Project features:

Our project has a Role-based access control; the roles available are Researcher, Doctor and patient.

Patients: Can search for diseases by symptoms and name, view disease profiles, and access their medical records including their diagnoses and treatments.

Doctors: Can perform all patient actions, create and modify medical records and diagnoses, delete diagnoses and create an XML and HTML file of a patient's full Medical Record.

Researchers: Can create/modify/delete disease attributes, symptoms, and treatment plans. They can also generate and search disease simulations based on virtual populations.

Our database will store different infectious diseases, and a user will be able to access each of them, update them, insert them, and delete them.

The **diseases** will contain features like an ID, a name, symptoms, mortality rate, infectious rate, cause (virus or bacteria), disease development (incubation period, development period, convalescence period), and treatment plan with an ID.

Will include a comments section like: this disease is prevalent in…. or Seasonal disease…

or for example the real name of each virus or bacteria.

Our database will also store Symptoms and Treatments linked to each disease.

The **symptoms** will be given an ID, name, and description including pain management (severe or mild).

The **treatments** will be given an ID, a name, and an observation section where the Doctor user can input the amount of treatment the patient should take, when to take it etc…

Our database will also store Patients with their own Medical Record in which Diagnoses linked to Diseases and specific Treatments will be stored and listed.

The **patients** each have an ID, a name, a surname and a Date of Birth although these patients have more info since these patients have to be registered **Patient Users**, therefore they have additional information as a user like DNI, email, phone number, Insurance Number and sex.

The **diagnoses** belonging to the patient include an ID, a name (may not match the one for the disease), the date it was made, an observation section, the disease that the patient has, and a list of treatments prescribed for that patient to take.

A patient's **Medical Record** will be accessed by that **Patient** and their **Doctors.**

Something special about our database is that the **Researcher** can create **Simulations** on **Virtual Populations** with a certain disease based on parameters like the initial number of population % of infected, % of healthy, and % of immune people in that population the user must also input an immunity period parameter which determines how much time a virtual person can be immune to the disease (like with vaccines and stuff).

As it was mentioned before these are the available options for each user:

Actors:

- Patients
    - Log into the web, access WikiDiseases (search diseases by symptom or by name, and search a disease by the symptoms that most match it), search symptoms, see the development graph of a disease, see their medical record, and edit some attributes in their profile.
- Doctors
    - Log into the web, access WikiDiseases, search symptoms, see the development graph of a disease, and add, modify and delete, a patient's diagnosis in a medical record, they can also create an XML and HTML file of a patient's full medical record.

- Researcher
    - Log into the web, add, modify and delete diseases, symptoms and treatments. Create a simulation on Virtual Populations with a certain disease based on parameters and also search for simulations already created.

The data will be manipulated for different reasons:

- to create a simulation of the disease development with its parameters
- show how many diseases are linked with certain symptoms and treatments
- show how a disease spreads
- determine a disease based on symptoms
- C
- R
- U
- D

Requirement: Capability

Functional: what the system does

- C (every entity)
- R (searcher of diseases (WikiDiseases)→ all info, symptoms, treatment…)
- U (every entity except medical record)
- D (every entity except medical record)
- Searcher of Diseases by symptoms and name
- Simulation of a disease spread in a population
- Development of a disease in a person
- List of diagnoses of each patient

Non Functional: how the system does

- Simulation:
    - We can only do a simulation of only one disease at a time.

  - Each person can only be in one state(ill or.., alive or…).
  - The user will enter the parameters of the simulation, except the characteristics of the disease.
  - Use parameters like infectious rate, incubation period, and development period from the disease to simulate its spread and progression.
  - The simulation will conclude when there is no more infection, when there are no more people alive or when the time of the disease development is up.

- Disease Development Graph:
  - Based on the time frames of the development periods, the database will create a graph showing the state of the disease at each point in time. Depending on the time frames the database will estimate a graph.
  - The maximum value is going to be the same for all the graphics of all the diseases. It will not be a quantitative but a qualitative value of the development of a disease.
  - The difference between the graphics will be the slope.
  - The graph of the same disease will remain the same if the data has not been modified.

