ЛАБОРАТОРНАЯ РАБОТА №2



Курс «Технологии разработки программного обеспечения»

Тема: Проектирование и макетирование программного продукта.

Цель: Научиться проектировать простейшие системы и составлять документацию по проектированию программного продукта.

Задание:

1. Спроектировать общую архитектуру приложения на основе спецификаций требований, составленных в лабораторной работе №1.

**Архитектура**

**Архитектурный стиль:** Многослойная архитектура (представление, бизнес-логика, уровень данных) с RESTful API для взаимодействия между клиентской и серверной частями.

Компоненты:

**Клиентская часть (Frontend):**

Модуль отображения каталога книг.

Модуль чтения книг.

Модуль авторизации и управления аккаунтом.

Модуль читательского дневника.

Модуль просмотра комментариев.

Модуль чата для книг (доступен зарегистрированным пользователям).

**Серверная часть (Backend):**

Модуль аутентификации и авторизации.

Модуль управления каталогом книг.

Модуль управления комментариями к книгам.

Модуль управления читательским дневником.

Модуль чата для книг (реализация WebSocket или аналогичная технология).

Модуль доступа к данным (работа с базой данных).

**База данных (Data Layer):**

Хранит информацию о пользователях, книгах, комментариях, записях дневника и сообщениях чата.

1. Разработать макет интерфейса программного продукта для демонстрации заказчику. Проанализировать и учесть его замечания и пожелания.

* При открытии приложения будет появляться окно загрузки с логотипом (рис 1). Светло-голубой круг будет немного уменьшаться в размере и возвращаться (рис1 и рис2).



Рисунок 1



Рисунок 2

* После будет появляться окно регистрации и входа (рис 3)

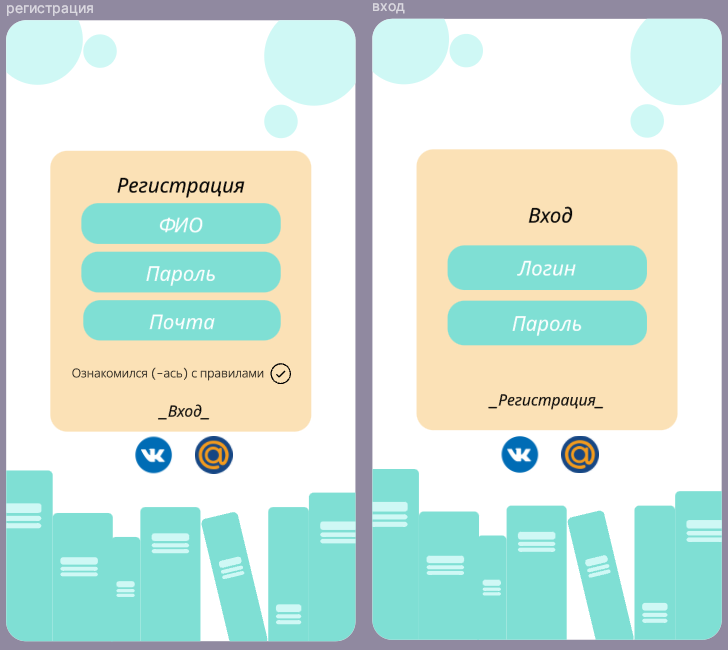


Рисунок 3

Между ними можно переключаться между ними с помощью «\_Вход\_» и «\_Регистрация\_» (рис 3). При «Регистрации», пользователь соглашается с правилами, установленными администратором, и контролируемые модераторами.

* После любой из этих махинаций, пользователь переходит на главную страницу

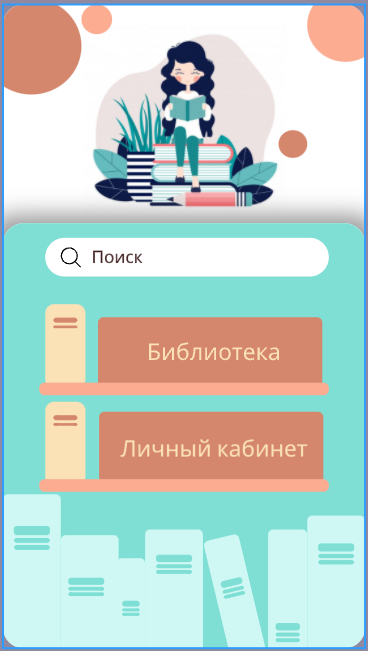


Рисунок 4

На ней можно сразу искать книги по названию или автору, или просто зайти в библиотеку (рис 5) в которой будет лента из самых популярных произведений этой недели (промежуток времени можно менять на «день», «месяц», «год», нажав на кружочек с гайкой в правом верхнем углу. По умолчанию стоит «неделя»). В левом верхнем углу можно было нажать на стрелку и вернуться обратно на «главный экран».

