

PROMise - Διπλωματική Εργασία

Γιώργος Μαθιουδάκης - csd4674

Overview:

1. Introduction
2. Methodology
3. Demonstration
4. Conclusion
5. Installation Instructions

Introduction

This project is about the development of a marketplace for PROMs/PREMs. The traditional way of the patient having to fill in endless Questionnaires on paper is no longer viable nor time efficient. The doctor can now help more patients from his computer using PROMise , which is the webApp allowing doctors to upload any questionnaire they want with full customization capabilities, store that, update it if needed, and then send it to patients with ease. Patients will be reminded by notifications to fill the questionnaire whenever needed and the doctor will have all the answers stored without any paper getting wasted or worse lost . This part of the project focuses on the questionnaire side of things. The creation of it, the update if needed and the fulfillment along with the creation and update of all response types the doctor wishes to have .

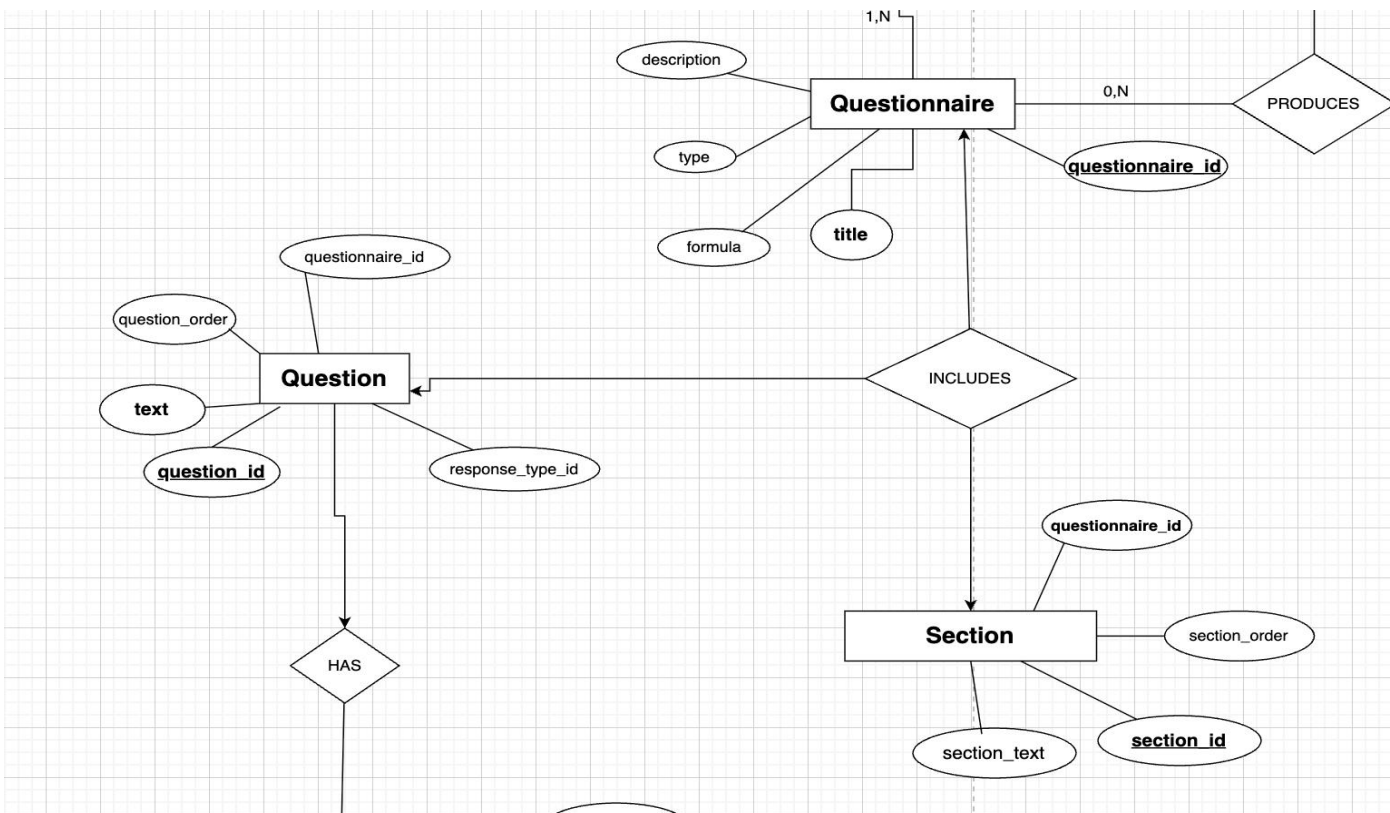
Methodology

Technology Stack:

