Project requirements

It is necessary to create a web application that allows monitoring of the work of several medical institutions. It is necessary to monitor information about: medical personnel employed by medical institutions, patients and examinations performed. For each of the entities (Organization, Practitioner, Patient and Examination) it is necessary to implement the following functions:

-**Inserting data about new entity**: user should be able to insert a new entity if the validation rules are fulfilled. If there are some validation errors the app needs to show validation messages to the user so that the user has the opportunity to fix the errors.

-**Show data about all entities of certain type**: server-side pagination should be implemented; user should be able to change number of entities displayed per page; implement server-side sorting and allow user to sort regarding different columns and also different directions (asc,desc), Any change to the pagination or sorting made by the user will trigger refresh of the displayed data. Users should be able to filter the received data using one or more filters appropriate to the entity in question.

-**Show detailed information about specific entity**: after the display of all entities of specific type, user can click on an entity (or an icon/button) to see more detailed information about that entity.

-**Editing data about specific entity**: editing data for some entity is allowed if validation rules are fulfilled, in case validation fails user should see validation error message.

-**Deleting entity**: user can delete the entity. Before deleting the entity, user is presented with a confirmation message where he must confirm his action.

In the following section will be listed all required functionalities, together with data structure for each entity.

Organization

(CRUD – inserting new Organization, show all existing Organizations, show/edit detailed information about Organization, delete Organization)

This entity represents a medical institution with the main purpose of providing some healthcare service.

Property name	Type	Required	Unique	Constraint
identifier	String	no	yes	Minimal number of characters 5
active	Boolean	yes	no	
type	String	yes	no	Related to table OrganizationType
name	String	yes	yes	Minimal number of characters 5
address	String	no	no	
phone	String	no	no	
email	String	no	no	Contains '@' char

Properties description:

- identifier: Identifies organization across multiple systems
- active: Whether the organization's record is still in active use
- type: Used to categorize the organization ('Hospital', 'Insurance Company', 'Educational Institute', 'Clinical Research', 'Other')
- name: Name used for the organization
- address: An address for the organization
- phone: Contact phone
- email: Contact mail

*Show detailed information about specific Organization

On this screen there should also be some info about related entities like number of Practitioners working for that Organization and number of currently running Examinations and total number of performed Examinations.

*Delete Organization

Delete operation is possible only if there are no running examinations being performed by that Organization. Soft Delete: Organization record is not deleted from database during this operation. Set active to false to indicate that this operation is not part of the system anymore

Practitioner

(CRUD – inserting new Practitioner, show all existing Practitioners, show/edit detailed information about Practitioner, delete Practitioner)

This entity represents a person with a formal responsibility in the provisioning of healthcare or related services.

Property name	Type	Required	Unique	Constraint
identifier	String	no	yes	Minimal number of characters 5
active	Boolean	yes	no	
name	String	yes	no	Minimal number of characters 3
surname	String	yes	no	Minimal number of characters 3
gender	String	no	no	[‘male’, ‘female’, ‘other’, ‘unknown’]
birthDate	Date	yes	no	
address	String	no	no	
phone	String	no	no	
email	String	no	no	Contains ‘@’ char
qualification	String	yes	no	[‘Doctor of Medicine’, ‘Medical Assistant’, ‘Nurse Practitioner’, ‘Doctor of Pharmacy’, ‘Certified Nurse Midwife’, ‘Emergency Medical Technician’]

Properties description:

- identifier: An identifier for the person across multiple systems
- active: Whether this practitioner's record is in active use
- name: Name associated with the practitioner
- surname: Surname associated with the practitioner
- gender: Practitioners gender
- address: An address of the practitioner
- phone: Contact phone
- email: Contact mail
- qualification: Certification, licenses, or training pertaining to the provision of care

*Show all Practitioners

One of the possible filters on this view should be ‘Unassigned’ boolean filter indicating Practitioner who is not assigned (employed) to any Organization.

*Delete Practitioner

This operation is possible only if the Practitioner is not performing some Examination.

Soft delete: Practitioner record is not removed from the database, set active to false to indicate that this Practitioner is not part of the system anymore

Note: During insertion of the Practitioner, the system should allow the selection of the Organization practitioner is working for.

Patient

(CRUD – inserting new Patient, show all Patients, show/edit detailed information about specific Patient, delete Patient)

This entity holds Information about an individual receiving health care services

Property name	Type	Required	Unique	Constraint
identifier	String	no	yes	Minimal number of characters 5
active	Boolean	yes	no	
name	String	yes	no	Minimal number of characters 3
surname	String	yes	no	Minimal number of characters 3
gender	String	no	no	[‘male’, ‘female’, ‘other’, ‘unknown’]
birthDate	Date	yes	no	
address	String	no	no	
phone	String	no	no	
email	String	no	no	Contains ‘@’ char
deceased	Boolean	no	no	
maritalStatus	String	no	no	[‘Annulled’, ‘Divorced’, ‘Married’, ‘Polygamous’, ‘Never Married’, ‘Widowed’]

Properties description:

- identifier: An identifier for this patient
- active: Whether this patient's record is in active use
- name: Name associated with the patient
- surname: Surname associated with the patient
- gender: Gender of the patient
- birthDate: Patient's date of birth
- address: An address for the individual
- phone: Contact phone
- email: Contact mail
- deceased: Indicates if the individual is deceased or not
- maritalStatus: Marital status of a patient

*Delete Patient
This operation is possible only if there are no running Examination being performed on the Patient.
Soft delete: Patient record is not removed from the database, set active to false to indicate that this Patient is not part of the system anymore

Note: During insertion of the patient, the system should allow selection of the Organization that is the custodian of the patient record, also it should be possible to set Patient’s nominated primary care provider (general practitioner, need to have qualification ‘Doctor of Medicine’)

Examination

(CRUD – inserting new Examination, show all Examinations, show/edit detailed information about specific Examination, delete Examination)

This entity represents an interaction during which services are provided to the patient.

