**Кейс №4**

**«Создание регистрации»**

**Выполнение работы:**

После объявления всех моделей в базе данных, мы можем начать создавать обработчики путей. Для этого я использовал router. Это позволит мне отделить логику от приложения и затем добавить ее, как модуль. Сам router выглядит так.



Рисунок 1

Вот весь route создания пользователя. Тут много моментов, нуждающихся во внимании.

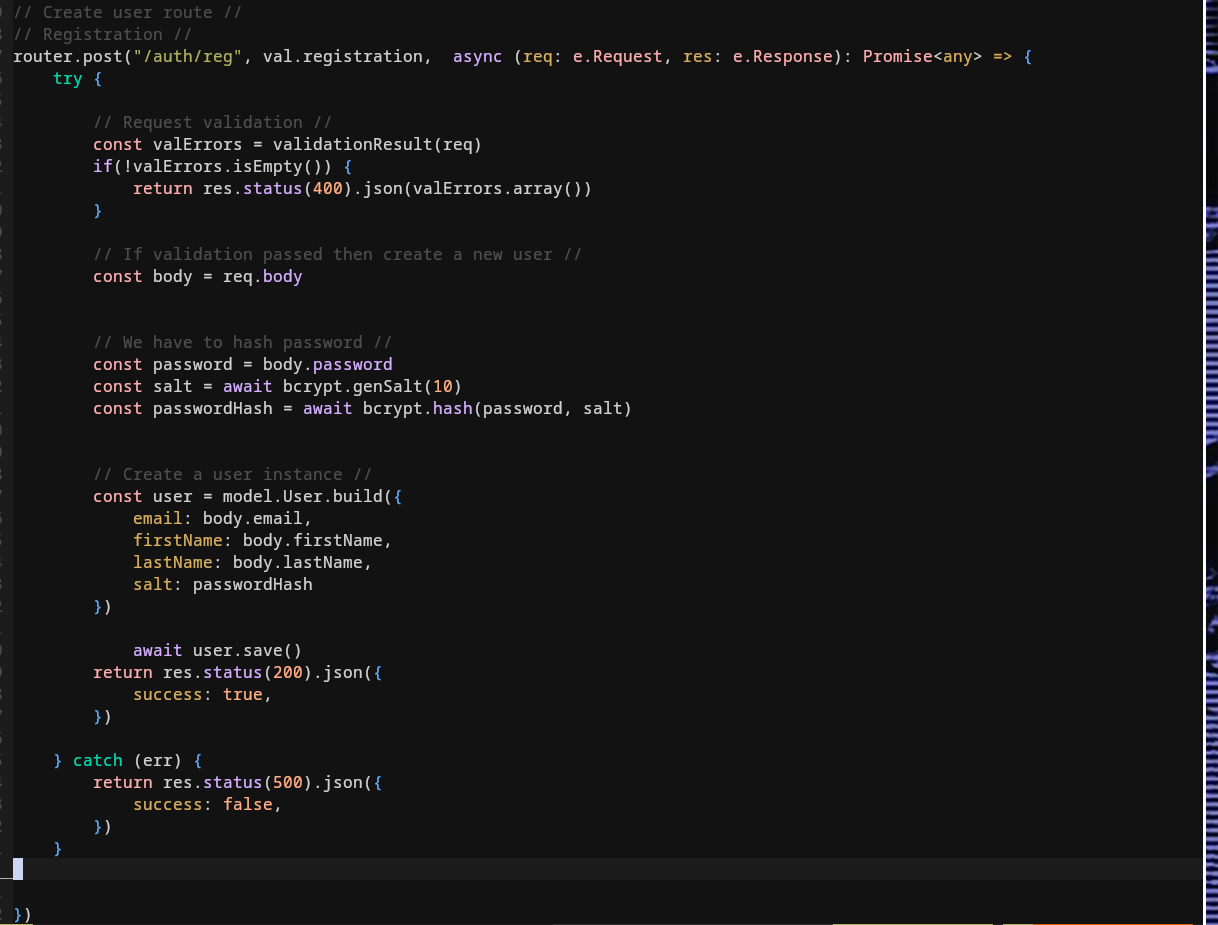


Рисунок 2

Что здесь используестся. Для начала отмечу, что я использую отдельный validator данных. Он смотрит тело приходящего запроса и проверяет данные на соответствие критериям. Например в данном случае мне нужно увидеть поля email, firstName, lastName и password для того, чтобы зарегестрировать пользователя в системе. В коде это выглядит так. У меня есть отдельная папка validator в которой и хранится проверка данных.

