WELCOME TO OUR PRESENTATION

by ButterflyAndBee

The following project is a collaboration effort between the following members on BNTA C11:

- Tarek Ahmed (Github: @TarekQMUL)
- Blezzy Dela Cruz (Github: @blezzydcruz)
- Jannah Anwar (Github: @jannahthecodemaster)
- Mohamed I. Hussain (Github: @essamcreates)

PROJECT TIMELINE 1

Stock Management
Can oversee products quantities and order histories allowing for seamless dispute resolution

Scalability
Can efficiently add new products to the database as your business expands

Data Analysis
Can track winning products and sale trends

1

Brainstorming
Researched different types and topics for API models.
Looked into different ideas including transaction tracking app, games, outfit generator and real time chat room.

2

Planning
Discussed entity relationships and classes and attributes.
Distinguished between MVP and extension.
Awaiting approval for diagram

3

Coding
Created a rotational schedule for driving and navigation of code.
Created repositories and branches so that we can collaborate.
Estimate around 30mins per individual to work on code.

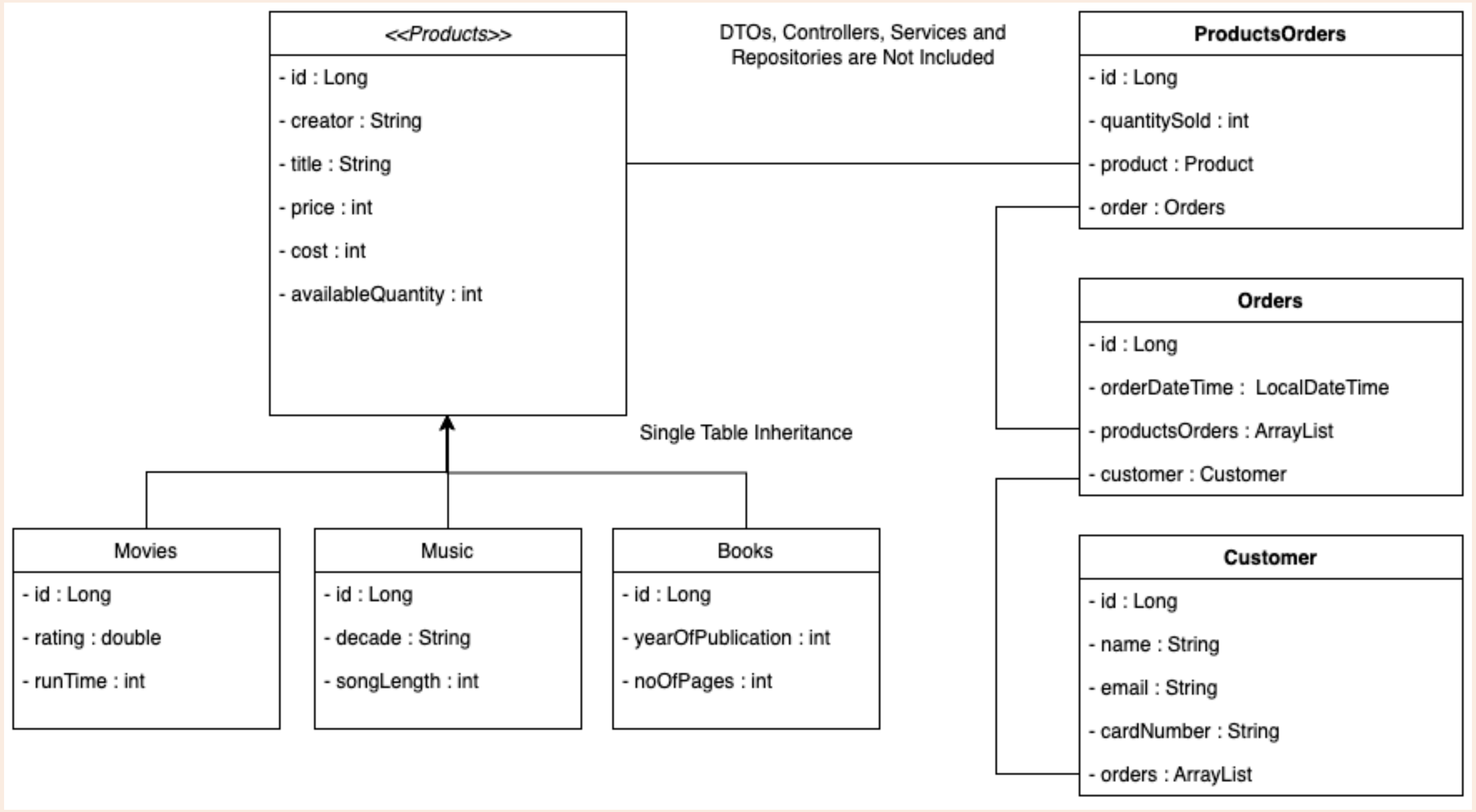
4

Finalisation
Amending errors, submitting requests on Postman and also thinking about extension tasks we've set.
Resolving merge conflicts.
Planning presentation and creating a detailed ReadME.