- Wikidiseases:
  - Users will search for the disease by entering its name or symptoms and the matching disease will be shown. Users can also search the disease development Graph.

- Symptoms research:
  - The user will insert a series of symptoms, and the database will return a list of possible diseases that the patient may have and the number of symptoms that align with each disease.
  - The program will show all the possible diseases based on the symptoms and also there is an option to search for the most matching disease to the symptoms inserted.

More focused outline for the database of infectious diseases:

1. **Database Structure**:
   - Designed a relational database schema with tables for storing different entities such as diseases, symptoms, treatment plans, diagnoses, patients, medical records, virtual populations and simulations.
   - Defined primary and foreign key relationships to establish connections between entities.

2. **Disease Table**:
   - Fields: Disease ID (Primary Key), Name, Infectious Rate, Mortality Rate, Cause (Virus, Bacteria), Incubation Period, Development Period, Convalescence Period, comments section.

3. **Symptoms Table**:
   - Fields: Symptom ID (Primary Key), Name, Pain Management (Mild or Severe).

4. **Treatment Plan Table**:
   - Fields: Treatment Plan ID (Primary Key), Description

5. **Diagnosis Table**:
   - Fields: Diagnosis ID (Primary Key), Name, Date, Disease ID (Foreign Key), Medical Record ID (Foreign Key)

6. **Medical Record Table**:
   - Fields: Medical Record ID (PK), Patient ID (FK).

7. **Patients Table**:
   - Fields: Patient ID (PK), Name, Surname, Date of Birth and username

8. **Virtual Population Table**:
   - Fields: Virtual Population ID (PK), Initial population, % of infected, % of healthy, % of immune, Immunity period, Disease ID (FK)

9. **Simulations Table**:
   - Fields: Simulation ID (PK), total number of infections, total deaths, total immunity, total population, simulation Graph (BLOB), Virtual Population ID (FK).
   -

10. **Many-To-Many Tables**:
    - Disease has Symptoms
    - Disease has Treatments
    - Diagnosis has Treatments

11. **Manipulating Data**:
    - Implemented CRUD (Create, Read, Update, Delete) operations to manage data within the database.

- Created procedures or functions to manipulate data according to specific requirements, such as disease simulation parameters or determining diseases based on entered symptoms.

12. **Simulation**:
    - Developed algorithms or simulations using disease parameters stored in the database to model disease development.
    - Used parameters like infectious rate, incubation period, and development period to simulate disease spread and progression.

13. **User Interface**:
    - Created a user-friendly interface for accessing, updating, inserting, and deleting all the data stored.
    - The user must identify themselves by logging in or filing for registration.
    - Includes forms for entering symptoms to determine potential diseases.
    - Added options for viewing disease profiles, symptoms, treatment plans, and associated comments.

14. **Visual Representation**:
    - Includes the ability to display images of the development of viral or bacterial diseases and the graph of the time spread of the disease during the simulation.

15. **Security and Authentication**:
    - Implemented authentication mechanisms to control access to data within the database.
    - Implemented encryption of passwords.

16. **Users and Roles**:
    - They all share the same attributes: Role Verification Number (PK), DNI, Date of Birth, Email, Name, Surname, Username, Password, sex and a Role ID (FK) can be either of the following 3:
        - Doctor: When registering for an account will be asked for their collegiate number (role verification ID)
        - Patient: When registering for an account will be asked for their Insurance number (role verification ID)
        - Researcher: When registering for an account will be asked for their collegiate number (role verification ID)

USES CASES: IDSw (infectious disease storage web)

Name:	***Sign up for an account.***
Preconditions:	Be a new user.
Postconditions:	log in and see a different menu depending on your user type
Primary Actor(s)	Patient, doctor and researcher.
Main Scenario:	**Action:**
	1. The user presses the sign-up button. 2. The user has to specify the type of user they are. 3. The user fills in the information. 4. Users press the button “Create”.
Extensions:	**Branching Action:**
	3.1 If the user is a patient, they have to fill in the name, surname, birthday, sex, email, phone number, DNI, social security number, username and password. 3.2 If the user is a doctor or a researcher, they have to fill in the professional ID, name, surnames, birthday, sex, email, phone number, DNI, username and password.