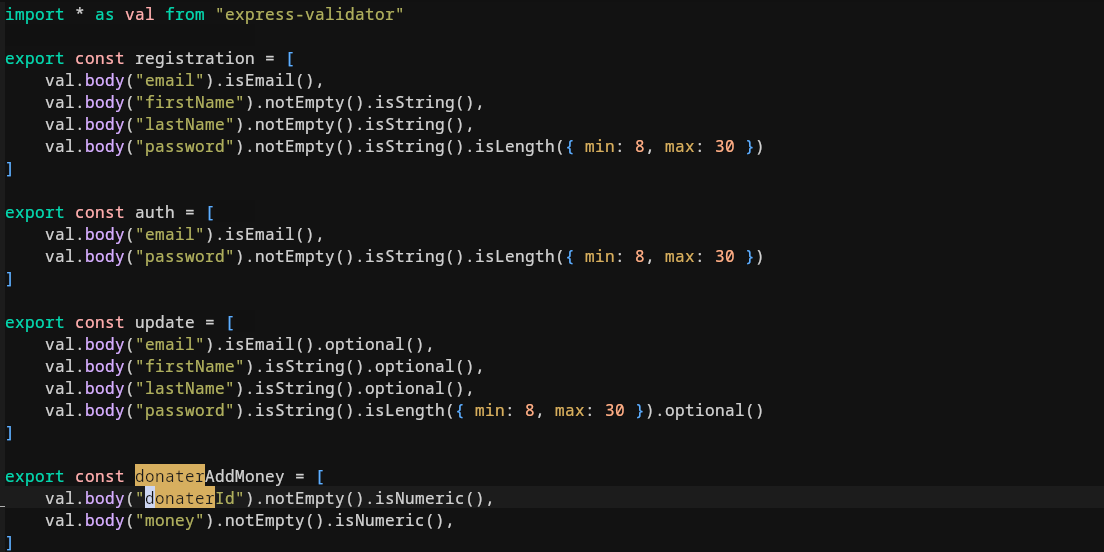


Рисунок 3

Если какое то поле не соответствует описанным критериям, то отправителю присылается ошибки, с описанием того, какие данные были отправлены не так. Как, например, здесь. Я не указал email и получил ошибку. Тело запроса указано слева, а ответ сервера справа.

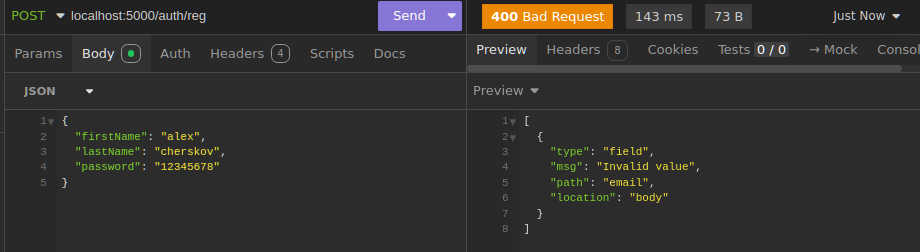


Рисунок 4

Указав email я повторил запрос и мне пришел ответ об успешной регистрации.