- ReactJs (HTML, JS)
- TailwindCss
- SpringBoot (Java)
- PostgresQl

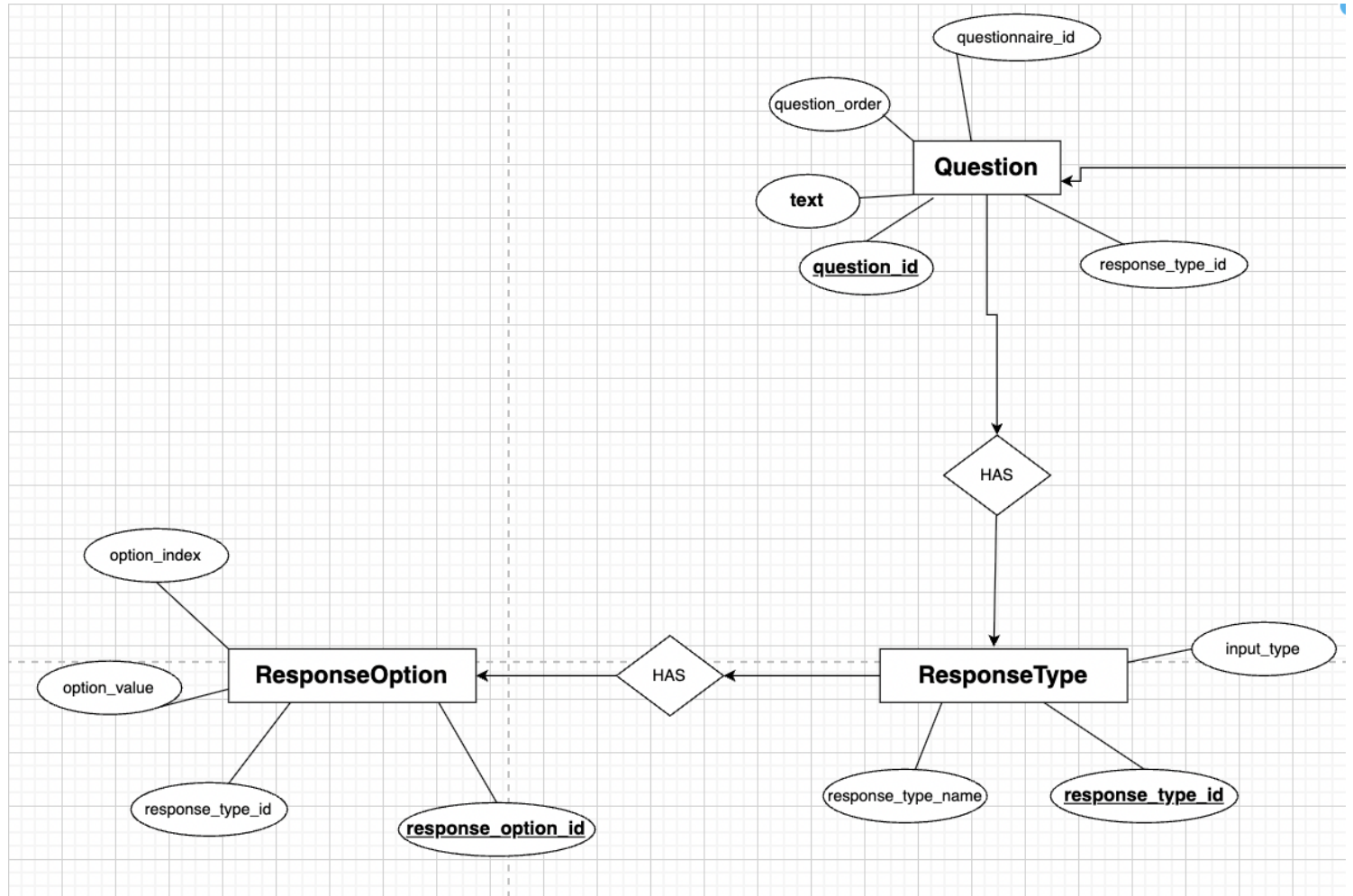
ER for the database step by step

First we have the Questionnaire entity connected to sections and questions .



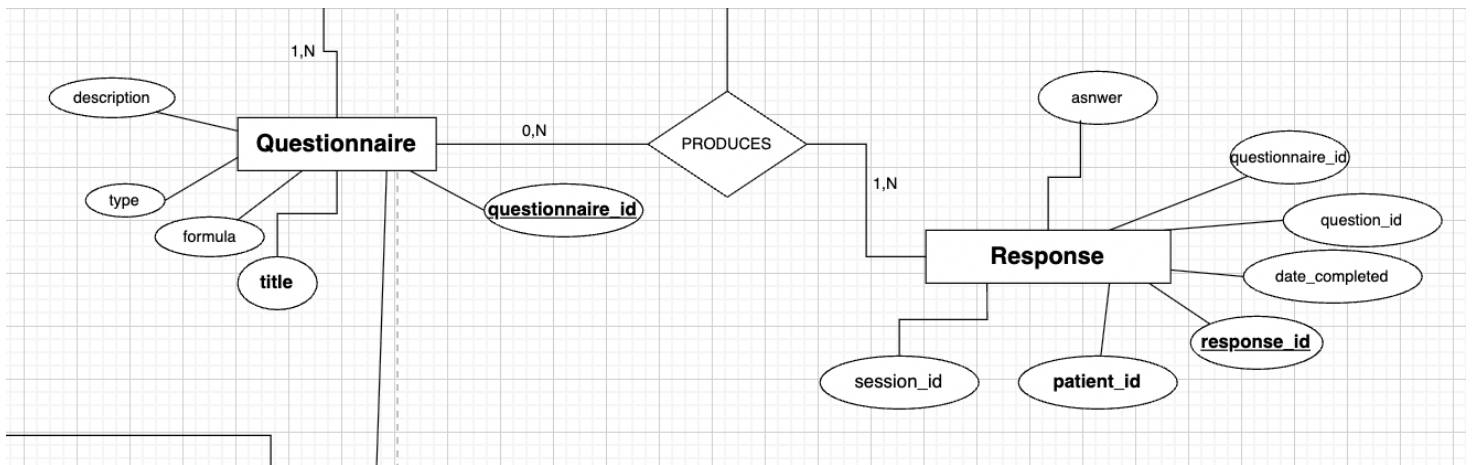
Questions and sections are their own separate entities, and they bind to the questionnaire by its id.

Then we have the ResponseType and ResponseTypeOption entities.



The ResponseType represents the many ways a user/patient can answer a question e.g. radio button, free text, number input etc. The ResponseTypeOption is every individual choice or index/setting of the response type. For example, multiple choice as a radio button is a response type, but each option of that, is the ResponseTypeOption .

Lastly, we have the Response entity.

