Project Dental Clinic

October 2023

Project Report

1 Καθορισμός Θέματος-Project Definition

Το project που δημιουργήθηκε, αφορά ένα web application που ονομάζεται “ToothHub” και έχει σκοπό να διευκολύνει τους εργαζομένους μιας οδοντιατρικής κλινικής (dental clinic). Συγκεκριμένα, πρόκειται για ένα σύστημα-ατζέντα που βοηθά τους χρήστες της εφαρμογής να διατηρούν τον έλεγχο των καθημερινών ραντεβού τους και πελατών τους αλλά και τη διαχείριση των εργαζομένων.

Αρχικά, κάθε χρήστης (στην περίπτωση του project, οδοντίατρος ή γραμματέας οδοντιατρείου) συνδέεται στο σύστημα παρέχοντας ένα προκαθορισμένο* συνδυασμό username και password (που υπάρχει καταγεγραμμένο στη βάση δεδομένων) και πατάει το κουμπί Login. Εφόσον πιστοποιηθεί η εγκυρότητα των στοιχείων του στο σύστημα, μπορεί πλέον να συνδεθεί στον προσωπική του σελίδα η οποία διακρίνεται από ένα πλαϊνό μενού επιλογών. Οι επιλογές που του δίνονται είναι να προβάλει τους ασθενείς του (Show Patients), να προβάλει τα ραντεβού του (Show Appointments), να προβάλει και να αλλάξει τα προσωπικά στοιχεία του προφίλ του(Profile Settings) και να αποσυνδεθεί από το σύστημα (Log Out).

Στην περίπτωση που επιλέξει να προβάλει τους ασθενείς του (Show Patients), εμφανίζεται στην main οθόνη η λίστα με τους υπάρχοντες ασθενείς. Η λίστα αποτελείται από 5 στήλες, οι δυο πρώτες παρέχουν το όνομα και το επίθετο του ασθενή ενώ οι τελευταίες τρείς αποτελούν κουμπιά όπου του παρέχονται οι δυνατότητες να προβάλει περισσότερες πληροφορίες για τον ασθενή (View Details) ή για τυχόν προηγούμενα ραντεβού (ViewPreviousAppointments) με τον ασθενή, να επεξεργαστεί τις αποθηκευμένες πληροφορίες του ασθενή (Edit Details) και τέλος να διαγράψει έναν ασθενή από τη βάση δεδομένων της κλινικής (Delete). Πάνω από τη λίστα δίνεται και η δυνατότητα να προσθέσει έναν νέο ασθενή πατώντας το κουμπί (Add Patient). Για καθεμία από τις προ αναφέρουσες λειτουργίες (View, Edit, Delete, Add) ανακατευθύνεται σε νέα σελίδα ύστερα από επιλογή τους.

Στην περίπτωση που επιλέξει να προβάλει τα ραντεβού του (Show Appointments), εμφανίζεται στην main οθόνη η λίστα με τους υπάρχοντα ραντεβού του. Τα ραντεβού του είναι ταξινομημένα ξεκινώντας από την παλιότερη προς τη νεότερη ημερομηνία, και είναι επίσης δευτερευόντως ταξινομημένα ξεκινώντας από τα πρωινά προς τα απογευματινά ραντεβού. Ωστόσο, αν θέλει, υπάρχει η δυνατότητα (και ενδείκνυται ως προς χρήση) να επιλέξει με βάση το κουμπί του ημερολογίου μια συγκεκριμένη ημερομηνία και να προβάλει τα ραντεβού μιας συγκεκριμένης ημερομηνίας μόνο. Στη συνέχεια, όμοια με την περίπτωση προβολής ασθενούς, η λίστα περιλαμβάνει 5 στήλες με τις 2 πρώτες να περιέχουν την ημερομηνία και την ώρα του ραντεβού, ενώ οι τελευταίες τρείς αποτελούν κουμπιά όπου του παρέχονται οι δυνατότητες να προβάλει περισσότερες πληροφορίες για το ραντεβού (View Details) , να επεξεργαστεί τις αποθηκευμένες πληροφορίες του ραντεβού (Edit Details) και τέλος να διαγράψει ένα ραντεβού από τη βάση δεδομένων της κλινικής (Delete). Πάνω από τη λίστα δίνεται και η δυνατότητα να προσθέσει έναν νέο ραντεβού πατώντας το κουμπί (Add Appointment). Για καθεμία από τις προ

αναφέρουσες λειτουργίες (View, Edit, Delete, Add) ανακατευθύνεται σε νέα σελίδα ύστερα από επιλογή τους.

Στην περίπτωση που επιλέξει να προβάλει και να αλλάξει τα προσωπικά στοιχεία του προφίλ του (Profile Settings), εμφανίζεται στην κύρια οθόνη μια κάρτα με τα υπάρχοντα προσωπικά στοιχεία του οδοντιάτρου, *ωστόσο δεν εμφανίζονται το username και το password*.

