**Лабораторна робота 26-27.**

**Реалізація SPA UI із використанням фреймворку Angular.**

**Мета роботи:** навчитись встановлювати Angular, використовуючи Angular CLI, створювати та використовувати компоненти та сервіси, виконувати http-запити, передавати параметри та обробляти події.

**Задачі:**

* Налаштувати середовище розробки та встановити фреймворк використовуючи angular cli
* Змінити зовнішній вигляд основної сторінки да додати вкладений компонент
* Реалізувати сторінку студентської відомості
* Створити сервіс для забезпечення компонент даними
* Реалізувати роботу із back-end rest-api із використанням модуля http
* Виконати індивідуальне завдання

1. Angular представляє фреймворк від компанії Google для створення клієнтських додатків. Перш за все він націлений на розробку SPA-рішень (Single Page Application), тобто одно сторінкових застосунків. Angular надає таку функціональність, як двостороннє зв'язування, що дозволяє динамічно змінювати дані в одному місці інтерфейсу при зміні даних моделі в іншому, шаблони, маршрутизація та багато іншого. Однією з ключових особливостей Angular є те, що він використовує в якості мови програмування TypeScript. Ми не обмежені мовою TypeScript. При бажанні можемо писати програми на Angular за допомогою таких мов як Dart або JavaScript. Однак TypeScript все таки є основною мовою для Angular.

Для роботи з Angular необхідно встановити сервер Node.js (https://nodejs.org) і пакетний менеджер npm, що може бути встановлено із використанням інтерфейсу командного рядку (https://docs.npmjs.com/cli/install).

Хороші інструменти для розробки роблять роботу більш комфортною. Angular CLI - це інтерфейс командного рядка, який дозволяє вам швидко створювати проекти, додавати файли та багато іншого. Встановлювати Angular CLI бажано глобально. Щоб встановити CLI за допомогою npm, можна відкрити вікно термінала / консолі та введіть команду «npm install -g @angular/cli».

1. Далі із використанням Angular CLI створення нового застосунку виглядає досить легко. Достатньо виконати команду «ng new my-app», де my-app – ім’я нашого проекту. Створимо новий проект «exam-client» (Рис.1).



Рис.1. Створення проекту «exam-client».

Angular включає в себе сервер, так що ми можемо без використання веб-серверу та компіляції typescript тестувати роботу своєї програми локально. Після завершення інсталяції фреймворку зайдемо у папку проекту «exam-client» на наберемо у командному рядку «ng serve» (рис.2)

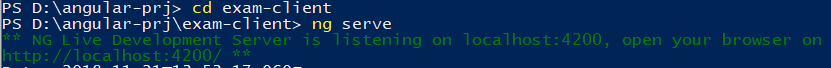


Рис.2. Запуск сервера Angular.

Переглянути вміст сторінки нашого застосунку можна за посиланням «localhost:4200» (рис.3)

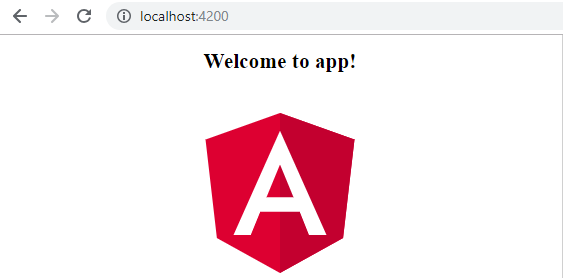


Рис.3. Зовнішній вигляд сторінки «exam-client»

Виконаємо невеликі зміни. Відкриємо файл шаблону корньового компонента «src\app\app.component.html» та змінимо його наступним чином (рис.4).

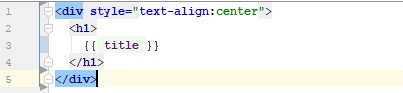


Рис.4. Зміни у app.component.html.

У класі компонента у файлі «src\app\app.component.ts» запишимо у змінну title «Екзаменаційна відомість» (рис.5)

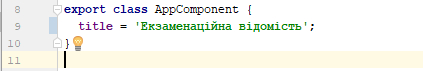


Рис.5. Зміни у app.component.ts

Сторінка тепер матиме наступний вигляд (рис.6)

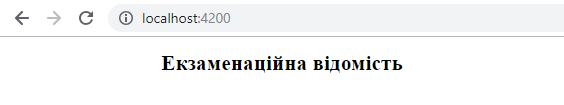


Рис.6. Вигляд сторінки після змін.