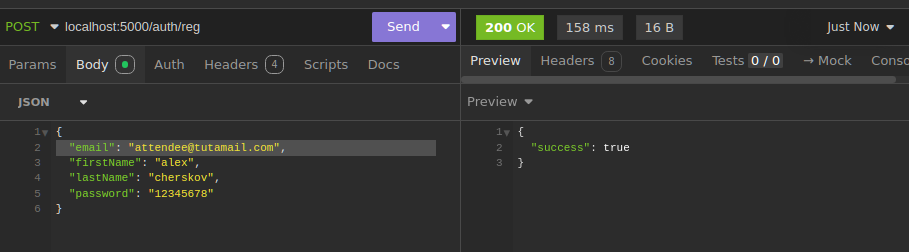
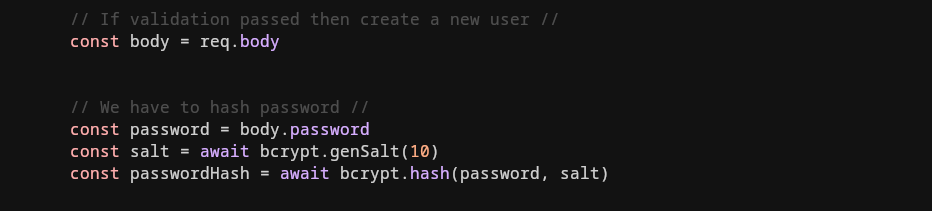
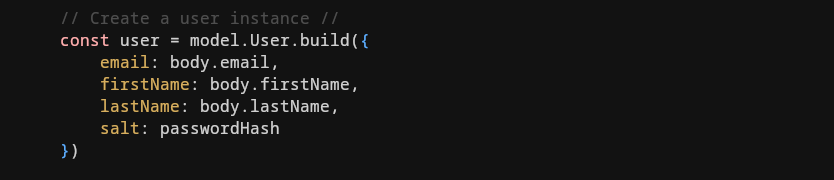


Рисунок 5

Далее идет обработка пароля. Хранить пароль в базе данных в открытом виде — не безопасно. Обычно его хешируют и сам хеш отправляют в базу данных. Для хеширования используется secret, взятый с config файла.

Рисунок 6

Далее идет обращение к базе данных, создается пользователь.

Рисунок 7

И в конце после того, как данные были записаны в экземпляр модели, делается сохранение и возвращается ответ пользователю.

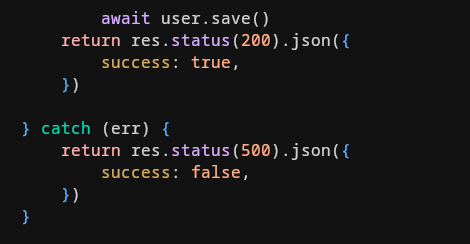
****

Рисунок 8

После того, как пользователь был создан, ему нужно дать возможность авторизоваться в системе. Для этого я написал отдельный обработчик. Выглядит он так