* Όπως αναφέρθηκε και προηγουμένως, για να συνδεθεί ένας οδοντίατρος και να επεξεργαστεί τα ραντεβού του και τους ασθενείς του καλείται να εισάγει προκαθορισμένο username και password. Ο προκαθορισμός των στοιχείων θεωρούμε ότι γίνεται ύστερα από συνεννόηση του οδοντίατρου (εργαζομένου) με τον διαχειριστή της κλινικής (εργοδότης). Για αυτό το σκοπό, δίνεται η δυνατότητα κατά την οθόνη εισόδου του οδοντιάτρου , να επιλεχθεί η επιλογή (σύνδεσμος) “Create a new dentist account ? (Login as administrator)”. Αν επιλεχθεί μπορεί ο διαχειριστής να συνδεθεί εισάγοντας το username του και το password του, ενώ για επιβεβαίωση ότι βρίσκεται σε mode διαχειριστή, εμφανίζεται η υπενθύμιση με κόκκινα έντονα γράμματα “Administration Login”. Αφού συνδεθεί επιτυχώς, ανακατευθύνεται σε νέα σελίδα όπου του παρέχεται μια λίστα με τους εργαζομένους-οδοντιάτρους. Η λίστα παρέχει στήλες όνομα, επίθετο, δυνατότητα προβολής/επεξεργασίας εργαζομένου (View/Edit), δυνατότητας διαγραφής(Delete). Επίσης πάνω από τη λίστα παρέχεται η δυνατότητα πρόσθεσης ενός οδοντιάτρου στην κλινική (Add Dentist), και αποσύνδεσης από το λογαριασμό Log Out. Επιλέγοντας, (Add Dentist), ανακατευθύνεται στη σελίδα εισαγωγής στοιχείων του νέου οδοντιάτρου και εκεί εισάγει και το username και το password που συμφωνήθηκε.

2 Ανάλυση Απαιτήσεων – Use Cases

Στην παρούσα ενότητα παρατείνονται οι περιγραφές των use cases με βάση τις καταγεγραμμένες απαιτήσεις που εξήχθησαν από τον καθορισμό θέματος.

2.1 LoginAdministrator

Use case ID	01
Description and Goal	The use case “LoginAdministrator” connects Administrator in the Web Application.
Actors	Administrator Of Dental Clinic
Preconditions	1.The administrator knows his username and the password. 2.The administrator clicks on the link “Create a new dentist account ? (Login as administrator)” on the homepage of WebApp.
Main flow of events	1. The use case starts when the administrator clicks on the link “Create a new dentist account ? (Login as administrator)” on the homepage of WebApp. 2. The system shows the hint “Administration Login” 3. The administrator enters his username and his password and clicks on button Login.
Extension/Variations	---
Post conditions	The system of Web App shows the admin page.

2.2 AddDentist

Use case ID	02
Description and Goal	The use case “AddDentist” adds a Dentist entity in the table “dentist” of the database of the Web App.
Actors	Administrator Of Dental Clinic
Preconditions	Administrator has successfully logged In the system.

Main flow of events	1. The use case starts when the administrator clicks on the button “Add Dentist”. 2. The Web App redirects to a new empty form with title “New Dentist Form”. 3. The Administrator complete the necessary fields and clicks on button “Save”.
Extension/Variations	3.In the case, that administrator wants to exit from the page of dentist form can click on “cancel”.
Post conditions	The new dentist profile is saved into the table dentist of the database of the system.

2.3 EditDentist

Use case ID	03
Description and Goal	The use case “EditDentist” edits an existing entity Dentist in the table “dentist” of the database of the Web App.
Actors	Administrator Of Dental Clinic
Preconditions	1.Administrator has successfully logged In the system. 2. There are available dentists in the dentist list.
Main flow of events	1. The use case starts when the administrator clicks on the button “Edit” in in dentist list, in the row of the dentist he wants to edit. 2. The Web App redirects to a form with some, or all completed input fields of the selected dentist. 3. The Administrator complete the necessary fields and clicks on button “Save”.
Extension/Variations	2. The password field isn’t visible unless he clicks on the button “Show Encrypted Password”. 3.In the case, that administrator wants to exit from the page of dentist form can click on “cancel”.
Post conditions	The updated dentist profile is saved into the table dentist of the database of the system.

2.4 DeleteDentist

Use case ID	04
Description and Goal	The use case “DeleteDentist” deletes an existing entity Dentist in the table “dentist” of the database of the Web App.
Actors	Administrator Of Dental Clinic
Preconditions	1.Administrator has successfully logged In the system. 2. There are available dentists in the dentist list.
Main flow of events	1. The use case starts when the administrator clicks on the button “Delete” in in dentist list, in the row of the dentist he wants to edit. 2. The Web App pops up a new window confirmation and asks for confirmation. 3. The Administrator clicks on the “ok” button of the window.
Extension/Variations	3.In the case, that administrator wants to cancel the deletion of dentist clicks on “cancel”.
Post conditions	The selected dentist entity is deleted from the database, but first any dependencies with patient and appointment tables are set to null.

