: Tabla de simulaciones

B.2.1. Ver una simulación

Si desea visualizar una simulación, deslice la tabla horizontalmente hasta que aparezca el botón de visualizar representado por un ojo (Figura 25).

Haga clic en dicho botón y aparecerán los valores de la simulación en un gráfico junto con los parámetros y restricciones como se muestra en la Figura 26.

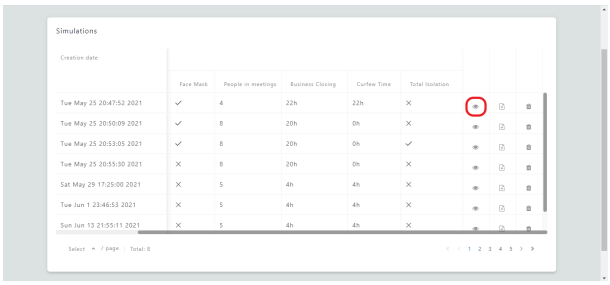

Figura 25: Simulaciones - Visualizar

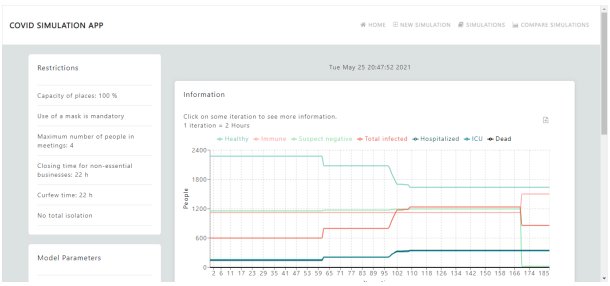

Figura 26: Visualizar simulación

Si desea ver los valores exactos en cada punto deslice el puntero sobre la gráfica, y si quiere ver los valores de una iteración específica haga clic sobre esta y aparecerá debajo de la primera gráfica, un gráfico de barras con los valores correspondientes, tal y como aparece en la Figura 27.

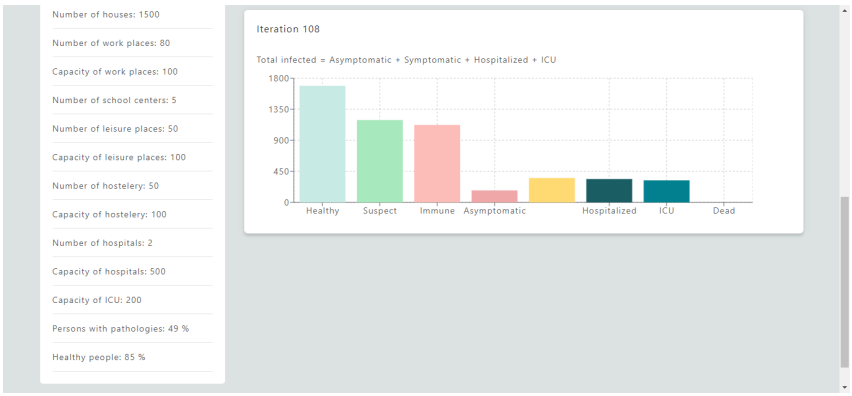

Figura 27: Visualizar simulación - Diagrama de barras

B.2.2. Eliminar una simulación

Para eliminar una simulación, deslice la tabla horizontalmente hasta que aparezca el botón con el símbolo de la papelera y haga clic en él (Figura 28).

Después aparecerá una ventana para confirmar la eliminación. Haga clic en el botón *Delete* para eliminar la simulación o en el botón *Cancel* si desea cancelarla (Figura 29).