Name:	***Login into an account.***
Preconditions:	Users need to have signed up for an account.
Postconditions:	see a different menu depending on your user type
Primary Actor(s)	Patient, doctor and researcher.
Main Scenario: (pasos que sigue después)	**Action:**
	1.User fills login information. 2. User press the continue button
Extensions: (subpasos)	**Branching Action:**
	2.1. If the user doesn't exist, the web returns to the first frame and suggests signing in. 2.2 If the user exists, the web go to a frame with the menu of each actor.

Name:	***Log out of the account***
Preconditions:	Users need to have login into an account.
Postconditions:	You see the initial menu (sign in, log in …)
Primary Actor(s)	Patient, doctor and researcher.
Main Scenario:	**Action:**
	1. User presses the user icon (on the right upper corner). 2. User presses the logout button.

Name:	***Search by symptoms.***
Preconditions:	Users need to have login into an account.
Postconditions:	Users see compatible diseases.
Primary Actor(s)	Patient and doctor.
Main Scenario:	**Action:**
	1. User has to press the “search” button. 2. User presses the “by symptom” button. 3.User write the symptom in the search bar. 4. The web shows the matched symptoms. 5.The user chooses the desired symptoms.

Name:	***Create a symptom.***
Preconditions:	Search by symptoms (at the end button to do it)
Postconditions:	Return to read symptoms menu with update list
Primary Actor(s)	Researcher.
Main Scenario:	**Action:**
	1. User has to press the “search” button. 2. User presses the “by symptom” button. 3. User presses the “Add” button. 4. User writes the new symptom. 5. User presses the “Save” button. 6. The web is going to return to the search by symptoms menu with the update list.
Extensions:	**Branching Action:**
	5.1 If the symptom is already created the web returns to the list without any change and a message telling “Symptom is already in the system”.

Name:	***Read symptoms.***
Preconditions:	Users need to have login into an account. inicial menu
Postconditions:	list with the symptoms
Primary Actor(s)	Patient, doctor and researcher.
Main Scenario:	**Action:**
	1. User has to press the “search” button. 2. User presses the “by symptom” button. 3. User sees the frame with the search bar and the list of symptoms.

Name:	***Eliminate a symptom.***
Preconditions:	Read symptoms (at the end button to do it)
Postconditions:	update list of symptoms
Primary Actor(s)	Researcher.
Main Scenario:	**Action:**
	1. User has to press the “search” button. 2. User presses the “by symptom” button. 3. User selects the symptom. 4. User presses the “Eliminate” button. 5. User presses the button of the warning elimination.. 6. The web is going to return to the search by symptoms menu with the update list.

Name:	***Modify a symptom.***
Preconditions:	Read symptoms (at the end button to do it) and been able to edit the content
Postconditions:	update list of symptoms
Primary Actor(s)	Researcher.
Main Scenario:	**Action:**
	1. User has to press the “search” button. 2. User presses the “by symptom” button. 3. User selects the symptom. 4. User presses the “Modify” button. 5. User changes the symptom. 6. User presses the “save” button. 7. The web is going to return to the search by symptoms menu with the update list.
Extensions:	**Branching Action:**
	4.1 If the user selects more than one, the web sends a message of warning..

Name:	***Simulation of a disease.***
Preconditions:	Users need to have login into an account. has to introduce the parameters for the disease
Postconditions:	See the result on the screen
Primary Actor(s)	Researcher.
Main Scenario:	**Action:**
	1. User has to press the “simulation” button. 2. The web asks the user to look for a disease.