1. Для спрощення розробки зручного користувацького інтерфейсу, встановимо та підключимо до нашого проекту jquery та bootstrap. Спочатку виконаємо установку (рис.7)







Рис.7. Установка bootstrap, jquery та popper.

Відрегагуємо .angular-cli.json для включення встановлених стилів та скриптів (рис.8).

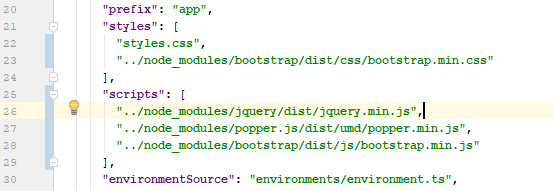


Рис.8. Редагування .angular-cli.json

Додамо компонент, що відображатиме загальні дані щодо відомості – дисципліну, викладача та дату іспиту. Для цього в командному рядку виконаємо «ng generate component» (рис.9).

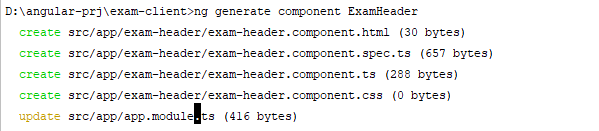
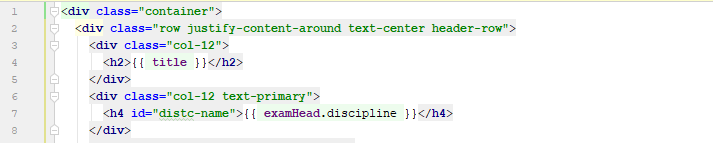
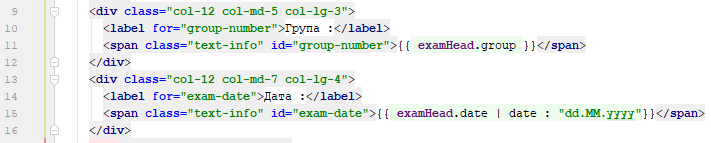


Рис.9. Створення нового компонента.

У директорії «src\app» створюється нова папка «exam-header». Відредагуємо html-шаблон (рис.10) та код класу (рис.11)





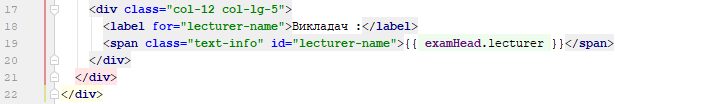
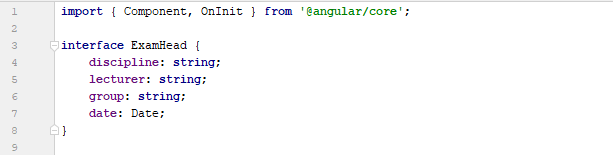


Рис.10. exam-header.component.html.





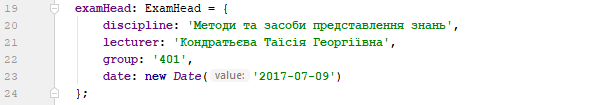




Рис.11. exam-header.component.spec.ts.

Підключемо новий компонент у корньовому компоненті app у файлі «src\app\app.component.html» (рис.12) та переглянемо результат (рис.13)



Рис.12. Модифікація app.component.html.

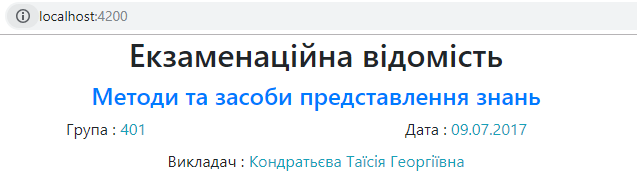


Рис.13. Зовнішній вигляд сторінки.

1. Реалізуємо сервіс, що буде забезпечувати наші компоненти даними. Виконаємо команду «ng generate service» (рис.14)



Рис.14. Генерація сервісу ExamData.