5

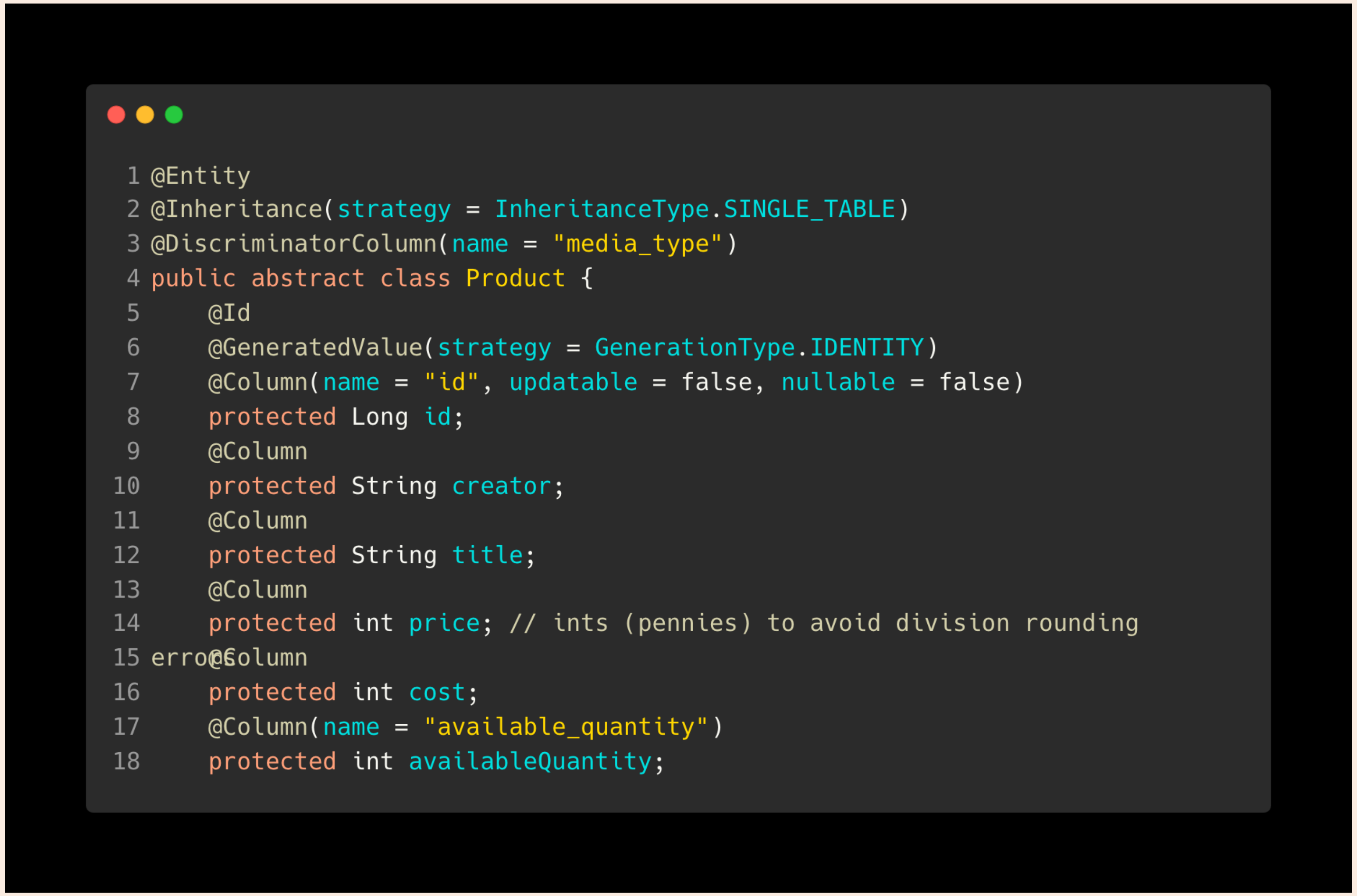
Today!