2.5 LogoutAdministrator

Use case ID	05
Description and Goal	The use case “LogoutAdministrator” disconnects the administrator from the system.
Actors	Administrator Of Dental Clinic
Preconditions	1.Administrator has successfully logged In the system.
Main flow of events	1. The use case starts when the administrator clicks on the button “Log Out” in the adminpage. 2. The Web App redirects the administrator to the homepage of the webApp.
Extension/Variations	---

Post conditions	---

2.6 LoginDentist

Use case ID	06
Description and Goal	The use case “LoginDentist” connects the dentist into the system.
Actors	Dentist (or Secretary)
Preconditions	Dentist has and knows predetermined username and password
Main flow of events	1. The use case starts when the dentist inserts the username and password and clicks on “Login” button. 2. The Web App redirects the dentist to the personalpage of the webApp.
Extension/Variations	1. In the case that the dentist inserts a wrong username, a message appears containing the message “username not exists”. 2. In the case that the dentist inserts a wrong password and a correct username, a message appears containing the message “password Not match”.
Post conditions	The system of Web App shows the dentist page.

2.7 ShowPatients

Use case ID	07
Description and Goal	The use case “ShowPatients” shows patients’ list .
Actors	Dentist (or Secretary)
Preconditions	Dentist has and knows predetermined username and password to connect to his account.
Main flow of events	1. The use case starts when the dentist clicks on the option Show Patients on the sidebar. 2. The Web App responds by showing the patients list (available from the database) in the same and main window.
Extension/Variations	1. In the case that there are no available patients, the database shows an empty list.
Post conditions	The dentistpage displays now the sidebar menu and the patient’s list.

2.8 ViewPatient (& viewPreviousAppointments: extension)

Use case ID	08
Description and Goal	The use case “ViewPatient” displays the details of a selected patient.
Actors	Dentist (or Secretary)
Preconditions	Dentist has and knows predetermined username and password in order to connect to his account. Dentist has previously selected the option Show Patients in the sidebar menu.
Main flow of events	1. The use case starts when the dentist clicks on the button “View” on the on the selected patient row. 2. The Web App redirects the dentist into a new page viewProfile, where patient’s details are displayed.
Extension/Variations	1. In the viewProfile page, the dentist can click on link “Previous Appointments” and see the previous Appointments of the dentist.
Post conditions	The dentistpage displays the details of the selected patient profile.

2.9 EditPatient

Use case ID	09
Description and Goal	The use case “EditPatient” edits the details of an existing patient.
Actors	Dentist (or Secretary)
Preconditions	Dentist has and knows predetermined username and password in order to connect to his account. Dentist has previously selected the option Show Patients in the sidebar menu.
Main flow of events	1. The use case starts when the dentist clicks on the button “Edit” on the on the selected patient row. 2. The Web App redirects the dentist into a new page editProfile, where existing patient’s details are displayed. 3. The dentist edits the details that he wants and clicks on “Submit” button
Extension/Variations	1. In the editProfile page, the dentist can click on “Cancel” button.
Post conditions	The edited patient profile is saved in the patient table into the database.

2.10 DeletePatient

Use case ID	10
Description and Goal	The use case “DeletePatient” deletes an existing patient.
Actors	Dentist (or Secretary)
Preconditions	Dentist has and knows predetermined username and password in order to connect to his account. Dentist has previously selected the option Show Patients in the sidebar menu.
Main flow of events	1. The use case starts when the dentist clicks on the button “Delete” on the on the selected patient row. 2.The web App pops up a new window and asks for confirmation. 3. The dentist clicks on “Ok” button.
Extension/Variations	1. The dentist may click on “Cancel” button at confirmation window and cancel the operation.
Post conditions	The selected patient profile is deleted from the patient table in the database.

2.11 AddPatient

Use case ID	11
Description and Goal	The use case “AddPatient” adds a new Patient into the database (table patient).
Actors	Dentist (or Secretary)
Preconditions	Dentist has and knows predetermined username and password in order to connect to his account. Dentist has previously selected the option Show Patients in the sidebar menu.
Main flow of events	1. The use case starts when the dentist clicks on the button “Add Patient”. 2. The web App redirects the dentist into a new page where an empty form is displayed. 3. The dentist inserts the new patient information into the fields and clicks button “Save”.

	4. The webApp redirects the dentist in a result page that displays the message “The patient with name {input name} and last name {input surname} was successfully registered in the system!”
Extension/Variations	---
Post conditions	The new patient profile is added into the patient table in the database.

2.12 ShowAppointments

Use case ID	12
Description and Goal	The use case “ShowAppointments” shows appointments’ list .
Actors	Dentist (or Secretary)
Preconditions	Dentist has and knows predetermined username and password to connect to his account.
Main flow of events	1. The use case starts when the dentist clicks on the option Show Patients on the sidebar. 2. The Web App responds by showing the appointments list (available from the database) in the same and main window sorted by date and time(ascending order).
Extension/Variations	1. In the case that there are no registered appointments, the database shows an empty list. 2. In the case that the dentist selects a specific date, appointments for the selected date are displayed
Post conditions	The dentistpage displays now the sidebar menu and the appointments list.

