Оглавление

[Bcrypt 1](#_Toc177116033)

[JWT 1](#_Toc177116034)

[API 2](#_Toc177116035)

[Создание нового аккаунта 2](#_Toc177116036)

[Аутентификация пользователя 3](#_Toc177116037)

# Bcrypt

Bcrypt – стандартаня библиотека golang

Процесс хеширования паролей преобразует исходный пароль в фиксированную строку символов (хеш) с помощью односторонней криптографической функции. К паролю добавляется случайная "соль", что делает каждый хеш уникальным даже для одинаковых паролей. Результат многократно обрабатывается алгоритмом, что замедляет процесс и усложняет подбор. Полученный хеш невозможно обратить в исходный пароль, что обеспечивает безопасность даже при компрометации базы данных.

# JWT

В проекте используется JWT токен с временной меткой 15 минут, для подтверждения авторизации пользователя.

Хранится в куки и отправляется вместе с каждой операцией, которая требует подтверждение пользователя.

JWT токен не хранится на серверной части, а создается каждый раз при авторизации/создании нового аккаунта/обновлении.

JWT состоит из закодированной структуры {id пользователя, секретного ключа из. env} + временная метка. JWT кодируется каждый раз одинаково, то обеспечит доступность авторизации на 15 минут. По истечении времени, токен не сможет закодироваться также, а значит не пройдет проверку на сервере.

[JSON Web Tokens](https://ru.wikipedia.org/wiki/JSON_Web_Token) — это открытый стандарт RFC 7519 для создания токенов доступа. Используется в передаче данных для аутентификации в клиент-серверных приложениях. В обычных веб-приложениях легко идентифицировать пользователей с помощью сессий, однако, когда API вашего веб-приложения взаимодействует, скажем, с клиентом Android или IOS, сессии становятся малопригодными для использования. С помощью JWT мы можем создать уникальный токен для каждого аутентифицированного пользователя. Этот токен будет включён в заголовок последующего запроса к API. Этот метод позволяет идентифицировать всех пользователей, которые выполняют вызовы API. Давайте посмотрим реализацию:

# API

В данный момент хешируется только пароль.

Реализовано два пути:

## Создание нового аккаунта

 Метод: POST

 Путь: /api/user/new

 Обработчик: controllers.CreateAccount

 Описание: Регистрация нового пользователя

- Запрос:

|  |
| --- |
| {         "email": "пользователь@example.com",         "password": "пароль123"       } |

- Ответ (успех):

|  |
| --- |
| {         "status": true,         "message": "Account has been created",         "account": {           "id": 1,           "email": "пользователь@example.com",           "token": "JWT\_токен"         }   } |

- Ответ (ошибка):

|  |
| --- |
| {         "status": false,         "message": "Invalid request"   } |

## Аутентификация пользователя

 Метод: POST

 Путь: /api/user/login

 Обработчик: controllers.Authenticate

 Описание: Вход пользователя в систему

- Запрос:

|  |
| --- |
| {         "email": "пользователь@example.com",         "password": "пароль123"       } |

- Ответ (успех):

|  |
| --- |
| {         "status": true,         "message": "Logged In",         "account": {           "id": 1,           "email": "пользователь@example.com",           "token": "JWT\_токен"         }       } |

- Ответ (ошибка):

|  |
| --- |
| {         "status": false,         "message": "Invalid login credentials. Please try again"       } |

# Описание пакетов

* Main – устанавливает маршруты, запускает проекта
* Console – включает тестирование api через консоль
* Handlers – хранит http обработчики
* Middleware – хранит обработчики, которые запускаются перед основными. В частности, проверка токена.
* Model – хранит структуры и их методы, связанные с БД.
* Utils – хранит унитарные функции, которые нельзя четко приписать к конкретному пакету.