CLASS DIAGRAM

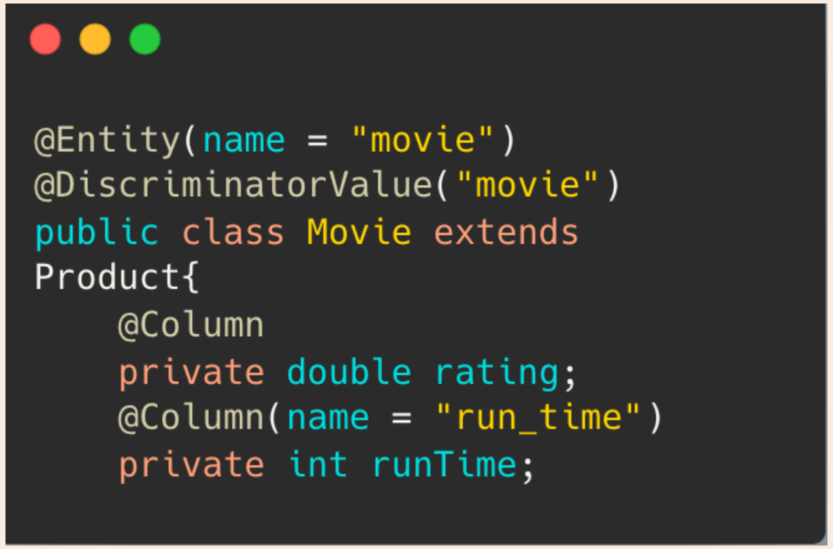


OUR API HIGHLIGHTS...

SINGLE TABLE ENTITY

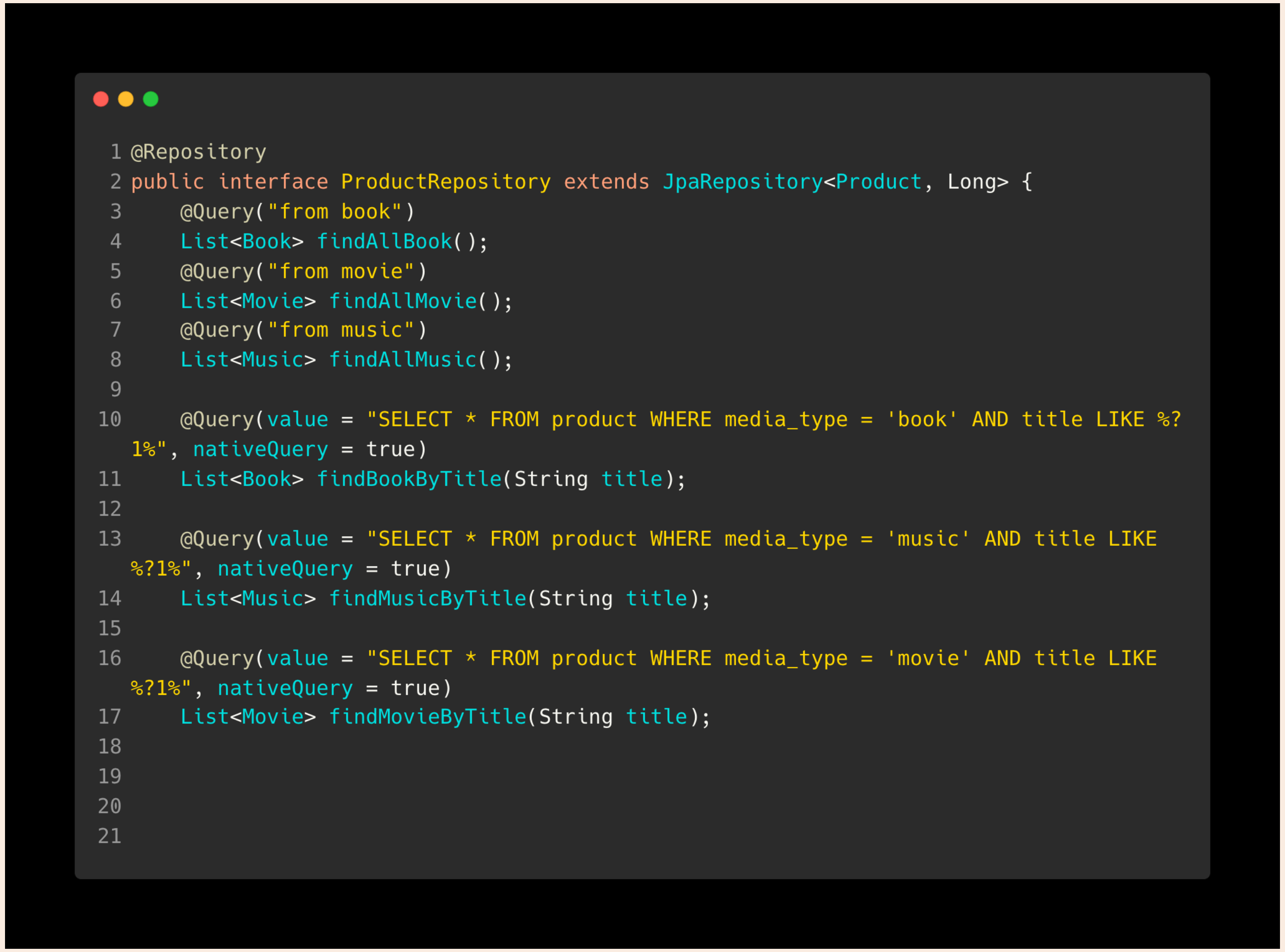


```java
@Entity
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name = "media_type")
public abstract class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", updatable = false, nullable = false)
    protected Long id;
    @Column
    protected String creator;
    @Column
    protected String title;
    @Column
    protected int price; // ints (pennies) to avoid division rounding
erro@Column
    protected int cost;
    @Column(name = "available_quantity")
    protected int availableQuantity;
```