2.13 ViewAppointments

Use case ID	13
Description and Goal	The use case “ViewAppointments” displays the details of a selected appointment.
Actors	Dentist (or Secretary)

Preconditions	Dentist has and knows predetermined username and password in order to connect to his account. Dentist has previously selected the option Show Appointments in the sidebar menu.
Main flow of events	1. The use case starts when the dentist clicks on the button “View” on the on the selected appointment row. 2. The Web App redirects the dentist into a new page viewAppointment, where appointment’s details are displayed.
Extension/Variations	---
Post conditions	The dentistpage displays the details of the selected appointment.

2.14 EditAppointment

Use case ID	14
Description and Goal	The use case “EditAppointment” edits the details of an existing appointment.
Actors	Dentist (or Secretary)
Preconditions	Dentist has and knows predetermined username and password in order to connect to his account. Dentist has previously selected the option Show Appointments in the sidebar menu.
Main flow of events	1. The use case starts when the dentist clicks on the button “Edit” on the on the selected appointment row. 2. The Web App redirects the dentist into a new page editAppointment, where existing appointment’s details are displayed. 3. The dentist edits the details that he wants and clicks on “Submit” button
Extension/Variations	1. In the editAppointment page, the dentist can click on “Cancel” button and cancel the edit operation.
Post conditions	The edited appointment is saved in the appointment table into the database.

2.15 DeleteAppointment

Use case ID	15
Description and Goal	The use case “DeleteAppointment” deletes an existing appointment.
Actors	Dentist (or Secretary)
Preconditions	Dentist has and knows predetermined username and password in order to connect to his account. Dentist has previously selected the option Show Appointments in the sidebar menu.
Main flow of events	1. The use case starts when the dentist clicks on the button “Delete” on the on the selected appointment row. 2.The web App pops up a new window and asks for confirmation. 3. The dentist clicks on “Ok” button.
Extension/Variations	1. The dentist may click on “Cancel” button at confirmation window and cancel the operation.
Post conditions	The selected appointment is deleted from the appointment table in the database.

2.16 AddAppointment

Use case ID	16
Description and Goal	The use case “AddAppointment” adds a new Appointment into the database (table appointment).
Actors	Dentist (or Secretary)
Preconditions	Dentist has and knows predetermined username and password in order to connect to his account. Dentist has previously selected the option Show Appointments in the sidebar menu.
Main flow of events	1. The use case starts when the dentist clicks on the button “Add Appointment”. 2. The web App redirects the dentist into a new page where an empty form is displayed. 3. The dentist inserts the new appointment information into the fields and clicks button “Save”.

	4. The webApp redirects the dentist in a result page that displays the message “The Appointment with date {input date} and time {input time} was successfully registered in the system!”
Extension/Variations	---
Post conditions	The new appointment is added into the appointment table in the database.

2.17 EditProfileSettings

Use case ID	17
Description and Goal	The use case “EditProfileSettings” edits dentist’s personal details.
Actors	Dentist (or Secretary)
Preconditions	Dentist has and knows predetermined username and password in order to connect to his account.
Main flow of events	1. The use case starts when the dentist clicks on the option “Profile Settings” in the sidebar menu. 2. The web App redirects the dentist into a new page where a form with dentist existing details is displayed. 3. The dentist edits the information that he wants and clicks button “Save”. 4. The webApp redirects the dentist into his dentist page.
Extension/Variations	1. The dentist may clicks on “Cancel” button to cancel the editing operation.
Post conditions	The updated dentist information is saved into the dentist table in the database.

2.18 LogoutDentist

Use case ID	18
Description and Goal	The use case “LogoutDentist” disconnects the dentist from the system.
Actors	Dentist (or secretary)
Preconditions	Dentist has and knows predetermined username and password in order to connect to his account.
Main flow of events	1. The use case starts when the dentist clicks on the option “Log Out” in the in the sidebar menu.

	2. The Web App redirects the dentist to the homepage of the webApp.
Extension/Variations	---
Post conditions	---

3 Σχεδίαση Ελέγχων – Test Cases

Για την σχεδίαση ελέγχων χρησιμοποιήθηκε το pattern Arrange-Act-Assert. Συνολικά πραγματοποιήθηκαν 51 τεστ με τη χρήση της βιβλιοθήκης Junit, Mockito και MockMvc. Πραγματοποιήθηκαν τεστ για τα βασικά πακέτα του backend, για τα πακέτα controllers, services, models, repositories.

Ενδεικτικά παρουσιάζονται κάποια από τα τεστ που έγιναν για κάθε πακέτο (2 για κάθε πακέτο).

3.1 AdministratorController_login_ReturnsOKStatus()

Arrange	1.Define the URL for the login endpoint: /administrator/login 2. Mock the behavior of passwordEncoder.matches to return true when provided with valid credentials. 3. Create a RequestBuilder for a POST request to the login endpoint with valid username and password parameters. 4. Set the content type of the request to MediaType.APPLICATION_JSON.
Act	1. Perform the HTTP POST request using the MockMvc framework. 2. Capture the MvcResult to inspect the response.
Assert	1. Get the HTTP response from the MvcResult. 2 . Extract the content of the response as a JSON string. 3. Define the expected success message as "Successful Login". 4. Verify the following: • The HTTP status code in the response is equal to HttpStatus.OK.value() (i.e., 200). • The content of the response matches the expected success message.
Test Result:	1. The test has passed successfully. 2. The AdministratorController login endpoint returns an HTTP OK (200) status when valid login credentials are provided.