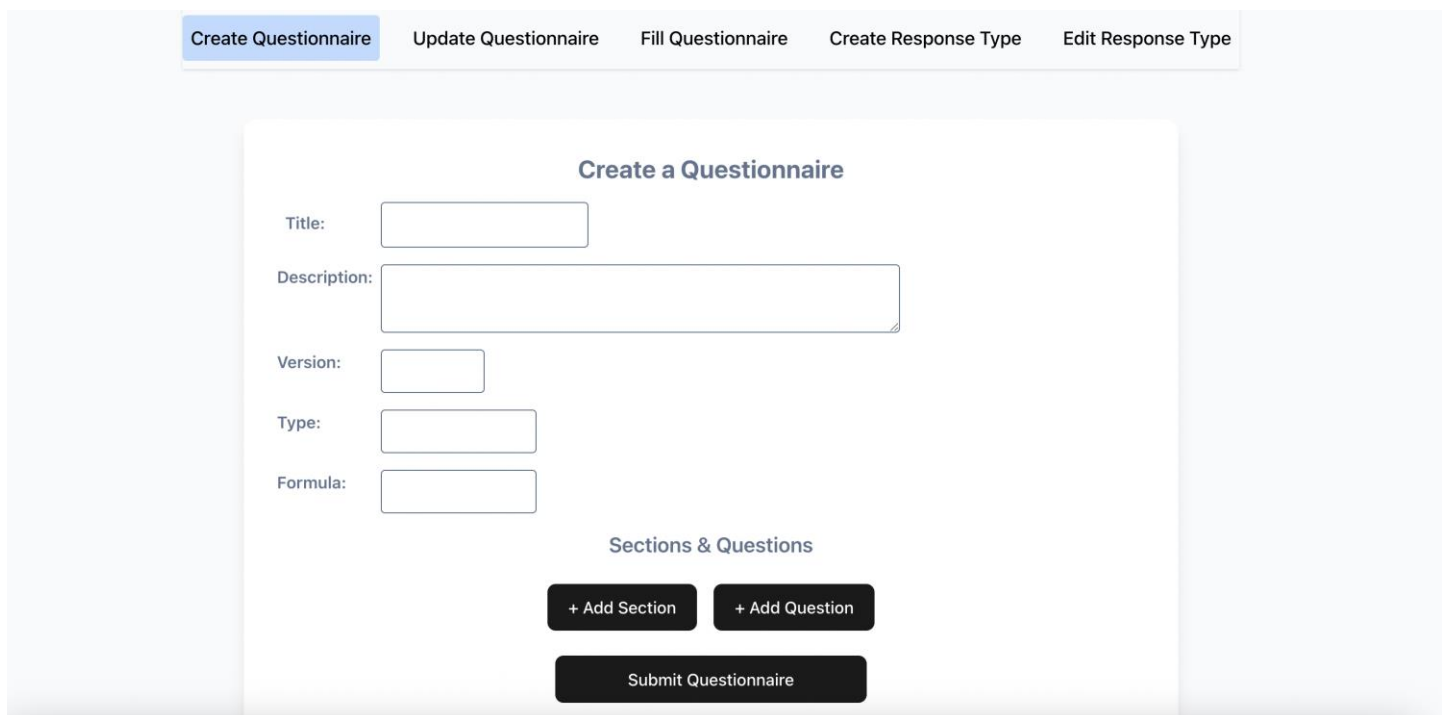


This is the final entity produced when submitting a questionnaire. Every single choice the patient made is stored as a response also bind to the questionnaire with a session_id and a date_completed to help identify the sum of responses of a questionnaire.

Demonstration

In this chapter, the whole process of creating, updating and submitting a questionnaire is going to be shown through screenshots.

Initial screen:



The screenshot displays the 'Create a Questionnaire' interface. At the top, a navigation bar contains five tabs: 'Create Questionnaire' (highlighted in blue), 'Update Questionnaire', 'Fill Questionnaire', 'Create Response Type', and 'Edit Response Type'. The main content area is titled 'Create a Questionnaire' and features several input fields: 'Title:' (a single-line text box), 'Description:' (a multi-line text area), 'Version:' (a single-line text box), 'Type:' (a single-line text box), and 'Formula:' (a single-line text box). Below these fields, the section 'Sections & Questions' is visible, containing two buttons: '+ Add Section' and '+ Add Question'. At the bottom of the form is a large 'Submit Questionnaire' button.