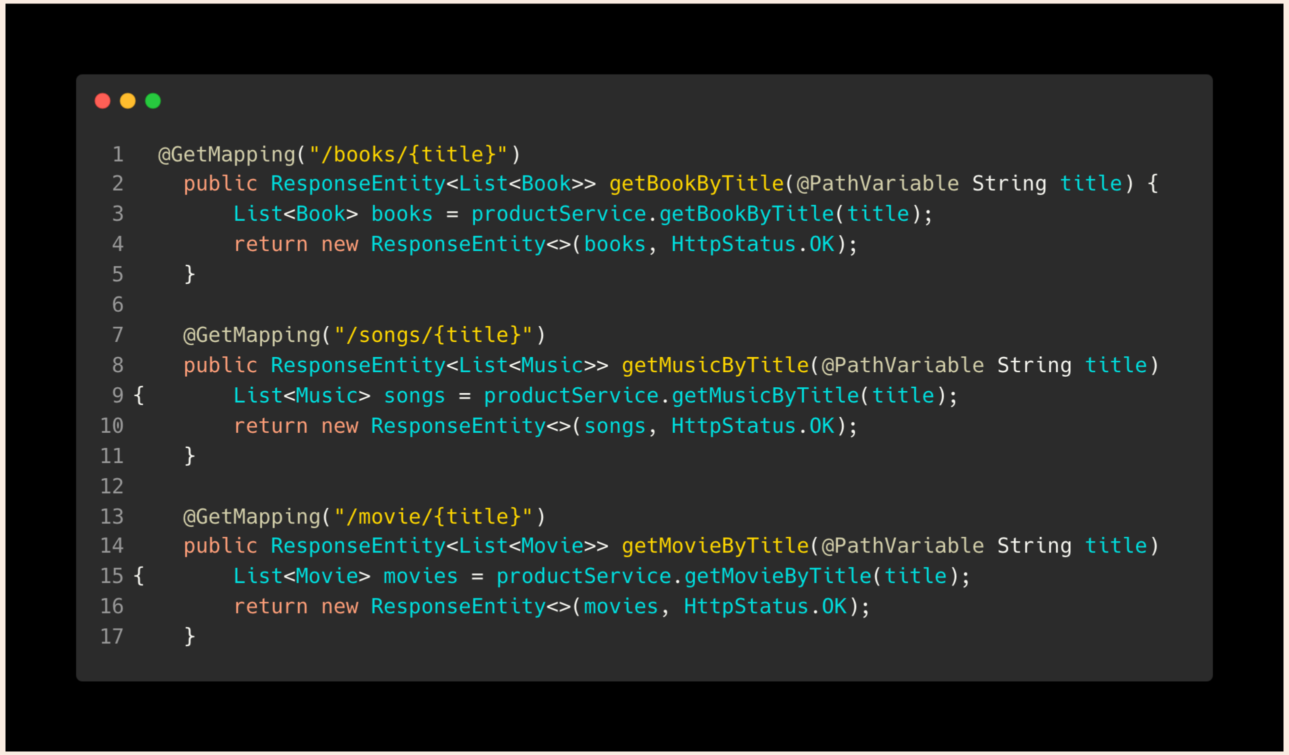
```java
@Entity(name = "movie")
@DiscriminatorValue("movie")
public class Movie extends
Product{
    @Column
    private double rating;
    @Column(name = "run_time")
    private int runTime;
```

PRODUCT REPOSITORY (SQL!)