Також згенеруємо інтерфейс ExamHead (рис.15)



Рис.15. Генерація інтерфейсу.

До файлу «exam-head.ts» перенесемо оголошення інтерфейсу з «exam-header.component.spec.ts» (рис.16)

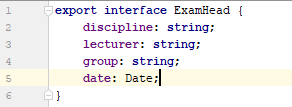
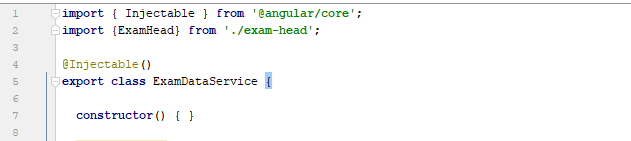


Рис.16. Інтерфейс ExamHead

У файлі «exam-data.service.ts» реалізуємо метод, що повертатиме дані шапки відомості (рис.17)



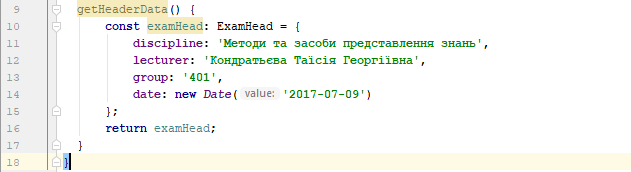


Рис.17. Сервіс exam-data.service.ts

У модулі «app.module.ts» включимо наш сервіс до списку провайдерів (рис.18)





Рис.18. Включення сервісу до провайдерів.

Далі змінимо код компонента exam-header.component.ts для роботи із сервісом (рис.19).

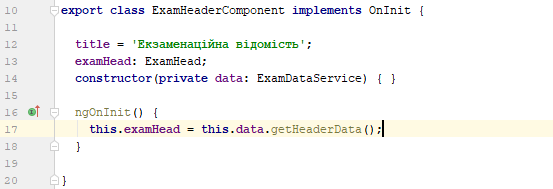


Рис.19. Зміни у exam-header.component.ts.

Перевіримо результат, перейшовши за посиланням «http://localhost:4200/». Зовнішній вигляд сторінки має бути незмінним.

1. Змінимо створений сервіс таким чином, щоб він звертався до створеного у попередній роботі rest api (рис.20).

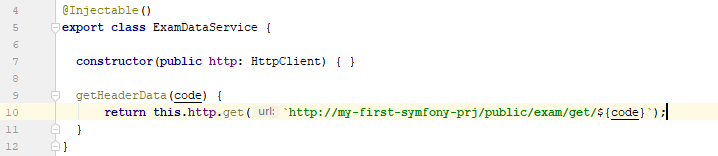


Рис.20. http-запит до rest-api.

Та виконаємо коректну обробку результату, що повертає даний метод у компоненті exam-header.component.ts (рис.21).

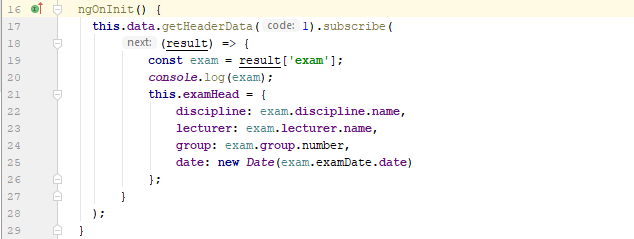


Рис.21. Зміни у exam-header.component.ts.

1. Встановимо font-awesome для використання іконок для кнопок редагування даних (рис.22-23).



Рис.22. Установка font-awesome.



Рис.23. Зміни у .angular-cli.json

Додамо до нашого застосунку компонент для відображення табличної частини відомості зі студентами та кількістю балів (рис.24-28)



Рис.24. Додавання компонента.



Рис.25. Додавання інтерфейсу.