	3. User searches for the disease and presses the desired one. 4. User introduces the parameters that the web asks. 5.The web shows a record with the total number of infected people, deaths and the final number of inmune people and final number of alive people. 6.The web also shows the simulation as a graph. 7. If the user wants to save the image, he has to press the “save” button.
Extensions:	**Branching Action:**
	4.1 User introduces “number of people”. 4.2 User introduces “ immunity period”. 4.3 User introduces “% ill”. 4.4 User introduces “% inmune”. (the healthy people will be obtain operating)

Name:	***Search a disease by name.***
Preconditions:	Users need to have login into an account. and access wikiDisease
Postconditions:	you can see the data of it and a button to save it
Primary Actor(s)	Patient and doctor.
Main Scenario:	**Action:**
	1. User has to press the “search” button. 2. User presses the “by name” button. 3.User write the disease in the search bar.

	4. The web shows the matched disease. 5.The user chooses the desired disease.

Name:	***Delete a disease.***
Preconditions:	Access to wikiDiseases, and search one.
Postconditions:	Return to the main menu of wikidisease
Primary Actor(s)	Researcher.
Main Scenario:	**Action:**
	1. User has to press the “search” button. 2. User presses the “by name” button. 3. User selects the disease. 4. User presses the “Eliminate” button. 5. User presses the button of the warning elimination. 6. The web is going to return to the search by name menu with the update list.

Name:	***See the development graph.***
Preconditions:	The user must have searched a disease.
Postconditions:	the picture and option to leave
Primary Actor(s)	Patient and doctor.
Main Scenario:	**Action:**
	1. User makes a double click on the disease that he wants to see. 2. The web shows the development graph of the desired disease below the rest of information..

Name:	***Save a document in the pc.***
Preconditions:	The user must have searched a disease, created a simulation or there must have been an existing diagnosis.
Postconditions:	it is save in the computer and you return to the menu
Primary Actor(s)	Patient, doctor and Researcher.
Main Scenario:	**Action:**
	1. User presses the “download” button.

Name:	***Modify a disease.***
Preconditions:	Users need to have login into an account. and enter wikiDisease and search one, and press a button to modify, been able to edit the content
Postconditions:	See the disease with the updates
Primary Actor(s)	Researcher.
Main Scenario:	**Action:**
	1. User has to press the “search” button. 2. User presses the “by name” button. 3. User selects the disease. 4.Modify button appears. 5. User presses the “Modify” button. 6. User changes the disease. 7. User presses the “save” button. 8. The web is going to return to the search by name menu with the update list.
Extensions:	**Branching Action:**

	4.1 If the user selects more than one, the button of “modify” disappears.

Name:	***Create a disease.***
Preconditions:	enter wikiDiseases press a button to add one
Postconditions:	See the disease created
Primary Actor(s)	Researcher.
Main Scenario:	**Action:**
	1. User has to press the “search” button. 2. User presses the “by name” button. 3. User presses the “Add” button. 4. User writes the new disease. 5. User presses the “Save” button. 6. The web is going to return to the search by name menu with the update list.
Extensions:	**Branching Action:**
	5.1 If the disease is already created the web returns to the list without any change and a message telling “Disease is already in the system”.

Name:	***Create the record of a patient (expediente)***
Preconditions:	Users need to have login into an account. and searched a patient
Postconditions:	See the new record
Primary Actor(s)	Doctor.

Main Scenario:	**Action:**
	1.The records are created when you sign up as a patient.

Name:	***See the record of patients (expediente)***
Preconditions:	Preexisting record
Postconditions:	see the record and if doctor (button to modify it)
Primary Actor(s)	Patient and doctor.
Main Scenario:	**Action:**
	1. User pressed the “record” button. 2. User searches the patient by the HI.ID number and presses it. 3.The record of the patient is seen with a list of the diagnosis.

Name:	***Modify the record***
Preconditions:	Users need to have seen the expedient of the patient. and select that option, and been able to edit the content
Postconditions:	See the update record.
Primary Actor(s)	Doctor.
Main Scenario:	**Action:**
	1. The record is modified when the user modifies its information.

Name:	***Create a diagnosis.***

Preconditions:	menu, look for recordclick, search patient, and select new diagnosis
Postconditions:	See the update expedient with the new diagnosis
Primary Actor(s)	Doctor.
Main Scenario:	**Action:**
	1. User pressed the “record” button. 2. User searches the patient by the HI.ID number and presses it. 3.The record of the patient is seen. 4. User presses the “add diagnosis” button. 5. User write the diagnosis (date, name, disease, treatment and observations) 6. User presses the “create” button. 7. User sees the update record.