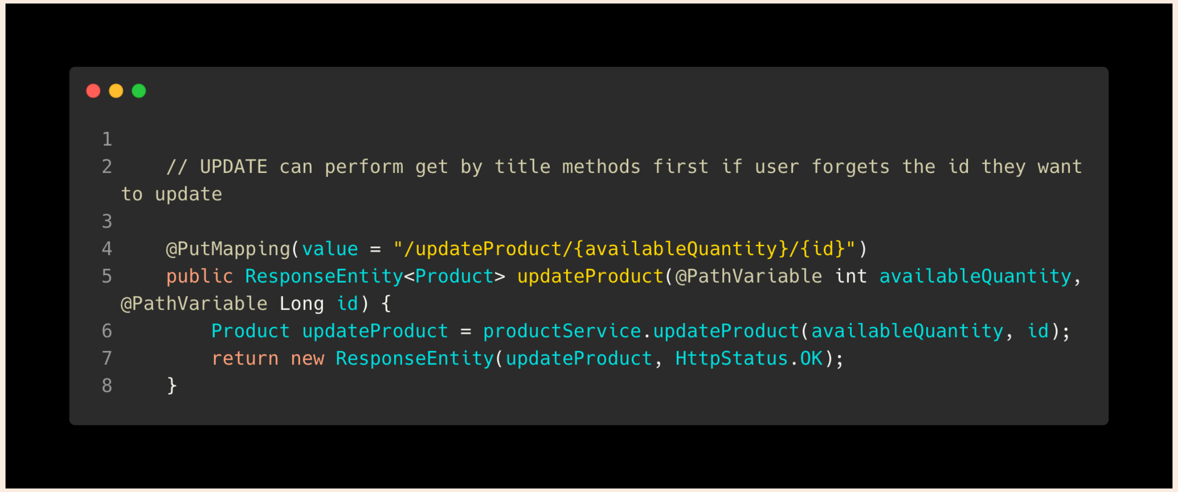


```java
@Repository
public interface ProductRepository extends JpaRepository<Product, Long> {
    @Query("from book")
    List<Book> findAllBook();
    @Query("from movie")
    List<Movie> findAllMovie();
    @Query("from music")
    List<Music> findAllMusic();

    @Query(value = "SELECT * FROM product WHERE media_type = 'book' AND title LIKE %?1%", nativeQuery = true)
    List<Book> findBookByTitle(String title);

    @Query(value = "SELECT * FROM product WHERE media_type = 'music' AND title LIKE %?1%", nativeQuery = true)
    List<Music> findMusicByTitle(String title);

    @Query(value = "SELECT * FROM product WHERE media_type = 'movie' AND title LIKE %?1%", nativeQuery = true)
    List<Movie> findMovieByTitle(String title);
```

UPDATING PRODUCT QUANTITIES



```java
@GetMapping("/books/{title}")
  public ResponseEntity<List<Book>> getBookByTitle(@PathVariable String title) {
      List<Book> books = productService.getBookByTitle(title);
      return new ResponseEntity<>(books, HttpStatus.OK);
  }

  @GetMapping("/songs/{title}")
  public ResponseEntity<List<Music>> getMusicByTitle(@PathVariable String title)
{
      List<Music> songs = productService.getMusicByTitle(title);
      return new ResponseEntity<>(songs, HttpStatus.OK);
  }

  @GetMapping("/movie/{title}")
  public ResponseEntity<List<Movie>> getMovieByTitle(@PathVariable String title)
{
      List<Movie> movies = productService.getMovieByTitle(title);
      return new ResponseEntity<>(movies, HttpStatus.OK);
  }
```