	3. The response content matches the expected success message, confirming a successful login.

3.2 PatientController_getPatientBySsn_ReturnsOkObject

Arrange	1.Create a mock Patient object with sample data. 2.Mock the behavior of the patientService.findBySsn method to return the mock patient when provided with any string argument. 3.Define the URI for the GET request to retrieve a patient by SSN, e.g., /patients/getPatient/12345. 4.Create a RequestBuilder for a GET request with the specified URI and accept JSON as the response format.
Act	1. Perform the HTTP GET request using the MockMvc framework. 2. Capture the MvcResult to inspect the response.
Assert	1. Serialize the mock Patient object into JSON format using the objectMapper. 2. Extract the content of the response as a JSON string. 3. Compare the obtained JSON response with the expected JSON representation of the mock patient. 4. Verify that the response content matches the expected JSON.
Test Result:	1. The test has passed successfully. 2. The PatientController is capable of retrieving a patient object by SSN and returning it in JSON format. 3. The JSON response matches the expected JSON representation of the mock patient.

3.3 AppointmentService_getAppointmentsBySsn_ReturnsCorrectListOfAppointments

Arrange	1. Create a mock Patient object with a sample SSN. 2. Create a list of mock Appointment objects and associate them with the mock patient.

	3. Mock the behavior appointmentRepository.getAppointmentsByPatient_Ssn method to return the list of mock appointments when provided with the patient's SSN.
Act	1. Invoke the getAppointmentsBySsn method of the AppointmentService with the mock patient's SSN. 2. Capture the list of actual appointments returned by the service.
Assert	1. Ensure that the actual list of appointments is not null. 2. Compare the size of the actual list of appointments with the size of the expected list of appointments. 3. Verify that the actual list of appointments matches the expected list of appointments in terms of content and count.
Test Result:	1. The test has passed successfully. 2. The AppointmentService is capable of retrieving a list of appointments for a given patient's SSN. 3. The actual list of appointments matches the expected list of appointments in terms of content and count.

3.4 DentistService_deleteDentist_ReturnsNothing

Arrange	1. Create a mock Dentist object with a sample ID, name, and surname. 2. Mock the behavior of the dentistRepository.findByDentistId method to return the mock dentist when provided with the specified ID. 3. Configure the dentistRepository.delete method to do nothing (i.e., not actually delete the dentist but simulate the behavior).
Act	1. Invoke the deleteDentist method of the DentistService with the specified ID. 2. Capture the interactions with the dentistRepository.
Assert	1. Verify that the dentistRepository.delete method is called exactly once with the mock dentist as its argument.
Test Result:	1. The test has passed successfully. 2. The DentistService can correctly delete a dentist. 3. The delete method of the DentistRepository is called exactly once with the correct dentist object.

3.5 AppointmentManager_UpdateAppointment_ReturnsCorrectAppointmentObject

Arrange	1. Create a sample Appointment object with initial data, including ID, date, time, duration, and therapy.
Act	1. Invoke the updateAppointment method of the AppointmentManager with the sample appointmentData. 2. Capture the updated Appointment object returned by the updateAppointment method.
Assert	1. Verify that the updatedAppointment is not null, indicating a successful update. 2. Check that the attributes of the updatedAppointment match the corresponding attributes of the appointmentData: • ID matches. • Date matches. • Time matches. • Duration matches. • Therapy matches.
Test Result:	1. The test has passed successfully. 2. The AppointmentManager correctly updates the Appointment object with the provided data. 3. The updatedAppointment reflects the updated values for ID, date, time, duration, and therapy.

3.6 DentistManager_updateDentist_returnsCorrectlyUpdatedObject

Arrange	1. Create an instance of Dentist representing the old dentist with initial information. 2. Create another instance of Dentist representing the new dentist information. 3. Populate the new Dentist instance with updated values for name, surname, username, email, telephone, and address.
Act	1. Invoke the updateDentist method of the DentistManager with the old Dentist and new Dentist instances. 2. Capture the updated Dentist object returned by the updateDentist method.
Assert	1. Check that the updatedDentist object has the following updated attributes: 2. Name matches the new name.

	3. Surname matches the new surname. 4. Username matches the new username. 5. Email matches the new email. 6. Telephone matches the new telephone number. 7. Address matches the new address
Test Result:	1. The test has passed successfully. 2. The DentistManager correctly updates the Dentist object with the provided new information. 3. The updatedDentist object reflects the updated values for name, surname, username, email, telephone, and address.

3.7 PatientRepository_FindByDentistId_ReturnCorrectedPatientLists()

Arrange	1. Create a new instance of a Dentist and save it using the dentistRepository.save method. 2. Create two new instances of Patient associated with the same dentist. 3. Set unique SSNs for each patient. 4. Save both patients using the patientRepository.save method
Act	1. Invoke the findByDentist_DentistId method of the PatientRepository with the dentist's ID. 2. Capture the list of found patients.
Assert	1. Verify that the foundPatients list is not null. 2. Ensure that the foundPatients list contains exactly 2 patients, as expected.
Test Result:	1. The test has passed successfully. 2. The PatientRepository correctly retrieves a list of patients associated with the specified dentist using the findByDentist_DentistId method. 3. The foundPatients list contains exactly 2 patients, matching the expected behavior.

