

**Design document**

**Author** : Andrii Matviienko

**Version**: 1.0

**Date**: 28.03.2025

[**Context :** 2](#_Toc198831504)

[**C1:** 3](#_Toc198831505)

[**Software Principles Applied:** 4](#_Toc198831506)

[**С2:** 5](#_Toc198831507)

[**Software Principles Applied:** 6](#_Toc198831508)

[**С3:** 6](#_Toc198831509)

[**Software Principles Applied:** 7](#_Toc198831510)

[**Frontend:** 8](#_Toc198831511)

[**Main page:** 8](#_Toc198831512)

[**Login page:** 9](#_Toc198831513)

[**Register page:** 10](#_Toc198831514)

# **Context :**

This design document defines the architecture of WiredSpace, explains technology choices, and provides C4 Model diagrams. It ensures clarity, consistency, and a shared understanding among developers and stakeholders.

# **C1:**

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

This **System Context (C1) diagram** shows how different entities interact with **WiredSpace**:

1. **Users** create, consume, and interact with social media content through the system.
2. **Admins** manage, moderate, and configure the platform.
3. **WiredSpace** handles API requests and processes interactions.
4. **Email System** sends confirmation emails and notifications to users.

## **Software Principles Applied:**

* **KISS (Keep It Simple, Stupid)** – The architecture ensures simplicity by clearly defining roles for each component, reducing unnecessary complexity.
* **YAGNI (You Ain’t Gonna Need It)** – Features that are not immediately required are not included to keep the system lightweight.

# **С2:**

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

The **Container Diagram (C2)** illustrates the high-level architecture of WiredSpace. The **Frontend (React)** provides the user interface and interaction layer. The **Backend (Spring Boot)** processes business logic and manages API requests. The **Database (MySQL)** stores user data and content. The **Email System** sends notifications and confirmations via SMTP. **Users** and **Admins** interact with the system through the frontend.

## **Software Principles Applied:**

* **SOLID**
  + **Single Responsibility Principle (SRP):** Each service (frontend, backend, database, email system) has a well-defined responsibility.
  + **Dependency Inversion Principle (DIP):** The backend communicates with the database via repositories, avoiding direct dependencies.
* **DRY (Don't Repeat Yourself):** Common functionalities such as authentication and notifications are centralized to avoid redundant code.

# **С3:**

Изображение выглядит как текст, снимок экрана, Самоклеющийся листок, желтый

Контент, сгенерированный ИИ, может содержать ошибки.

С3 diagram represents the high-level architecture of the **WiredSpace** API application.

1. The **Frontend** (React-based) allows users to interact with the system by sending API requests.
2. The **API Application** (Spring Boot) consists of controllers, services, and repositories that handle business logic and data persistence.
3. The **Post Controller** and **User Controller** manage HTTP requests, delegating logic execution to their respective services.
4. The **Post Service** and **User Service** execute business logic and interact with repositories for data persistence.
5. The **Post Repository** and **User Repository** handle database operations using MySQL.
6. The **Email Service** manages communication with the **Email System**, which sends emails to users.
7. The **Database (MySQL)** stores user-generated content and system information.

## **Software Principles Applied:**

* **SOLID**
  + **Open/Closed Principle (OCP):** Services are designed to be extensible without modifying existing code.
  + **Liskov Substitution Principle (LSP):** Repositories and services follow interfaces, ensuring flexibility in implementation.
* **DRY:** Reusable components (e.g., shared services for logging and notifications) minimize code duplication.
* **KISS:** The API architecture follows clear separation of concerns, making it easy to understand and maintain.
* **YAGNI:** Advanced optimizations (e.g., caching, load balancing) are deferred until necessary to avoid premature complexity.

# **Frontend:**

## **Main page:**

**Изображение выглядит как текст, снимок экрана, Шрифт, логотип

Контент, сгенерированный ИИ, может содержать ошибки.**

**The Main Page serves as the landing page for WiredSpace and acts as the entry point for both new and returning users.**

* **Functionality:**

Displays a welcome message and a brief introduction to the platform.

Provides navigation buttons to the Login and Register pages.

Optionally showcases public posts or content previews for non-authenticated users.

* **Purpose:**

Attract new users and guide them toward account creation or login.

Provide quick access to the core features of the platform.

* **Implementation:**

Built with React and styled for responsiveness.

Integrates with routing to transition to login/register or authenticated dashboard.

## **Login page:**

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

The Login Page allows existing users to authenticate and access their personalized dashboard within WiredSpace.

* **Functionality:**

Accepts user credentials (email/username and password).

Displays a link to the Register Page for new users.

* **Security:**

Sends authentication requests to the backend via secure API endpoints.

Handles token/session storage for future requests.

* **Implementation:**

Uses React hooks for form state and API handling.

Authentication logic is separated via authService and authApi modules.

## **Register page:**

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

The Register Page allows new users to create an account and join the WiredSpace community.

* **Functionality:**

Accepts user input (username, email, password, etc.).

Includes client-side validation and feedback for form fields.

Displays success or error messages based on server response.

* **Security & Usability:**

Sends registration data securely to the backend.

Upon success, either redirects to login or logs the user in automatically.

* **Implementation:**

Built using React with modular form components.

Registration logic is encapsulated within authService and authApi.