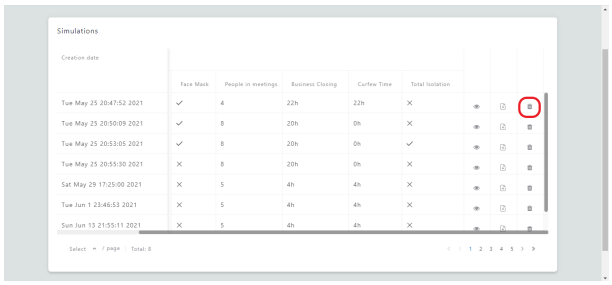

Figura 28: Simulaciones - Eliminar

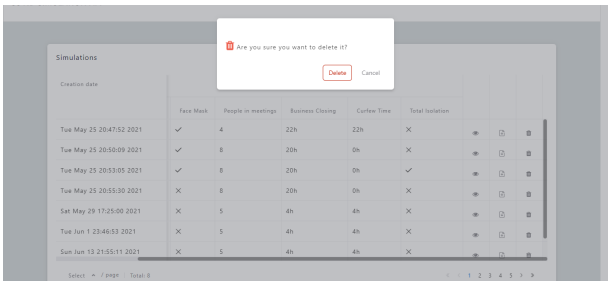

Figura 29: Confirmar la eliminación

Posteriormente si ha aceptado la eliminación aparecerá un mensaje de éxito indicando que se ha eliminado correctamente (Figura 30).

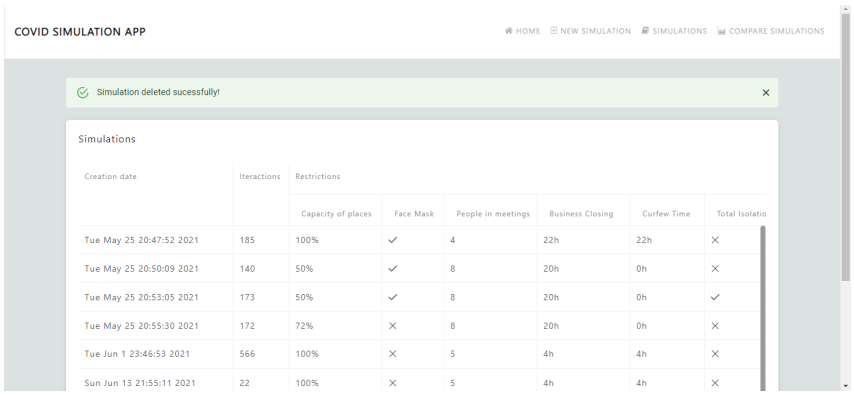

Figura 30: Mensaje de éxito de eliminación

B.2.3. Exportar una simulación

Para exportar una simulación, deslice la tabla horizontalmente hasta que aparezca el botón con el símbolo de exportar (1). Después haga clic en él y comenzará la descarga (2). Véase la Figura 31.

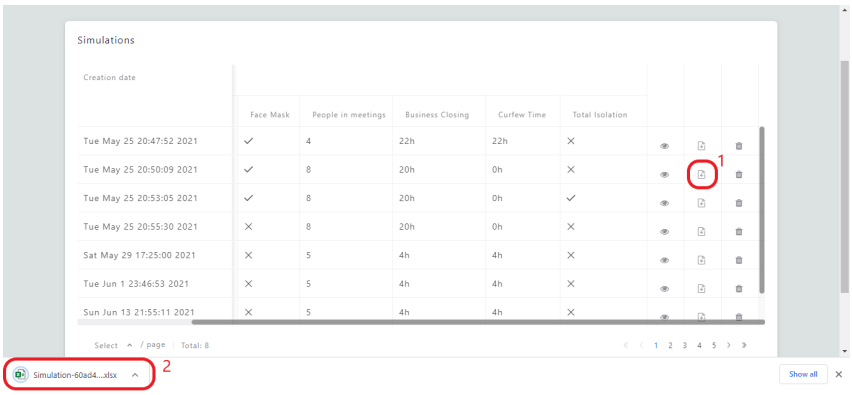

Figura 31: Simulaciones - Exportar a Excel

B.3. Comparar simulaciones

Si desea comparar las gráficas de varias simulaciones, haga clic sobre la opción *COMPARE SIMULATIONS* del menú (Figura 32).