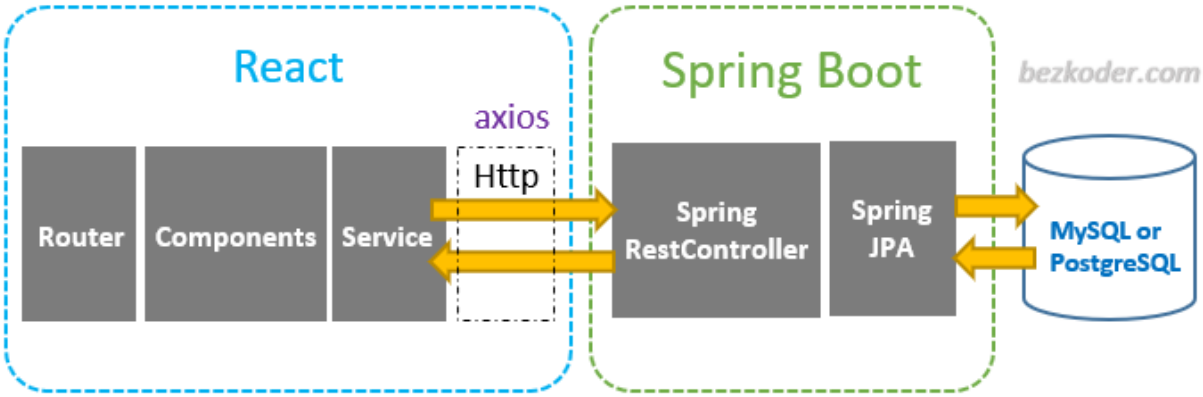
3.8 AppointmentRepository_FindByDateAndTime_ReturnCorrectAppointment()

Arrange	1. Create a Date object representing the desired date (September 30, 2023) and a time string ("10:00 AM"). 2. Create an Appointment object with the expected date and time.

	3. Save the expectedAppointment using the appointmentRepository.save method.
Act	1. Invoke the findByDateAndTime method of the AppointmentRepository with the final date and time. 2. Capture the found appointment.
Assert	1. Verify that the foundAppointment is not null, indicating that an appointment was found. 2. Ensure that the date and time of the foundAppointment match the expected date and time.
Test Result:	1. The test has passed successfully. 2. The AppointmentRepository correctly retrieves an appointment by date and time using the findByDateAndTime method. 3. The foundAppointment has the expected date and time.

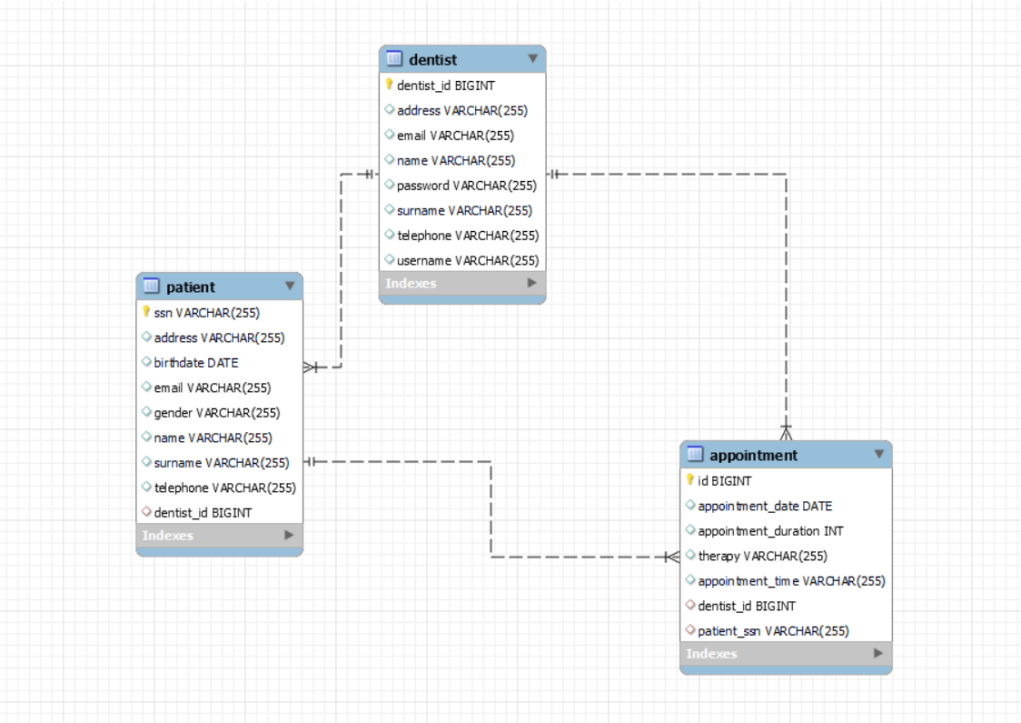
4 Σχεδίαση Λογισμικού– Software Architecture

Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκε για το frontend: javascript framework της **React** ενώ για το backend: **Spring Boot**. Η βάση δεδομένων που χρησιμοποιείται είναι η **mySQL**. Ενώ η σύνδεση του frontend με το backend γίνεται μέσω του javascript framework **axios**.