Name:	***Modify a diagnosis.***
Preconditions:	Pre Existing diagnosis and been able to edit the content
Postconditions:	see the update diagnosis
Primary Actor(s)	Doctor.
Main Scenario:	**Action:**
	1. User pressed the “record” button. 2. User searches the patient by the HI.ID number and presses it. 3.The record of the patient is seen. 4. User presses the check box of one diagnosis. 5. Two buttons appear: “modify” and “delete”.

	6. User selects the “modify” button. 7. User modifies the diagnosis and presses “save”. 8. The web shows the update record.
Extensions:	**Branching Action:**
	5.1 If the user selects more than one, the “modify” button disappears.

Name:	***Eliminate a diagnosis.***
Preconditions:	Pre Existing diagnosis. and press a button to eliminate
Postconditions:	Return to the updated record.
Primary Actor(s)	Doctor.
Main Scenario:	**Action:**
	1. User pressed the “record” button. 2. User searches the patient by the HI.ID number and presses it. 3.The record of the patient is seen. 4. User presses the check box of one or more diagnosis. 5. Two buttons appear: “modify” and “delete”. 6. User selects the “delete” button. 7. User presses the button of the warning elimination. 8. The web is going to return to the record with the update list of diagnosis.
Extensions:	**Branching Action:**

	5.1 If the user selects more than one, the “modify” button disappears.

Name:	***Read a diagnosis.***
Preconditions:	Pre Existing diagnosis, and record searcher
Postconditions:	Access the diagnosis information
Primary Actor(s)	Patient and doctor.
Main Scenario:	**Action:**
	1. User pressed the “record” button. 2. User searches the patient by the HI.ID number and presses it. 3.The record of the patient is seen with a list of the diagnosis. 4. User presses the diagnosis that he wants to see. 5. The web shows the diagnosis.

Name:	***Create a treatment.***
Preconditions:	Users need to have login into an account of a researcher, look for a disease (wikiDisease), select a new treatment option. new treatment in the front menu
Postconditions:	.txt page to write treatment
Primary Actor(s)	Researcher.
Main Scenario:	**Action:**

	1. 1. User has to press the “treatment” button. 2. User presses the “Add” button. 4. User writes the new treatment. 5. User presses the “Save” button. 6. The web is going to return to the menu with the update list.
Extensions:	**Branching Action:**
	5.1 If the treatment is already created the web returns to the list without any change and a message telling “Treatment is already in the system”.

Name:	***Modify a treatment.***
Preconditions:	Preexisting treatment. and being able to edit the content. or add or substrate from the disease attachment (by buttons).
Postconditions:	page with the treatment save
Primary Actor(s)	Researcher.
Main Scenario:	**Action:**
	1. User has to press the “treatment” button. 2. User selects the treatment.. 4.Modify button appears. 5. User presses the “Modify” button. 6. User changes the treatment. 7. User presses the “save” button. 8. The web is going to return to the menu with the update list.
Extensions:	**Branching Action:**

	4.1 If the user selects more than one, the button of “modify” disappears.

Name:	***Eliminate a treatment.***
Preconditions:	Preexisting treatment. press a button to suppress the treatment in the front menu treatment section
Postconditions:	the treatment is out from the database you return to the front menu
Primary Actor(s)	Researcher.
Main Scenario:	**Action:**
	1. User has to press the “treatment” button. 2. User selects the treatment. 3. User presses the “Eliminate” button. 4. User presses the button of the warning elimination. 5. The web is going to return the update list.

Name:	***Read a treatment .***
Preconditions:	Preexisting treatment. Doctor from a diseases display section of treatment ()
Postconditions:	The treatment is seen.
Primary Actor(s)	Doctor, researcher,.
Main Scenario:	**Action:**
	1. User pressed the “treatment” button. 2. User makes a double click in the treatment that he wants to see.

	3. The web shows the treatment.