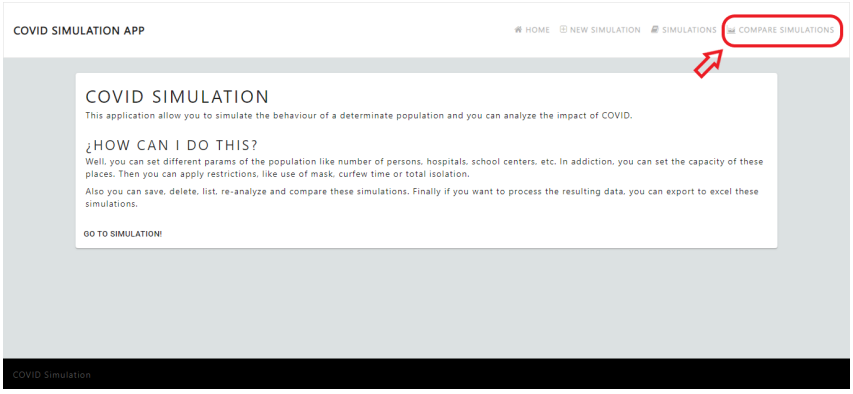

Figura 32: Página de inicio - Comparar simulaciones

Aparecerá la página de la Figura 33, haga clic en el botón *Select simulations*.

Y posteriormente se abrirá una pantalla a la izquierda que contiene una tabla con las simulaciones como aparece en la Figura 34.

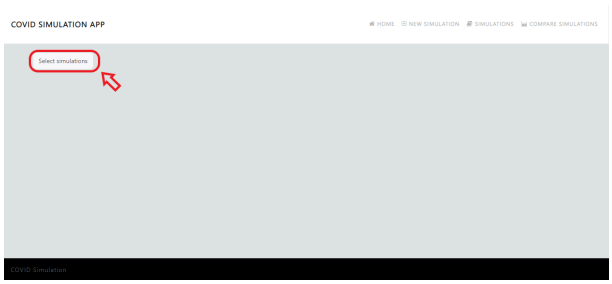

Figura 33: Comparar simulaciones - Seleccionar

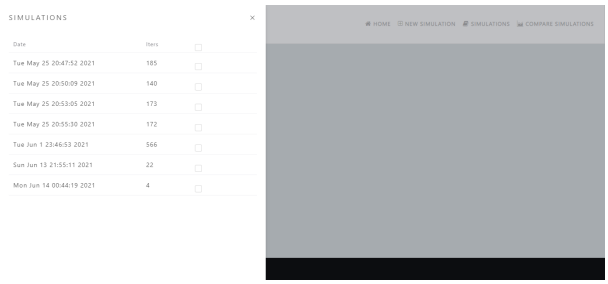

Figura 34: Comparar simulaciones - Tabla simulaciones

Selecciona las simulaciones que desea mostrar. Si pasa el puntero por la fecha de las simulaciones aparecerá las restricciones de dicha simulación (Figura 35).

Una vez haya seleccionado las simulaciones estas aparecerán una tras otra como en la Figura 36.

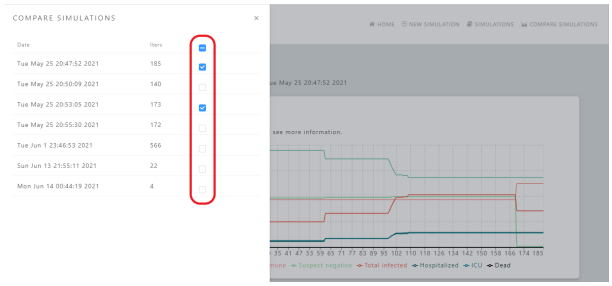

Figura 35: Comparar simulaciones - Tabla simulaciones

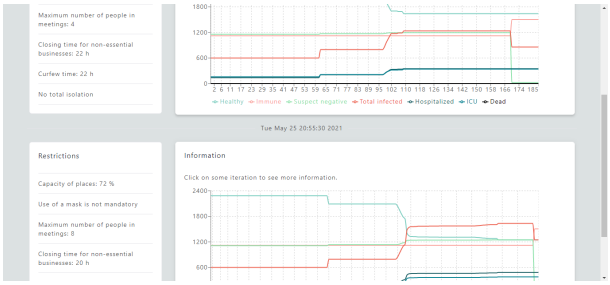

Figura 36: Comparar simulaciones - Gráficas

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA