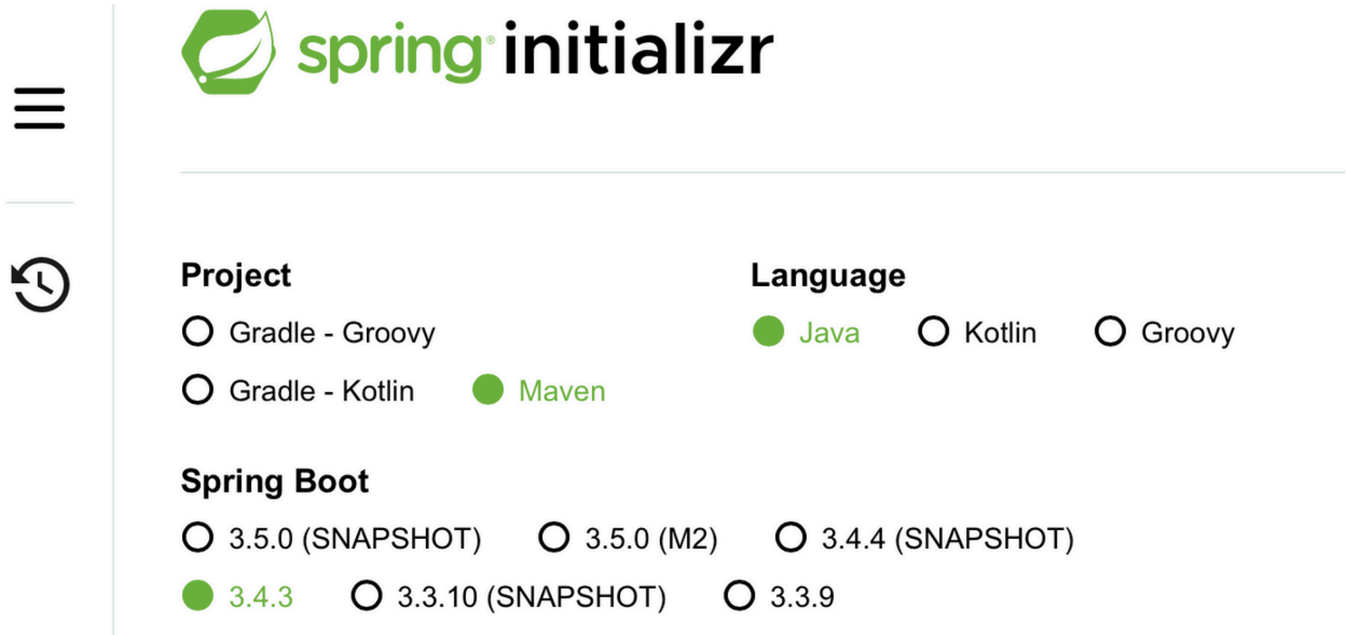
DESIGNING THE ARCHITECTURE

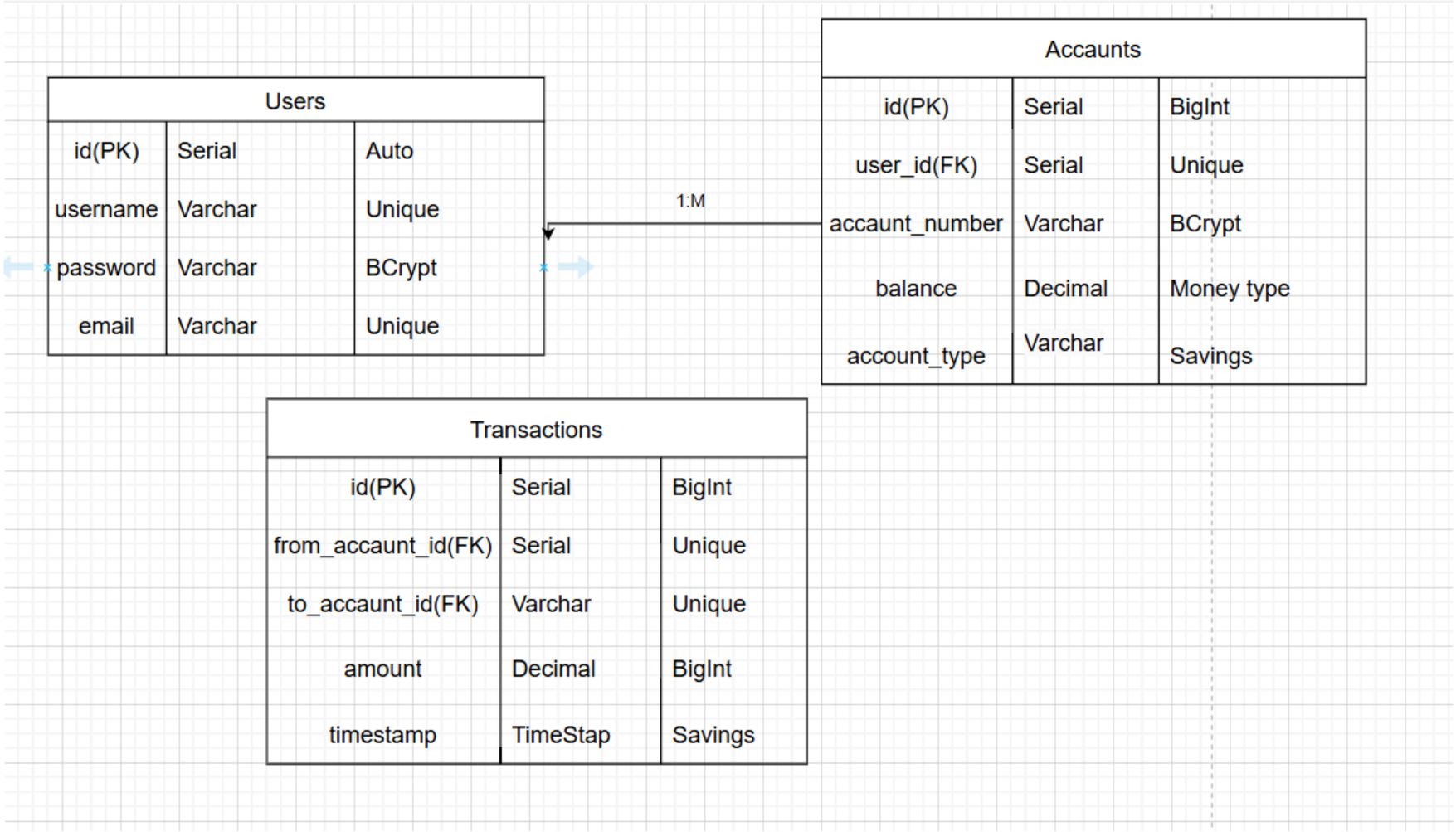
PRESENTED BY READML GROUP

Technology Stack

Layer	Technology
Backend	Java, Spring Boot
Database	PostgreSQL, JPA
Security	Spring Security, JWT
API	RESTful, Swagger
Build Tools	Maven
Deployment	Docker (если есть), Localhost



Our ER-DIAGRAMM



users ↔ accounts: one-to-many relationship (one user — many accounts)

accounts ↔ transactions: each transaction refers to two accounts (sender and recipient

List of API Endpoints

1. Authentication

GET /login

Returns the login page (Thymeleaf).

POST /login

Processes the login form (handled by Spring Security).

GET /register

Registration page (Thymeleaf).

POST /register

Creates a new user (via JSON or form submission).

2. Users (User)

GET /users

Returns a list of all users (JSON).

GET /users/{id}

Returns a user by ID.

POST /users

Creates a user (JSON).

PUT /users/{id}

Updates an existing user (JSON).

DELETE /users/{id}

Deletes a user by ID.

3. Accounts (Account)

GET /accounts

Returns a list of all accounts.

GET /accounts/{id}

Returns details for a specific account.

POST /accounts

Creates a new account.

PUT /accounts/{id}

Updates an account (if necessary).

DELETE /accounts/{id}

Deletes an account (optional).

Brief Overview of the Architecture (MVC)

Controller

Receives HTTP requests (REST endpoints or Thymeleaf pages).
Calls the corresponding services.
Returns either JSON responses or rendered HTML templates.

Repository (DAO)

Interfaces that extend Spring Data JPA (e.g., UserRepository, AccountRepository, TransactionRepository).
Responsible for CRUD operations on the database (create, read, update, delete).

Service

Contains the business logic (e.g., checking the account balance before a transaction).
Calls repositories for data storage and retrieval.
May invoke other services (e.g., an EmailService) if needed.