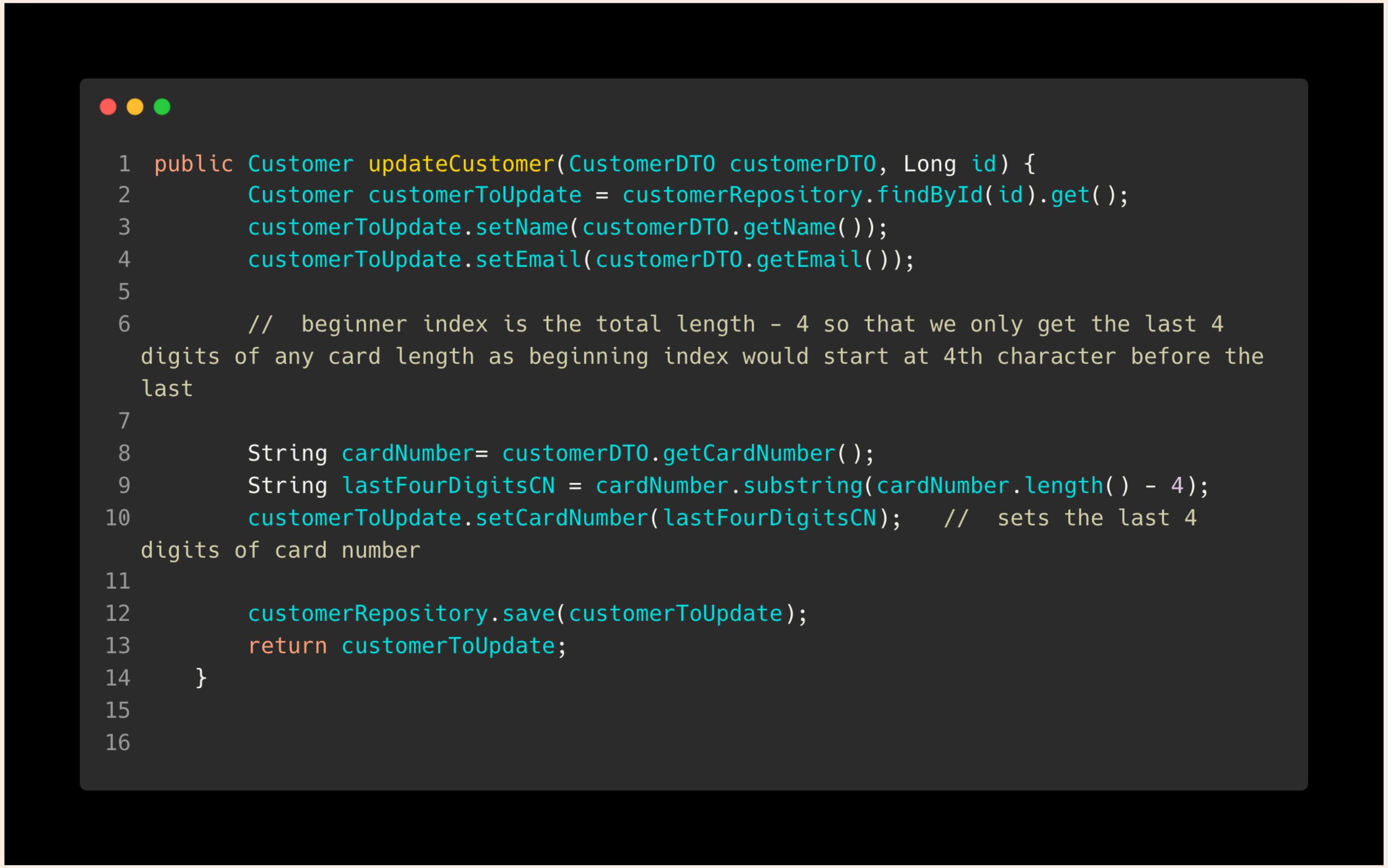


```java

    // UPDATE can perform get by title methods first if user forgets the id they want
to update

    @PutMapping(value = "/updateProduct/{availableQuantity}/{id}")
    public ResponseEntity<Product> updateProduct(@PathVariable int availableQuantity,
@PathVariable Long id) {
        Product updateProduct = productService.updateProduct(availableQuantity, id);
        return new ResponseEntity(updateProduct, HttpStatus.OK);
    }
```

