Xebia Coding Assignment

Things to keep in mind

1. Please don't post the solution on public Github, Bitbucket, Gitlab repositories.
2. Please proceed in story-wise manner so if you find yourself running out of time you can at least submit the already completed stories.

Evaluation criteria

Please keep the following in mind:

1. We are expecting solution built using Java 8 and Spring Boot 2.x. You can use https://start.spring.io/
2. This application will need a backing database - you can use either H2 (or another in-memory DB) or feel free to use any other SQL/NoSQL DB. Please provide details in README.md on your choice of database and how to connect to it. For relational DBs, kindly provide schema creation script and mention the data model.

Your code will be evaluated on following parameters:

- Correctness
- Code quality and cleanliness
- Good use of OOPS concepts
- Extensible – easily adaptable to future enhancements
- Automation testing
- REST API Design
- A README.md that clearly mentions steps needed to build and run the application.

Requirments

You have to create a REST API service which implements the following stories.

Story 1: REST API to create an article

Example request body:

```
{
"title": "How to learn Spring Booot",
"description": "Ever wonder how?",
"body": "You have to believe"
}
```

Required fields: `title` , `description` , `body`

The response should be below. A slug is the part of a URL which identifies a particular page on a website in an easy to read form.

```
{
"id": 1,
"slug": "how-to-learn-spring-boot",
"title": "How to learn Spring Booot",
"description": "Ever wonder how?",
"body": "You have to believe",
"createdAt": "2019-11-24T03:22:56.637Z",
"updatedAt": "2019-11-24T03:48:35.824Z"
}
```

Story 2: Update an article

We will update the article using its slug identifier. Below is an example request body.

```
{
"title": "How to learn Spring Boot by building an app",
"description": "Ever wonder how?",
"body": "You have to believe"
}
```

All the fields are optional. The slug also gets updated when article is changed.

```
{
"id": 1,
"slug": "how-to-learn-spring-boot-by-building-an-app",
"title": "How to learn Spring Booot",
"description": "Ever wonder how?",
"body": "You have to believe",
"createdAt": "2019-11-24T03:22:56.637Z",
"updatedAt": "2019-11-25T03:48:35.824Z"
}
```

Story 3: Get an article by slug id

This returns single article

```
{
"id": 1,
"slug": "how-to-learn-spring-boot-by-building-an-app",
"title": "How to learn Spring Booot",
"description": "Ever wonder how?",
"body": "You have to believe",
"createdAt": "2019-11-24T03:22:56.637Z",
"updatedAt": "2019-11-25T03:48:35.824Z",
}
```

Story 4: Delete an article by slug Id

Delete the article using slug id. After deletion GET should return 404 not found.

Story 5: Find time to read for an article using its slug id

The GET request should return following response:

```
{
"articleId": "slug-uuid",
"timeToRead": {
"mins" : 3,
"seconds" : 50
}
}
```

The logic to calculate time to read is `total number of words / speed of average human`

The speed of average human should be configurable.