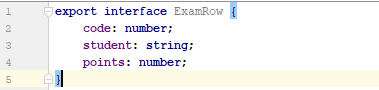
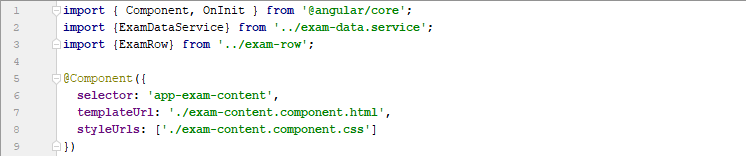
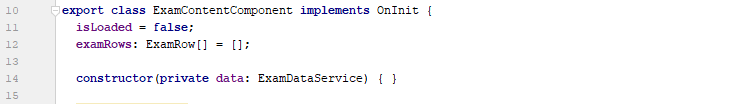
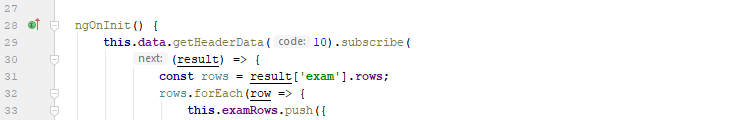


Рис.26. Інтерфейс «рядок».









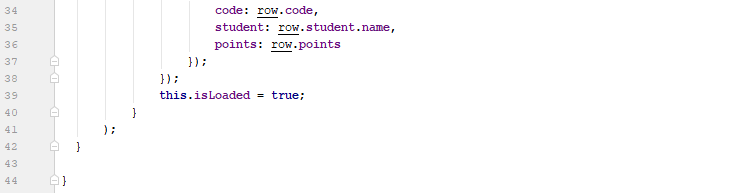
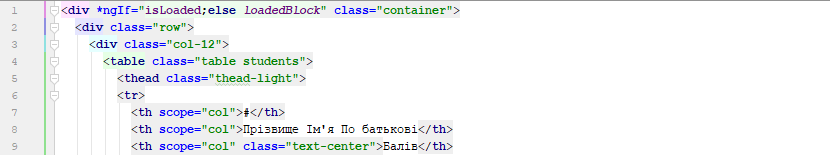
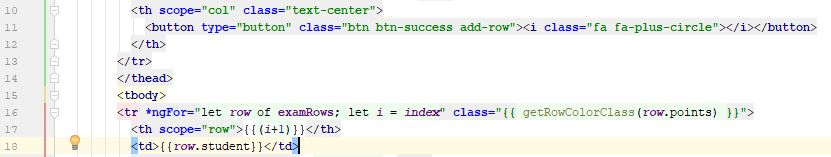


Рис.27. Клас компонента.





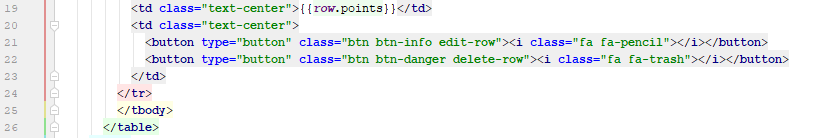




Рис.28. Шаблон компонента.

Додамо новий компонент відображення табличної частини до корньового компоненту (рис.29-30)

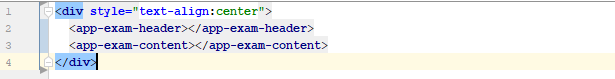
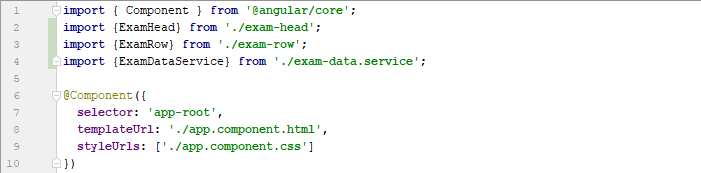