The doctor/user (the doctor will be referred to as 'user') is greeted with a simple UI with all his actions supported in the top bar.

Assuming he is creating a questionnaire:

Create a Questionnaire

Title:

EORTC QLQ-C30

Description:

Ενδιαφερόμαστε για ορισμένες πληροφορίες που αφορούν εσάς και την υγεία σας. Παρακαλούμε απαντήστε εσείς προσωπικά σε

Version:

3.0

Type:

physical

Formula:

$Q1-Q2+Q3*0,5$

Sections & Questions

Section 1:

Κατά τη διάρκεια της τελευταίας εβδομάδας:

Question 2:

Περιοριστήκατε στην εργασία σας ή σε άλλες καθημερινές ασχολίες σας;

Response Type

Satisfaction Scale ▾

Question 3:

Περιοριστήκατε στις ερασιτεχνικές σας ασχολίες ή σε άλλες δραστηριότητες 1

Response Type

checkbox ▾

+ Add Section

+ Add Question

Submit Questionnaire

After submitting the questionnaire and everything is saved in the database, we move on to the next screen which is the one responsible for updates/edits.

[Create Questionnaire](#)[Update Questionnaire](#)[Fill Questionnaire](#)[Create Response Type](#)[Edit Response Type](#)

Select a Questionnaire to Edit

Fetch Selected Questionnaire

All the questionnaires are listed for the doctor to choose.

Select a Questionnaire to Edit

Fetch Selected Questionnaire

Let us update the response type of the second question from checkbox to satisfaction scale.

Sections & Questions

Section 1:

Κατά τη διάρκεια της τελευταίας εβδομάδας:

Question 2:

Περιοριστήκατε στην εργασία σας ή σε άλλες καθημερινές ασχολίες σας;

Response Type

Satisfaction Scale

Question 3:

Περιοριστήκατε στις ερασιτεχνικές σας ασχολίες ή σε άλλες δραστηριότητες του ελεύθερου χρόνου;

Response Type

Satisfaction Scale

+ Add Section

+ Add Question

Update Questionnaire

And now proceed to fill the questionnaire on the next screen by choosing it like we did on the screen for the update :

Update Questionnaire

Fill Questionnaire

Create Response Type

Select a Questionnaire to Fill

serious-test

eortc-test-qcl

EORTC QLQ-C30

title-test

Fetch Selected Questionnaire

This is the fetched Questionnaire, and we selected the answers.

[Create Questionnaire](#)[Update Questionnaire](#)[Fill Questionnaire](#)[Create Response Type](#)[Edit Response Type](#)

EORTC QLQ-C30

(version 3)

Ενδιαφερόμαστε για ορισμένες πληροφορίες που αφορούν εσάς και την υγεία σας. Παρακαλούμε απαντήστε εσείς προσωπικά σε όλες τις ερωτήσεις, σημειώνοντας μέσα σε ένα κύκλο τον αριθμό που σας ταιριάζει καλύτερα. Δεν υπάρχουν «σωστές» και «λάθος» απαντήσεις. Οι πληροφορίες που θα δώσετε θα παραμείνουν αυστηρώς εμπιστευτικές.

Κατά τη διάρκεια της τελευταίας εβδομάδας:

2. Περιοριστήκατε στην εργασία σας ή σε άλλες καθημερινές ασχολίες σας; ☒ καθολου ☐ λίγο ☐ αρκετά ☐ πολυ
3. Περιοριστήκατε στις ερασιτεχνικές σας ασχολίες ή σε άλλες δραστηριότητες του ελεύθερού σας χρόνου; ☐ καθολου ☒ λίγο ☐ αρκετά ☐ πολυ

[Fill Questionnaire](#)

We press 'Fill Questionnaire' and we can see the responses in the database with their own session_id to bind them together.

	response_id	answer	session_id	questionnaire_id	question_id	date_completed	patient_id
1	43	καθολου	7d49485f-e57b-49d4-9a9f-7b94377...	14	15	2025-07-04 00:07:09.033990	
2	44	λίγο	7d49485f-e57b-49d4-9a9f-7b94377...	14	16	2025-07-04 00:07:09.033990	

Response types were also used when creating the questionnaire . Let us see how the doctor can create a response type.

