Setup

1. Create a database and run the sql script (current script is only for MariaDB/MySQL) which is included in the zip
2. Configure the following application.properties file

example configuration:

spring.datasource.username=root

spring.datasource.password=

spring.datasource.url=jdbc:mysql://localhost:3306/blog2

3. Set JVM properties private.key and jwt.expiration.minutes

exmple configuration:

-Dprivate.key=secret

-Djwt.expiration.minutes=100

After all the steps are complete the application will be ready for usage

Api Documentation

Auth

Method	Url	Decription
POST	/register	Sign up
POST	/login	Log in

Blog entries

Method	Url	Decription
POST	/entries	Create entry
GET	/entries	Get entries list
PUT	/entries/{entryId}	Update entry
DELETE	/entries/{entryId}	Delete entry

Sample authentication requests and responses

Register

URL : `/register`

HTTP Method : `POST`

Auth required : No

Body :

```
{
    "email": "sample@mail.com",
    "password": "samplePassword"
}
```

Success Response

Code : `200 OK`

Non valid email or blank password

Code : `400 Bad Request`

Registration with the same email

Code : `409 Conflict`

Login

URL : `/login`

HTTP Method : `POST`

Auth required : Email and password in body

Body :

```
{
    "email": "sample@mail.com",
    "password": "samplePassword"
}
```

Success Response

Code : `200 OK`

```
"jwt" : "{JWT_TOKEN}"
```

Ivalid credentials

Code : `401 Unauthorized`

Non valid email or blank password

Code : `400 Bad Request`

Sample blog requests and responses

Create blog entry

URL : `/entries`

HTTP Method : `POST`

Auth required : JWT Token in authorization header e.g. `Authorization: Bearer {JWT_TOKEN}`

Body :

```
{
    "tittle": "sample tittle",
    "text": "sample text"
}
```

Success Response

Code : `201 Created`

Ivalid or expired JWT token

Code : `401 Unauthorized`

Blank tittle

Code : `400 Bad Request`

Get user all blog entries

URL : `/entries`

HTTP Method : `GET`

Auth required : JWT Token in authorization header e.g. `Authorization: Bearer {JWT_TOKEN}`

Success Response

Code : `200 OK`

```
[
  {
    "entryId": 0,
    "tittle" : "sample tittle",
    "text" : "sample text"
  },
  {
    "entryId": 1,
    "tittle" : "sample tittle",
    "text" : "sample text"
  }
]
```

Ivalid or expired JWT token

Code : `401 Unauthorized`

Update blog entry

URL : `/entries/{entryId}`

HTTP Method : `PUT`

Auth required : JWT Token in authorization header e.g. `Authorization: Bearer {JWT_TOKEN}`

Path parameter : {entryId} - Integer of the entry that will be updated

Body:

```
 {
    "tittle" : "sample tittle",
    "text" : "sample text"
  },
```

Success Response

Code : `200 Ok`

Ivalid or expired JWT token

Code : `401 Unauthorized`

Blank tittle

Code : `400 Bad Request`

Entry does not exist

Code : `404 Not Found`

Delete blog entry

URL : `/entries/{entryId}`

HTTP Method : `DELETE`

Auth required : JWT Token in authorization header e.g. `Authorization: Bearer {JWT_TOKEN}`

Path parameter : {entryId} - Integer of the entry that will be deleted

Success Response

Code : `200 Ok`

Ivalid or expired JWT token

Code : `401 Unauthorized`

Entry does not exist

Code : `404 Not Found`