1

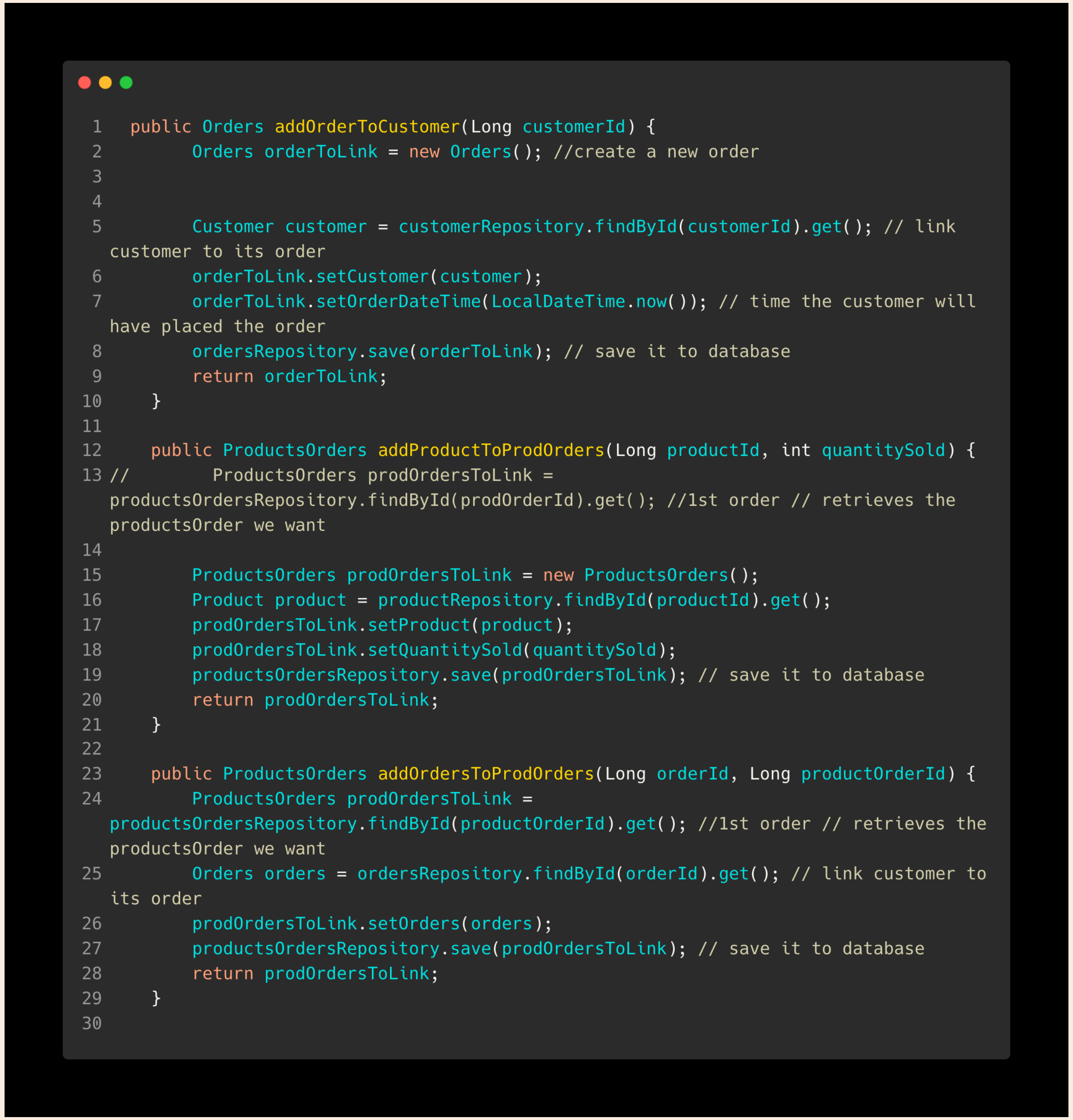
2

CARD NUMBER SPLICING

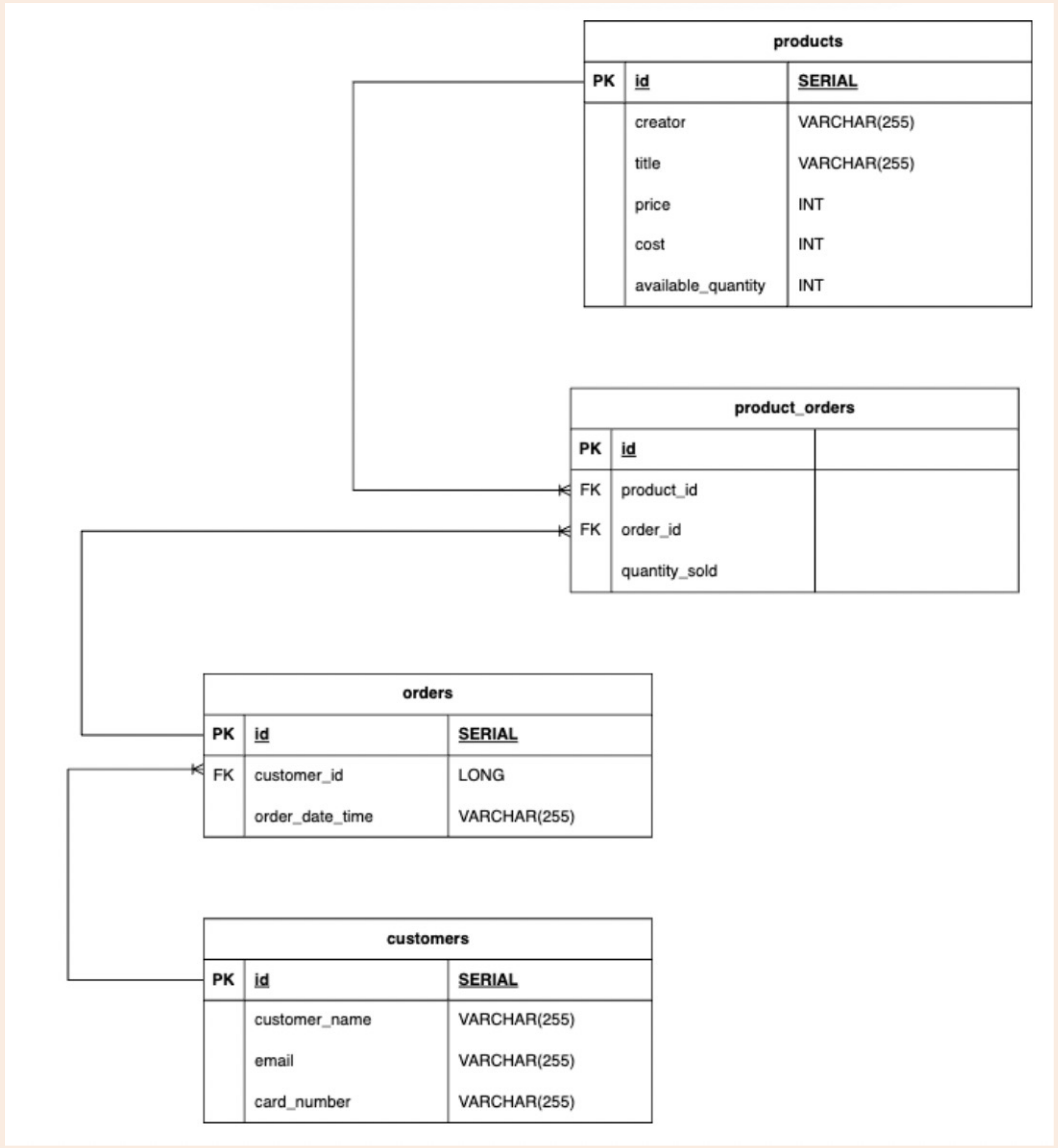


```
public Customer updateCustomer(CustomerDTO customerDTO, Long id) {
        Customer customerToUpdate = customerRepository.findById(id).get();
        customerToUpdate.setName(customerDTO.getName());
        customerToUpdate.setEmail(customerDTO.getEmail());

        //  beginner index is the total length - 4 so that we only get the last 4 digits of any card length as beginning index would start at 4th character before the last

        String cardNumber= customerDTO.getCardNumber();
        String lastFourDigitsCN = cardNumber.substring(cardNumber.length() - 4);
        customerToUpdate.setCardNumber(lastFourDigitsCN);   //  sets the last 4 digits of card number

        customerRepository.save(customerToUpdate);
        return customerToUpdate;
    }
```