Model (Entity)

Classes such as User, Account, and Transaction annotated with @Entity.
Fields correspond to columns in the database (e.g., id, username, balance, etc.).

Mockups

```java
public class AccountController {

    @Autowired
    private AccountService accountService;

    @GetMapping
    public List<Account> getAllAccounts() { return accountService.getAllAccounts(); }

    @PostMapping
    public ResponseEntity<Account> createAccount(@RequestBody Account account) {
        Account newAccount = accountService.createAccount(account);
        return ResponseEntity.ok(newAccount);
    }
}
```

```java
public class RegistrationController {
    private final UserService userService;  2 usages

    public RegistrationController(UserService userService) { this.userS

    @GetMapping("/register")
    public String showRegistrationForm(Model model) {
        System.out.println("Открывается страница регистрации");
        model.addAttribute( attributeName: "user", new User());
        return "register";
    }


    @PostMapping("/register")
    public String processRegistration(@ModelAttribute("user") User user) {
        userService.createUser(user);
        return "redirect:/login";
    }
```

```java
public class DashboardController {

    @GetMapping("/dashboard")
    public String dashboard(@AuthenticationPrincipal org.springframework.secur
                            Model model) {
        if (user != null) {
            model.addAttribute( attributeName: "username", user.getUsername());
        } else {
            model.addAttribute( attributeName: "username",  attributeValue: "Guest");
        }
        return "dashboard";
    }
}
```

Mockups

```java
public class TransactionController {
    @Autowired
    private TransactionService transactionService;

    @GetMapping
    public List<Transaction> getAllTransactions() { return transactionService.getAllTransactions(); }

    @GetMapping("/{id}")
    public Optional<Transaction> getTransactionById(@PathVariable Long id) {
        return transactionService.getTransactionById(id);
    }

    @PostMapping
    public Transaction createTransaction(@RequestBody Transaction transaction) {
        return transactionService.createTransaction(transaction);
```

```java
public class Account {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne
    @JoinColumn(name = "user_id", nullable = false)
    private User user;

    @Column(nullable = false, unique = true)
    private String accountNumber;

    @Column(nullable = false)
    private Double balance;

    @Column(nullable = false)
    private String accountType; // "CHECKING" или "SAVINGS"

    @OneToMany(mappedBy = "fromAccount", cascade = CascadeType.ALL)
    private List<Transaction> sentTransactions;
```

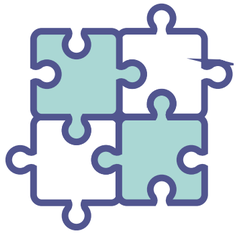
```java
public class UserController {
    @Autowired
    private UserService userService;
    @GetMapping
    public List<User> getAllUsers() { return userService.getAllUsers(); }

    // Получить пользователя по ID
    @GetMapping("/{id}")
    public ResponseEntity<User> getUserById(@PathVariable Long id) {
        Optional<User> user = userService.getUserById(id);
        return user.map(ResponseEntity::ok)
                .orElseGet(() -> ResponseEntity.notFound().build());
    }
    // Создать нового пользователя
    @PostMapping
    public ResponseEntity<User> createUser(@RequestBody User user) {
        User newUser = userService.createUser(user);
        return ResponseEntity.ok(newUser);
    }
```