Update Questionnaire

Fill Questionnaire

Create Response Type

Create Response Type

Response Type Name

Input Type

-- Select --

Create Response Type

Let's make an example of a radio button for a response. We write the name, select 'radio' as input type and add as many options as we want.

Create Response Type

Response Type Name

TestResponseType

Input Type

Radio

Options

none

little

somewhat

a lot

+ Add Option

Create Response Type

We then press the 'Create Response Type' button and the response type is ready for use. Similar way to the questionnaire, we can update/edit the response type by simply selecting it on the list.

Select a Response Type to Edit

checkbox

FreeText

num_one_to_ten

Satisfaction Scale

TestResponseType

Fetch Selected Response Type

And we can edit it as shown below. The only thing that is not editable is the input type as that would cause issues when transforming from one type to another.

Edit Response Type

Response Type Name

TestResponseType

Input Type (not editable)

radio

Options

none

little

somewhat

a lot

Update Response Type

Conclusion

This project presents the design and implementation of a full-stack web-based application aimed at supporting the digitization of medical questionnaire workflows. It facilitates dynamic questionnaire creation by healthcare professionals and structured response collection from patients.

The core functionality includes:

A responsive frontend interface (React.js) that allows medical practitioners to define questionnaire content—including sections, questions, and response types—with customizable input formats (e.g., multiple choice, free text, numeric ranges).

A robust backend (Spring Boot with PostgreSQL) that ensures data persistence, validation, and modular management of entities such as questionnaires, questions, sections, and response types.

A submission flow whereby patients interactively complete and submit questionnaires, with built-in validation ensuring completeness and data consistency.

The system architecture promotes scalability, code modularity, and ease of maintenance. By introducing reusable components like response types and formula definitions, the application reduces redundancy and encourages standardization in clinical assessments.

This platform has clear applicability in healthcare environments, particularly for streamlining patient intake, conducting standardized evaluations, and supporting data-driven decision-making. Furthermore, the modular approach and separation of concerns make it extensible for future features such as data analytics, role-based access control, or integration with hospital information systems (HIS).

From a pedagogical perspective, this project also demonstrates key software engineering principles including full-stack integration, RESTful API design, database normalization, and frontend-backend communication patterns. It serves as a practical example of applying modern web technologies to solve domain-specific problems in healthcare informatics.

Installation instructions

Requirements & Setup Instructions

This project consists of a React.js frontend and a Spring Boot backend using Java and PostgreSQL. Below are the necessary tools, dependencies, and configurations.

System Requirements

Component	Required Version	Notes
Java JDK	17+	Used to compile and run Spring Boot
Node.js	18+	Required for the React frontend
npm	Comes with Node.js	Package manager for React
PostgreSQL	13+	Relational database used
IDE (optional)	IntelliJ IDEA / VS Code	For development

Backend Setup (Spring Boot + PostgreSQL)

1. Install Java JDK
 - a. Ensure Java 17+ is installed -> `java -version`
2. Install PostgreSQL
 - a. Create a database
3. Configure application.properties
4. Run the Backend
 - a. In IntelliJ: Right-click the main class and run.
 - b. Or via terminal: `./mvnw spring-boot:run`

Frontend Setup (React + Vite)

1. Install Node.js & npm
 - a. Check installation:
 - i. `node -v`
 - ii. `npm -v`
2. Navigate to the frontend folder
3. Install dependencies
 - a. `npm install`
4. Run the development server
 - a. `npm run dev`
5. Frontend should be accessible at <http://localhost:5173>