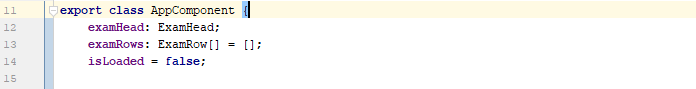
Рис.29. Зміни у корньовому компоненті

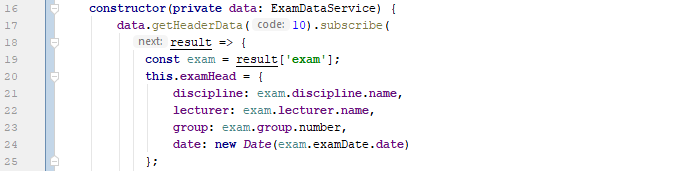


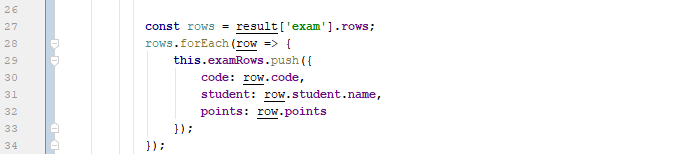
Рис.30. Зовнішній вигляд сторінки

1. Недолік роботи застосунку на даний момент проявляється у подвійному запиті до БД із компоненту exam-header та exam-content. Виправимо це, виконавши запит із корньового компоненту та передавши необхідні дані у дочірні. По перше, виконаємо зміни у класі корньвого елементу app-component (рис.31) та його шаблоні (рис.32).









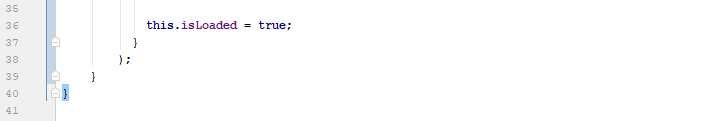


Рис.31. Клас компоненту app-component

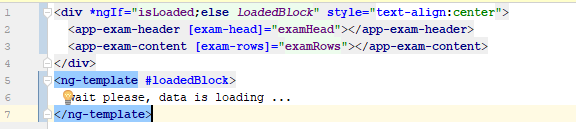
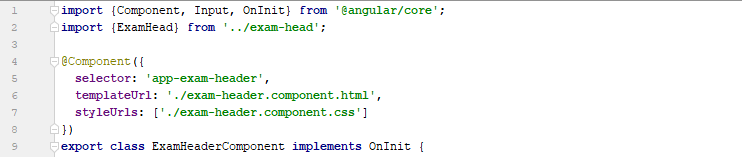


Рис.32. Шаблон компоненту.

Приберемо звернення до даних із дочірніх компонентів (рис.33-34). Також приберемо перевірку на завантаження даних із шаблонім дочірніх компонент.



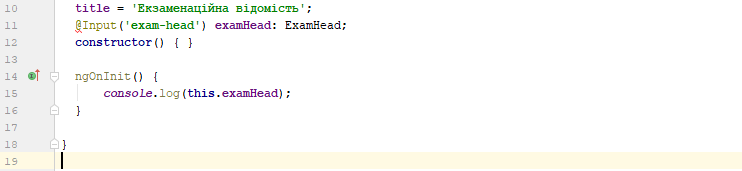
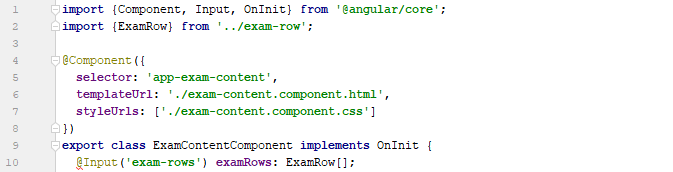


Рис.33. Клас компоненту шапки відомості



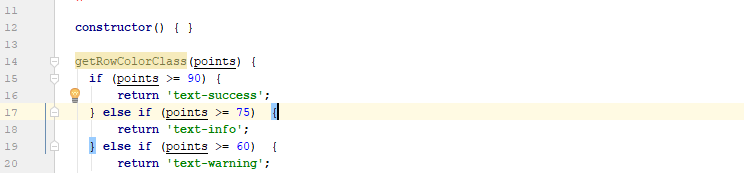




Рис.34. Клас для компоненту табличної частини.

Після виконання наведених вище змін, звернення до rest-api сервісу, а отже і до БД виконується 1 раз.

1. Наступний крок – реалізувати додавання рядку. Для цього виконаємо реалізацію відповідних методів у сервісі ExamDataService (рис.35-36)

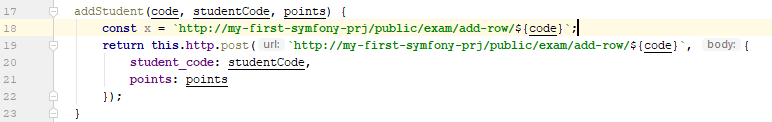


Рис.35. Метод додавання рядку



Рис.36. Метод визначення доступних для додавання студентів.