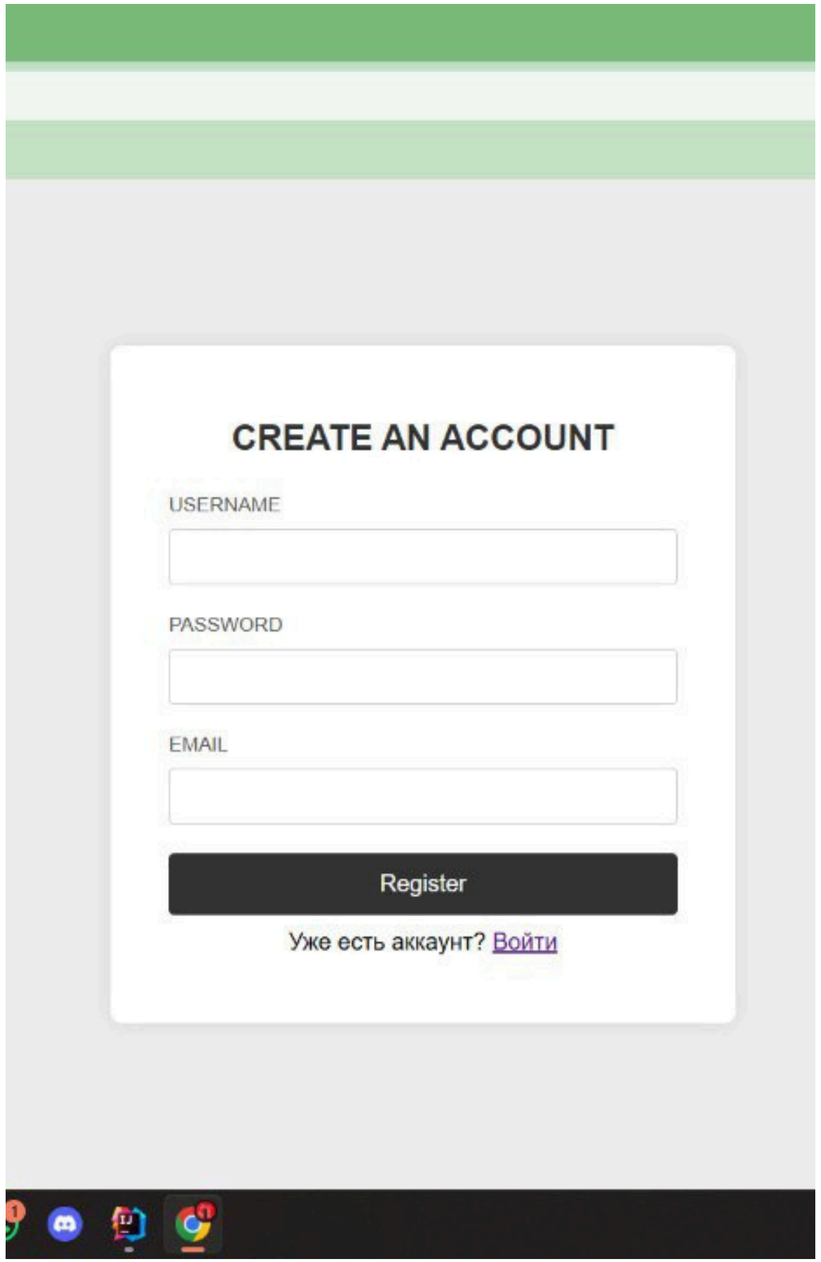
Mockups

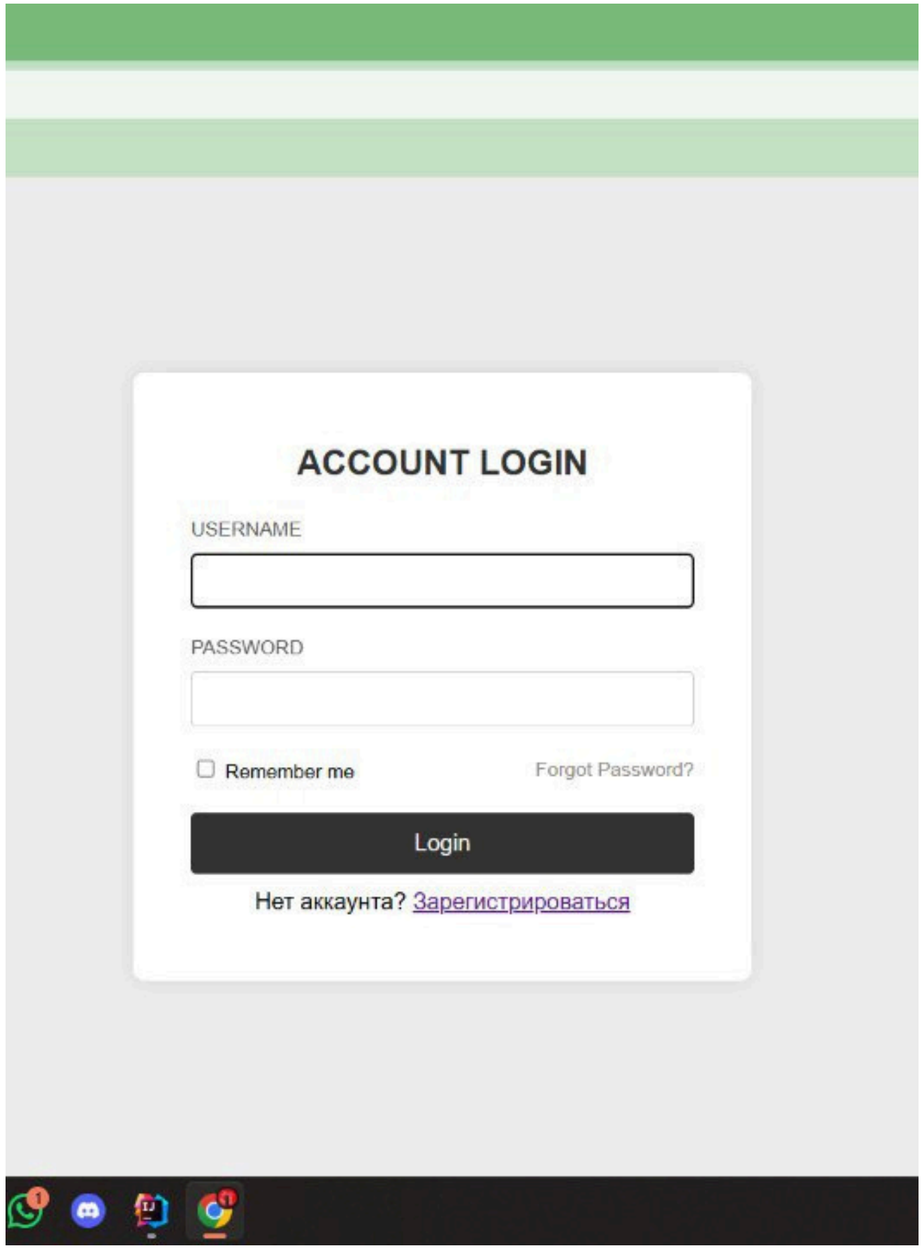
```
<body>
<div class="login-wrapper">
    <div class="login-box">
        <h2>ACCOUNT LOGIN</h2>
        <div th:if="${param.error}" class="alert error">
            Неверные учетные данные пользователя
        </div>
        <form th:action="@{/login}" method="post">
            <div class="form-group">
                <label for="username">USERNAME</label>
                <input type="text" name="username" id="username" required cla
            </div>
            <div class="form-group">
                <label for="password">PASSWORD</label>
                <input type="password" name="password" id="password" required
            </div>

            <div class="options">
                <label>
                    <input type="checkbox" name="remember-me"/> Remember me
                </label>
                <a href="#" class="forgot">Forgot Password?</a>
```

```
<body>
<div class="login-wrapper">
  <div class="login-box">
    <h2>CREATE AN ACCOUNT</h2>
    <form th:action="@{/register}" th:object="${user}" method="post">
      <div class="form-group">
        <label for="username">USERNAME</label>