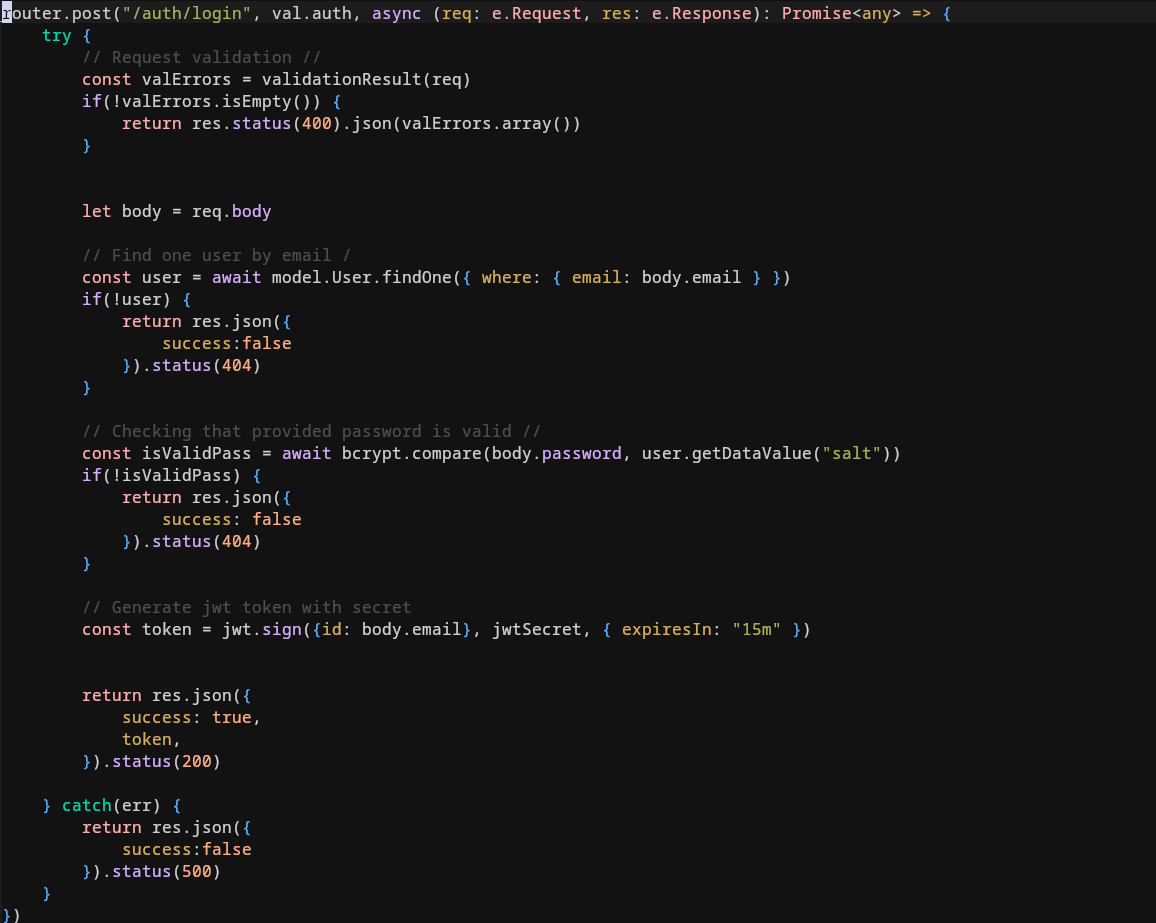


Рисунок 9

Разберем этот обработчик. Для начала я также, как и при регистрации валидирую приходящие данные. Затем пытаюсь найти запись в базе данных о пользователе и в случае отсутствия какой-либо записи делаю вывод, что такого пользователя не существует и возвращаю ошибку.

Далее происходит валидация пароля. Как я сказал ранее, пароли в открытом виде — это небезопасно. В базе данных хранятся хешированные пароли. И теперь нам нужно убедится, что предоставленный пароль — правильный пароль. Для этого нам нужно захешировать предоставленный пользователем пароль и получившийся захешированный пароль сравнить с хешем, полученным из базы данных. В случае совпадения, делаем вывод, что пароль, предоставленный пользователем верен, в ином случае отправляем обшибку.

Далее создаем jwt token. Это зашифрованная строка с данными, которую мы возвращаем пользователю. Работает это так. Мы предоставляем функции jwt.sign некоторый объект, скажем json, и его задача зашифровать этот объект таким образом, чтобы пользователь на стороне клиента, мог извлечь данные, но не мог их изменить. Этот зашифрованный объект пользователь отправляет на сервер каждый раз, когда ему нужно сделать какое-либо действие. И если пользователь попробует поменять данные в token, то при валидации на сервере мы получим ошибку и поймем, что предоставленный token не является валидным. На этом и построена авторизация.

Для простоты я создал специальный middleware. Это функция, которая проверяет предоставленный пользователем jwt token на целостность. Вот код.