LINKING OUR TABLES



```java
  public Orders addOrderToCustomer(Long customerId) {
        Orders orderToLink = new Orders(); //create a new order


        Customer customer = customerRepository.findById(customerId).get(); // link customer to its order
        orderToLink.setCustomer(customer);
        orderToLink.setOrderDateTime(LocalDateTime.now()); // time the customer will have placed the order
        ordersRepository.save(orderToLink); // save it to database
        return orderToLink;
    }

    public ProductsOrders addProductToProdOrders(Long productId, int quantitySold) {
//          ProductsOrders prodOrdersToLink = productsOrdersRepository.findById(prodOrderId).get(); //1st order // retrieves the productsOrder we want

        ProductsOrders prodOrdersToLink = new ProductsOrders();
        Product product = productRepository.findById(productId).get();
        prodOrdersToLink.setProduct(product);
        prodOrdersToLink.setQuantitySold(quantitySold);
        productsOrdersRepository.save(prodOrdersToLink); // save it to database
        return prodOrdersToLink;
    }

    public ProductsOrders addOrdersToProdOrders(Long orderId, Long productOrderId) {
        ProductsOrders prodOrdersToLink = productsOrdersRepository.findById(productOrderId).get(); //1st order // retrieves the productsOrder we want
        Orders orders = ordersRepository.findById(orderId).get(); // link customer to its order
        prodOrdersToLink.setOrders(orders);
        productsOrdersRepository.save(prodOrdersToLink); // save it to database
        return prodOrdersToLink;
    }

```



OUR ACHIEVEMENTS

1

COMMUNICATION

Open and honest communication about feelings and commitments. Pulled everyone's strengths together

2

JOINT OWNERSHIP

Moving forward as a team so that we all felt equally responsible. Coding together with everyone's support

3

'LEARNING EXPERIENCE'

Trying something new and not cracking down under the pressure, with a positive reception to each challenge

4

PROGRESS

Finishing our MVP and extensions, understanding more about the front-end/back-end split

NEXT TIME WE WOULD...

RESEARCH	• Spend time researching and comparing different possible solutions whilst analysing efficiency, stability and security
DIAGRAMS	• Create more detailed diagrams showing our CRUD routes and controller, repository and service classes
RE-FRAME GROUP OBJECTIVES	• Clarfiy the distinction between our MVP and extension as our project evolved and as we faced new challenges