        <input type="text" th:field="*{username}" id="username" required class="form-control"/>
      </div>
      <div class="form-group">
        <label for="password">PASSWORD</label>
        <input type="password" th:field="*{password}" id="password" required class="form-control"/>
      </div>
      <div class="form-group">
        <label for="email">EMAIL</label>
        <input type="email" th:field="*{email}" id="email" required class="form-control"/>
      </div>

      <button type="submit" class="btn">Register</button>
    </form>
    <p style="margin-top: 10px; text-align: center;">
      Уже есть аккаунт? <a th:href="@{/login}">Войти</a>
```

RESULT




CREATE AN ACCOUNT
USERNAME
PASSWORD
EMAIL
Register
Уже есть аккаунт? Войти


ACCOUNT LOGIN
USERNAME
PASSWORD
Remember me
Forgot Password?
Login
Нет аккаунта? Зарегистрироваться

Dashboard

localhost:8080/dashboard

Яндекс Gmail Перевести Все закладки

User icon

Hello, Ruslanbek Tolametov!

- Home
- Lorem
- Ipsum
- Elit
- Logout

User Panel

Пример простой панели пользователя

28%	80%	1000₸
Progress	Another Stats	Balance

Chart 1

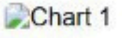
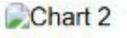


Chart 2



Поиск

8°C Cloudy

ENG

12:20:26 07.03.2025

Thank you very much!

PRESENTED BY READML GROUP