Додамо у клас компонента exam-content 2 метода, що відображає модальну форму, та додають новий рядок (рис.36-37)



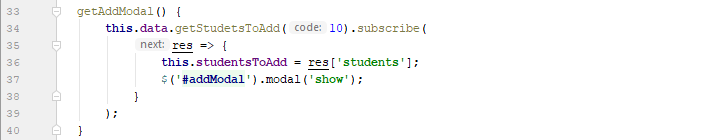
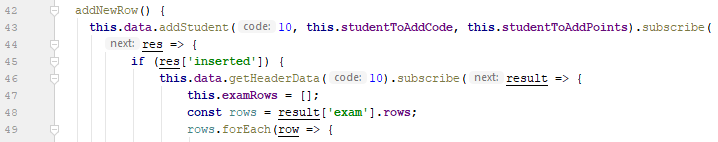


Рис.36. Метод, що отримає список студентів та відображає форму.



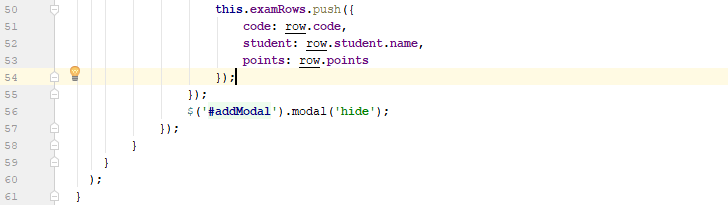
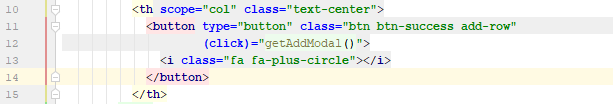
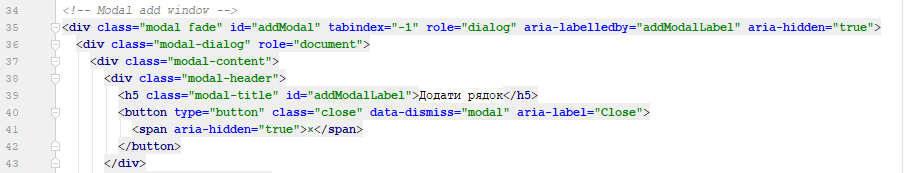
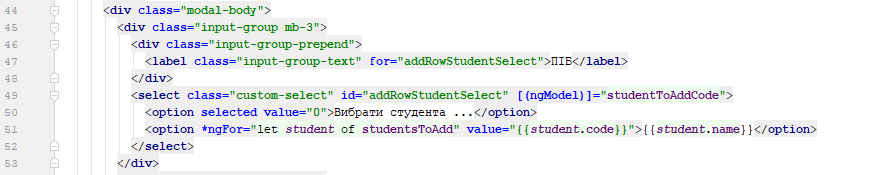


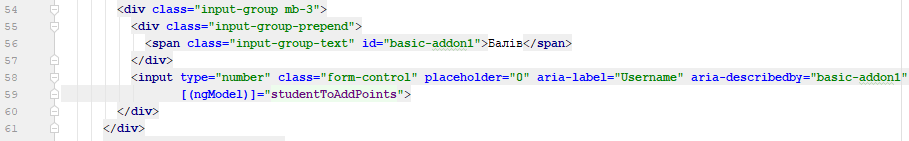
Рис.37. Метод, що додає новий рядок до відомості.

Додамо саму модальну форму та обробимо натискання кнопки «додати» у шаблоні компонента «exam-content» (рис.38)









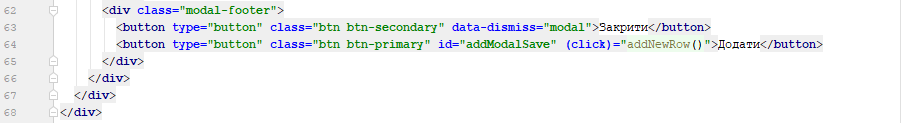


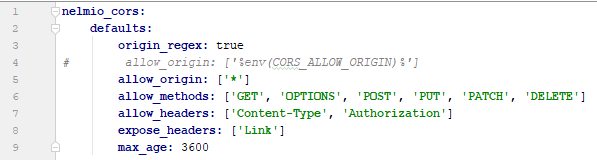
Рис.38. Зміни у шаблоні «exam-content»

Також нам доведеться трохи змінити наш back-end resp-api для уникнення помилки No 'Access-Control-Allow-Origin' header, що пов’язана із надсиланням запиту до back-end з іншого домену. Для цього встановимо у наш symphony проект NelmioCorsBundle (рис.39)



Рис.39. Встановлення NelmioCorsBundle.