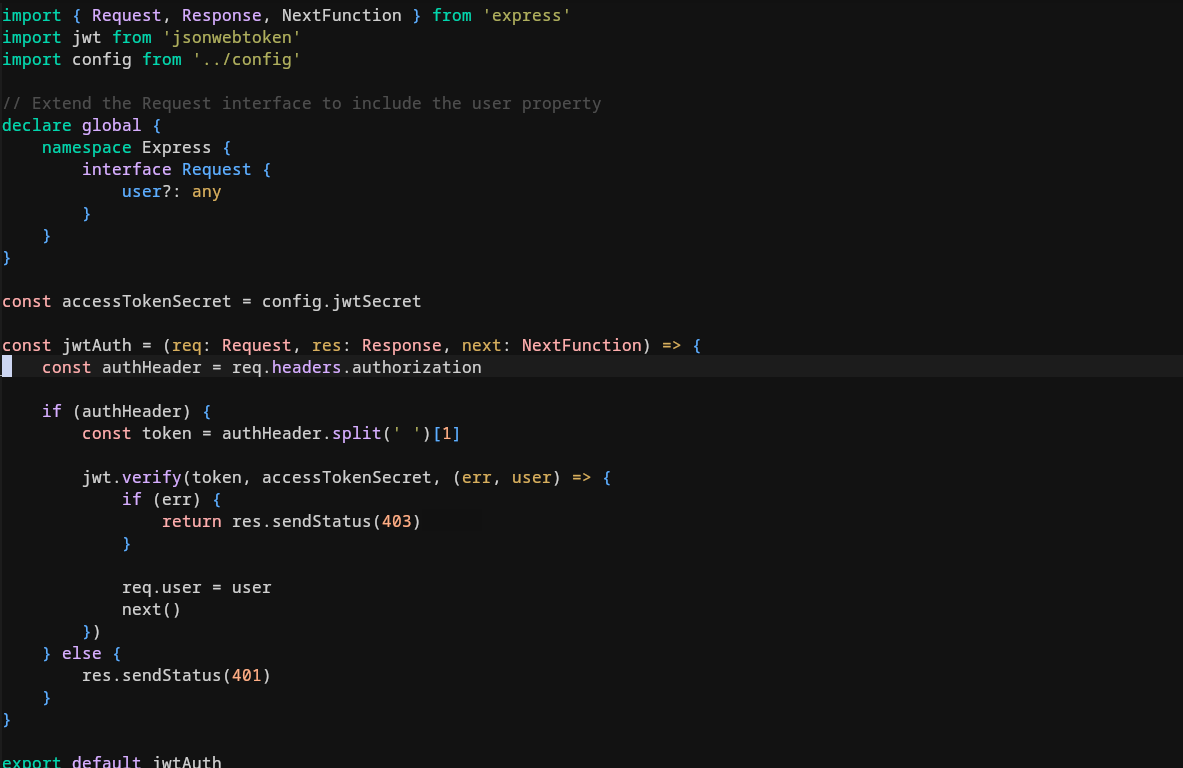


Рисунок 10

Теперь я могу прикрепить этот middleware к любому обработчику и обработчик будет защищен от неавторизованного пользователя. Вот пример обработчика, выводящего всех пользователей

Рисунок 11

Как вы можете видеть, jwtAuth подключен. Попытаемся получить пользователей без token ключа.

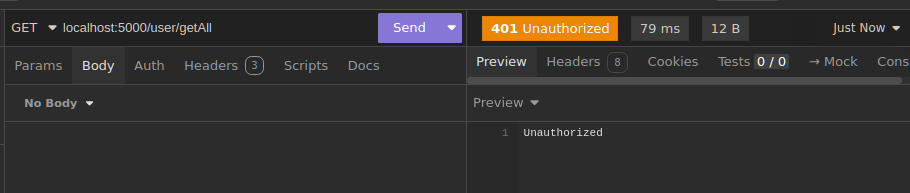


Рисунок 12

Как видим, попытка получить пользователей без jwt token привела к ошибке, как и должно быть. Теперь route защищен.

Попробуем зайти с помощью jwt token, полученного при заходе в систему.