Property name	Type	Required	Unique	Constraint
identifier	String	no	yes	Minimal number of characters 5
status	String	yes	no	[‘planned’, ‘triaged’, ‘in-progress’, ‘suspended’, ‘finished’, ‘cancelled’, ’entered-in-error’]
serviceType	String	yes	no	Related to table ServiceType
priority	String	no	no	[‘asap’, ‘callback results’, ‘emergency’, ‘routine’, ‘rush reporting’, ‘timing critical’]
startDate	Date	no	no	
endDate	Date	no	no	
diagnosis	String	no	no	

Properties description:

- identifier: Identifier by which this encounter is known
- status: status of the examination
- serviceType: Type of care service provided to the patient
- priority: Indicates the urgency of the encounter
- startDate: starting date of the Examination
- endDate: ending date of the Examination
- diagnosis: Condition patient is treated for

*Show all Examinations

System is expected to handle huge amounts of Examinations so the ability to filter data by almost any property is advisable. Filters for properties that represent known set of values (status, priority) should be modeled as dropdown/combo box where user can select value instead of typing it. Implement filtering by Organization, Practitioner and Patient. If none of the filters are set, system should not return the Examination data.

*Deleting Examination

Soft delete: Examination record is not removed from database during this operation, set status to entered-in-error to indicate that Examination is not part of the system anymore.

Note: During creation of new Examination, the system should allow
-selection of the Patient who is receiving services
-selection of the Organization responsible for providing care services
-selection one or more Practitioners who are providing services

<<special features>>
Consider the following features as bonus requirements once base set has been implemented.

*Authentication
Protect the application from unauthorized access by implementing base authentication.
Extend Practitioner entity with username and password properties. Application will present user with login screen where Practitioner is expected to enter his credentials in order to gain access the system.

*Dashboard
Set Dashboard view as initial view after the login. On this view user should be presented with graphical overview of the system composed of several graphs and widgets.
--Organization cards with number of employed Practitioners and number of currently performed Examinations
--Chart with data for the last 30 days about number of running Examinations each day per Organization
--Chart with employed Partitioners qualification's structure (pie chart with Doctors, Nurses…)
-- << feel free to add any interesting visualization on Dashboard >>

*Bulk import
There are some data like serviceType of the Examination that represent know set of data but is huge in nature and is not easy do maintain manually. Implement functionality where the user can upload CSV file with new set of values for examinations Service Type data.

(!) Entity data structure tables show before hold basic information about entities, its allowed to add additional properties and references to other tables
(!) Independently model the entities OrganizationType and ServiceType, as well as the entities necessary to implemented all other functionalities in the system

(!) Posible values for ServiceType table: [‘Aged Care Assessment’, ‘Aged Residential Care’, ‘Home Care/Housekeeping Assistance’, ‘Acupuncture’, ‘Bowen Therapy’, ‘Blood Donation’, ‘Family Planning’, ‘Immunization’, ‘Optometry’, ‘Osteopathy’, ‘Physiotherapy’, ‘Podiatry’, ‘Endodontic’, ‘Dental’, ‘Oral Surgery’, ‘Emergency Medical’, ‘Psychology’, ‘Dermatology’]

Technical specification:

For the given project assignment, the following technologies requirements are to be followed.

1. It is required to design database structure and implement it in any relational database management system (like *Oracle, MySQL, Postgres, H2* ...). You have to create only a database using database management system but you should use ***hibernate to create and/or update tables in database***. You should have ***data.sql*** file (*in resource folder*) with initialization ***insert*** scripts in order to set initial set of data.

2. It is required to develop Java backend over implemented database. Required technology is Spring Boot. Backend has to communicate with the database, and to return requested data to the user over REST API.

3. Project is required to be a Maven project. You must name your project according to the following template **FirstName-LastName-BE.** Use appropriate folder structure, (base namespace firstname.lastname appropriate packages, manage dependencies, and manage build process with Maven.

4. It is required to use Object relational mapping (ORM) – Java Persistence API (JPA) and Hibernate. Use appropriate JPA annotations. Validate data with Hibernate Validator. As previously stated, use Hibernate to create (update) tables in database.

5. Develop the fronted with ***Angular 13***. You must name your project according to the following template **FirstName-LastName-FE.**

6. It is required to use GIT during development. Required technology is BitBucket. You have to create your *BitBucket* account using company email account. You have to add Djordje Mandic (djordje.mandic@eng.it), as well as Marco Mastroianni (marco.mastroianni@eng.it) and Alessio Mugnaioli (alessio.mugnaioli@eng.it) in ***BitBucket***. You must commit and push changes of your code to remote repository at least **at the end of each day**. You must name your repository according to the following template **FirstName-LastName** and put both BE and FE project in the same repository.

You must have at least two branches ***master and develop.*** You must keep your last stable release in ***develop*** branch. It is recommended to open a feature branch and work there on some feature. For instance, create ***feature/x*** when you work on **x** branch. When you have something stable merge and push the changes to **develop branch**. When you have the release with all required things related to one feature, ***merge the changes to master branch as well.***