Та виконаємо налаштування файлу config\packages\ nelmio\_cors.yaml



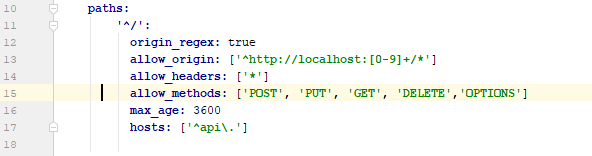


Рис.40. nelmio\_cors.yaml.

Також виконаємо невеликі зміни у методі addRow контролеру ExamController (рис.41), оскільки angular надсилає post як тіло запиту.

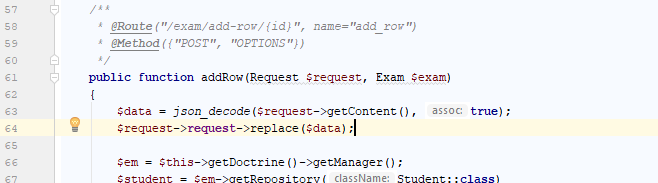


Рис.41. Зміни у методі addRow у back-end rest-api

Тепер ідемо на сторінку застосунку та пробуємо додати новий рядок до відомості.

1. Реалізація зміни рядку має схожі етапи із додаванням нового рядку. Почитаємо із класу-сервісу front-end, де реалізуємо метод отримання даних рядку та модифікації рядку (рис.42-43).

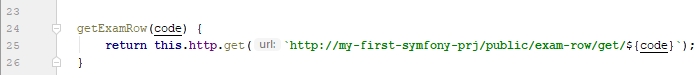


Рис.42. Метод отримання даних рядку у сервісі exam-data.

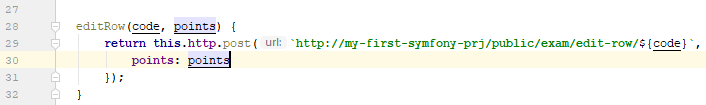
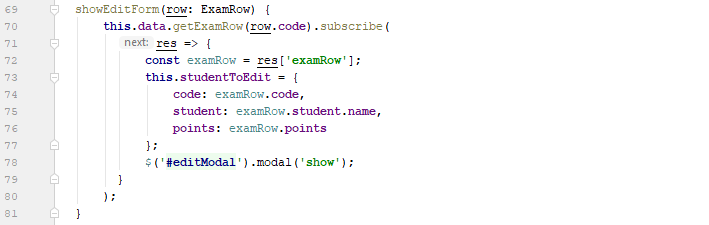


Рис.43. Метод редагування рядку у сервісі exam-data.

Далі реалізуємо методи для подій натискання кнопок «редагувати» у списку та на модальній формі. Також додамо змінну studentToEdit для зберігання змінених даних (рис.44). Дані зміни стосуються класу компоненту exam-content.





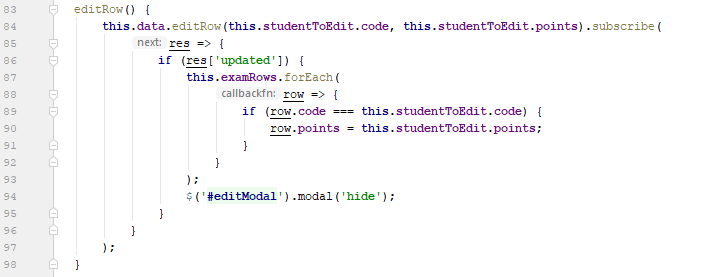


Рис.44. Зміни у компоненті exam-content.