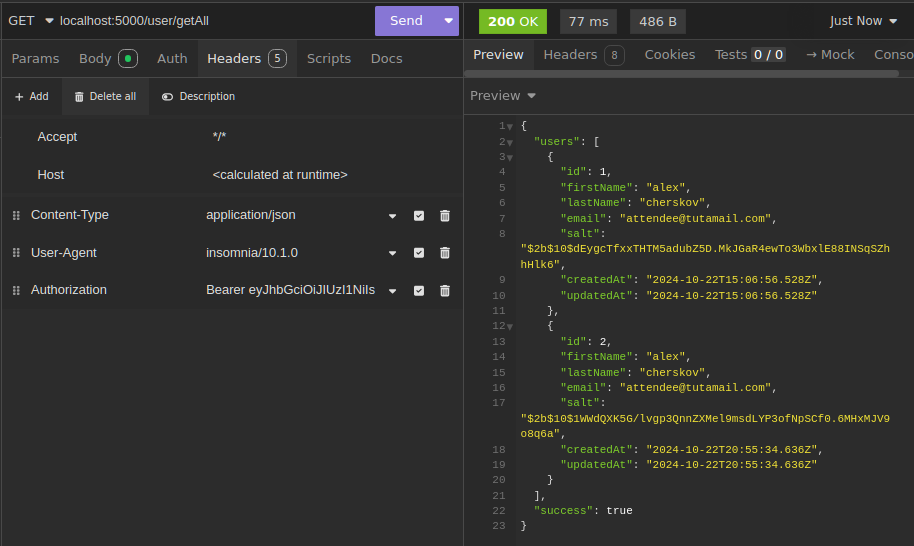


Рисунок 13

Все работает. Авторизация добавлена. Однако я забыл показать вам route для авторизации в систему. Давайте это исправвим.

Вот код.

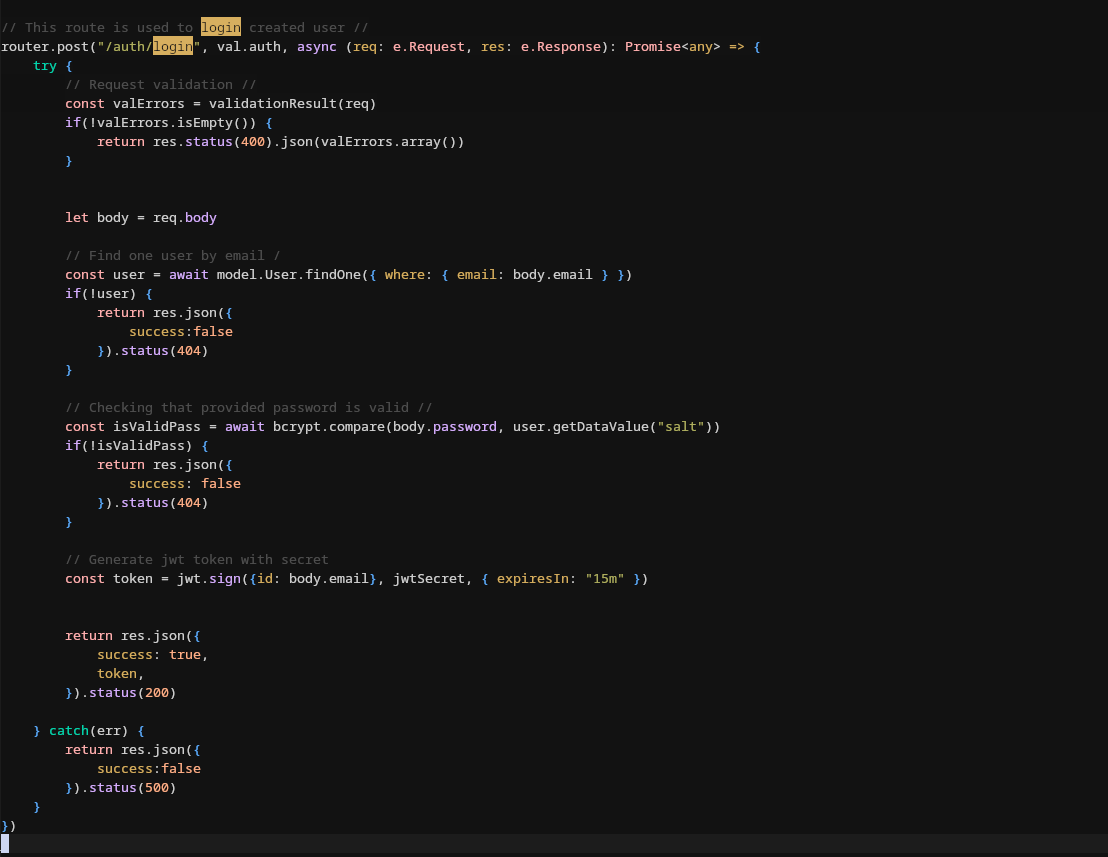


Рисунок 14

Здесь ничего нового, за исключением одного — создания jwt token. Заметьте, что token имеет срок действия, равный 15 минутам.