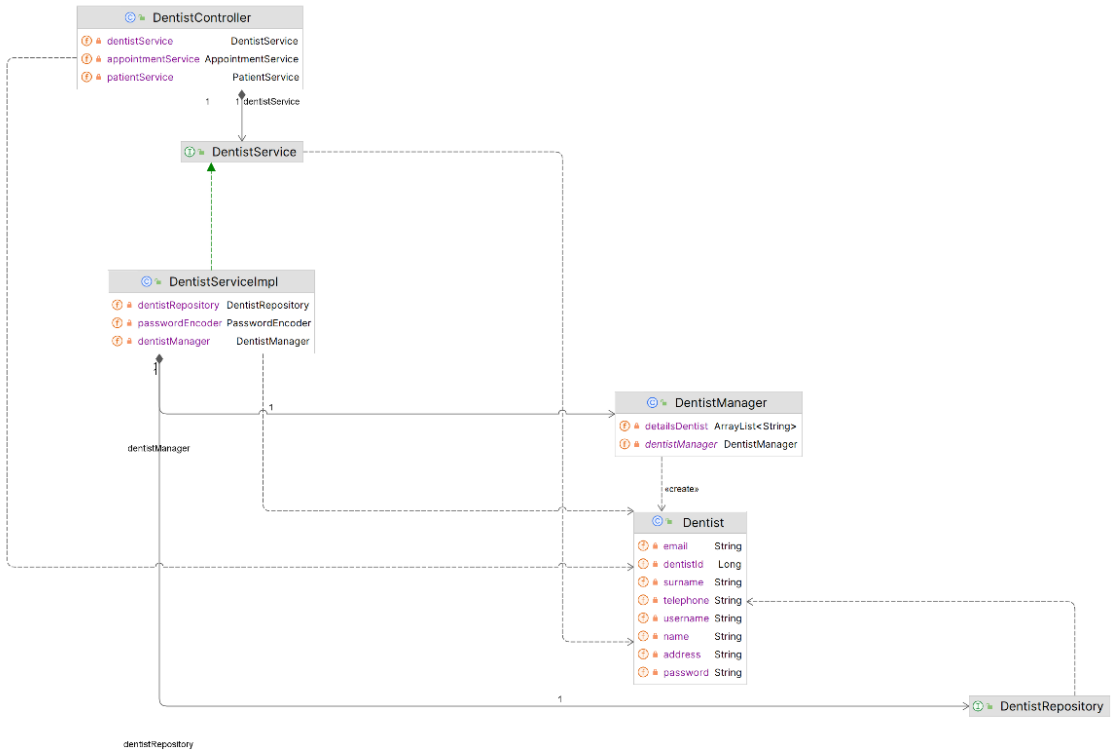


Εικόνα 1: Αρχιτεκτονική της web app εφαρμογής

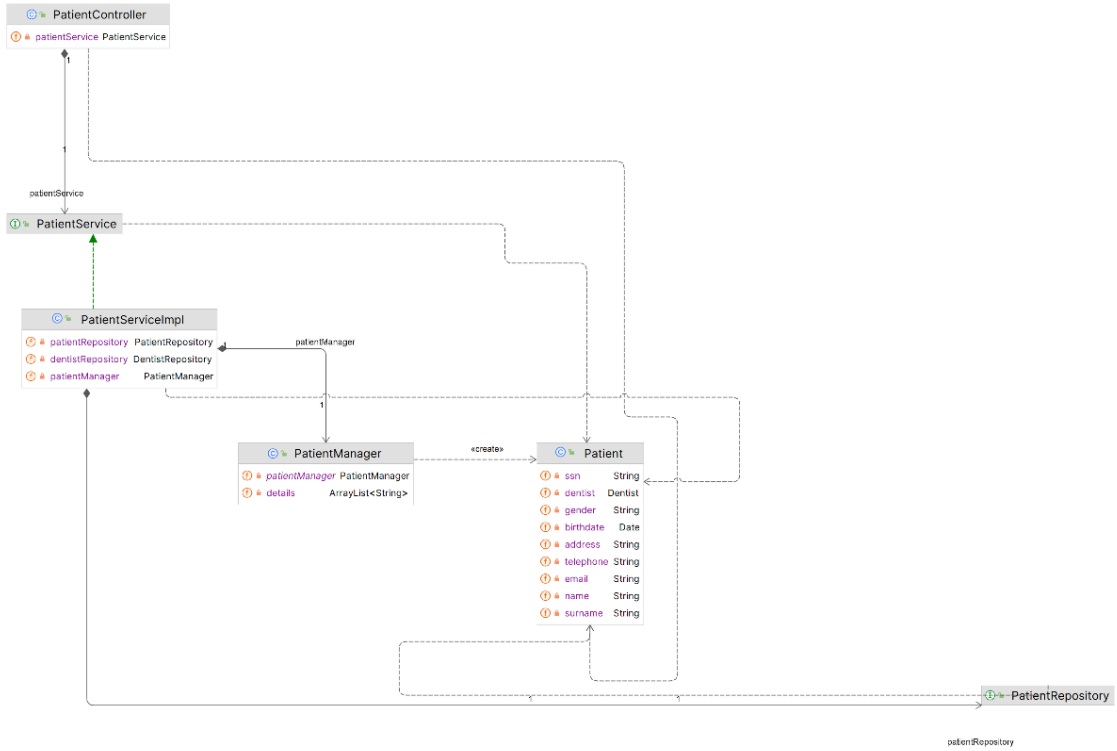
Παρακάτω παρουσιάζονται οι συνδέσεις μεταξύ των βασικών κλάσεων για τις 3 οντότητες του γραφήματος καθώς και το γράφημα σχήματος της βάσης.