До шаблону компоненту exam-content додаємо модальну форму для редагування рядку та подію для натискання кнопки «редагувати» (рис.45-46)

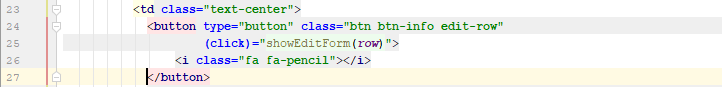
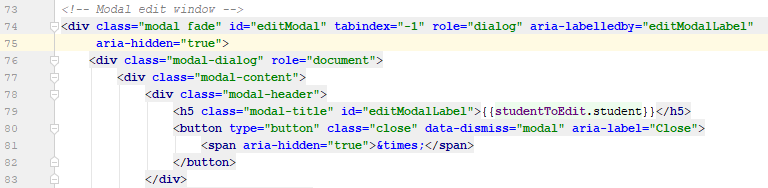
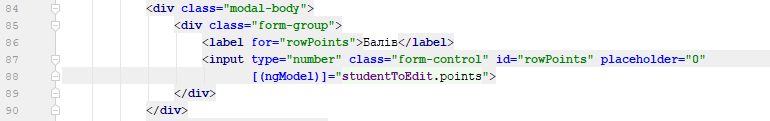


Рис.45. Обробка події натискання на кнопку.





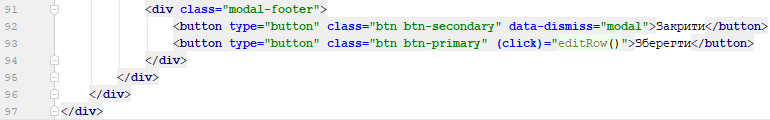


Рис.46. Модальна форма редагування рядку.

Також виконуємо невеликі зміни у методі editRow контролера ExamController нашого back-end rest-api для обробки тіла post-запиту angular (рис.47).

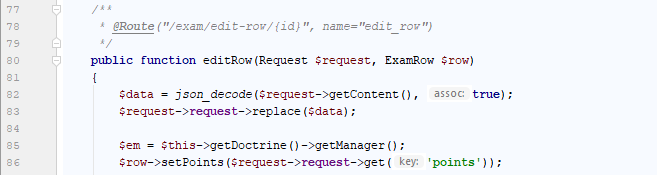


Рис.47. Зміни у методі editRow контролера ExamController

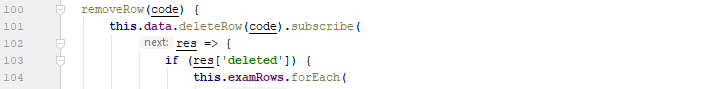
Після всіх указаних змін та доповнень виконаємо перевірку роботи редагування рядку у нашому застосунку.

1. Останній крок – реалізація операції видалення рядку. Послідовність кроків така ж сама – спочатку реалізуємо метод у сервісі exam-data (рис.48)



Рис.48. Метод deleteRow

Далі у класі компоненту exam-content додаємо метод, що буде спрацьовувати на подію натискання на кнопку «видалити»



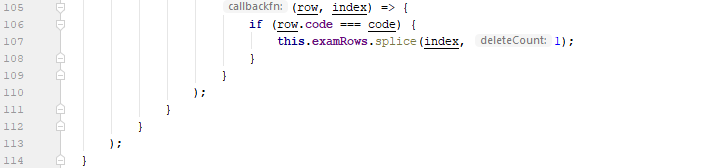


Рис.49. Метод removeRow компонента exam-content

І, нарешті, додаємо полію для натискання на відповідну кнопку у шаблоні компонента exam-content (рис.50)

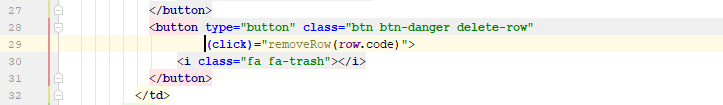


Рис.50. Зміни у шаблоні компонента exam-content.

Перевіряємо роботу кнопки видалення рядку.

1. Кінець роботи.

**Завдання для індивідуального виконання.**

.Для створеного у попередніх роботах back-end rest-api сервісу реалізувати користувацький інтерфейс із використанням фреймворку angular.