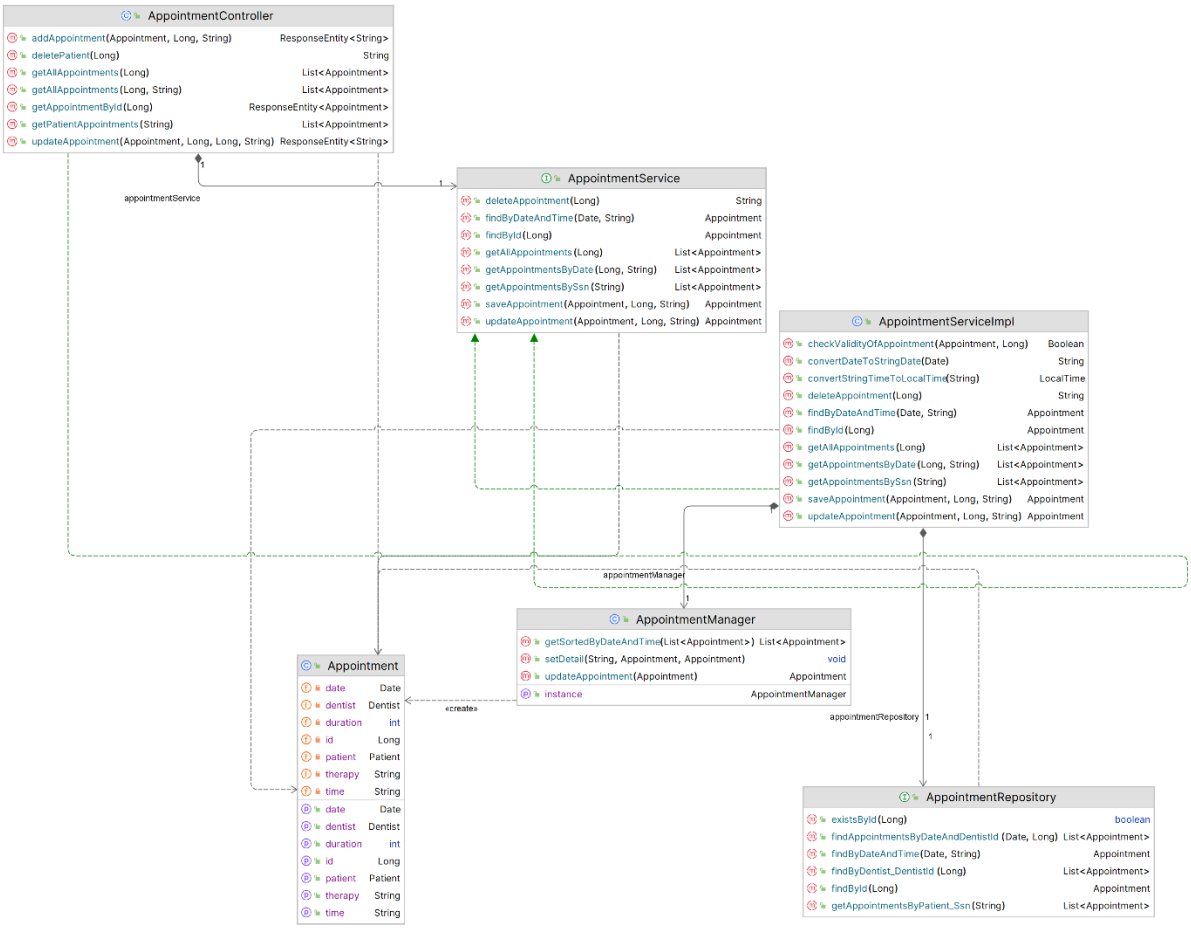
Εικόνα 2: Γράφημα σχήματος βάσης (MySQL).



Εικόνα 3: UML διάγραμμα για τις υλοποιήσεις των βασικών λειτουργίων της οντότητας Dentist



Εικόνα 4: UML διάγραμμα για τις υλοποιήσεις των βασικών λειτουργιών της οντότητας Patient.



Εικόνα 5: : UML διάγραμμα για τις υλοποιήσεις των βασικών λειτουργιών της οντότητας Appointment.