Так же тут есть кнопка на переход в личный кабинет (рис 4). Если пользователь не зарегистрирован, то всплывает предупреждающее окно (рис 6). При нажатии «да» мы возвращаемся на окна регистрации и входа (рис3). При выборе «нет», всплывающее окно уходит, и пользователь просто остается на «главном экране».



Рисунок 5

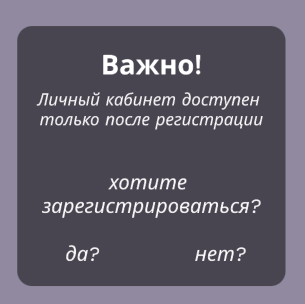


Рисунок 6

* В случае выполнения условия «регистрация», пользватель переходит в свой личный кабинет (рис 7).

Тут отображается личная информация пользователя:

1. Никнейм пользователя
2. установленное им изображение
3. статус. (слово «Статус» пропадает при его замене с системного на пользовательский.
4. Уровень (в соответствии с определенной системой, он повышается (например: за посещение приложения больше 2 дней подряд, прочтении +1 книги или активном участии в чате, комментариях) или понижается (например: при не соблюдении правил, с которыми пользователь будет соглашается при регистрации)
5. В правом верхнем углу карандаш, при нажатии, дает возможность редактировать пункты 1,2,3.
6. В левом верхнем углу стрелочка переносит обратно на «главный экран»
7. В правом нижнем углу книга с узором шестеренки называется «настройки». (Там можно редактировать: уведомления, размер шрифта и темную и светлую тему. После пару обновлений возможны добавления новых пунктов типа: выбрать цветовую палитру приложения)
8. В левом нижнем углу, книга с названием «жалобы». Там пользователь может пожать жалобу на цензуру или плагиат. (открывается после достижения Lv.10 пользователя)
9. Изображение полки с книгами, представляют собой кнопку для перехода в «читательский дневник (рис 8)



Рисунок 7

* В читательском дневнике (рис8) отображаются добавленные пользователем книги. Отображается обложка книги, ее название и оценка пользователя (10/10). Так же быстрый переход в чат произведения (рис9).

В верхнем правом углу волшебная палочка отвечает за редактирование дневника (добавление и удаление книг)

В левом верхнем углу стрелочка переносит обратно на «читательский дневник»

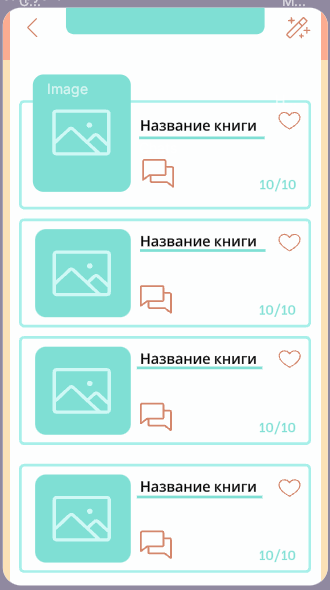


Рисунок 8

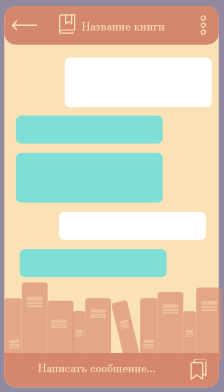


Рисунок 9

* В чате можно писать сообщения. Отвечать на сообщения других пользователей. Вставлять цитаты с помощью флажков в правом нижнем углу. (чат предназначен для обсуждения конкретного произведения)
* Нажав на любую из книг (не важно, где в библиотеке или дневнике), пользователь переходит в превью Книги (рис 10).

На превью есть:

1. Обложка книги
2. Название книги (так же на английском и оригинальном языке)
3. Издательство
4. Пролог
5. В левом верхнем углу стрелочка переносит обратно на предыдущую страницу
6. В правом верхнем углу карандашик дает возможность добавить/удалить книгу в читательском дневнике.
7. Сердечки позволяют кратко отреагировать на книгу
8. В левом нижнем углу переход в чат произведения (рис9)
9. В правом нижнем углу отображены черточки позволяющие перейти в комментарии (рис11).



Рисунок 10

* В комментариях нельзя отвечать на комментарии других пользователей. Можно читать их, писать свои и оставлять реакции в виде «лайка» и «дизлайка», означающие положительный и отрицательный факт.   
  Сверху стрелочка позволяет вернуться к «превью» книги. (рис10)

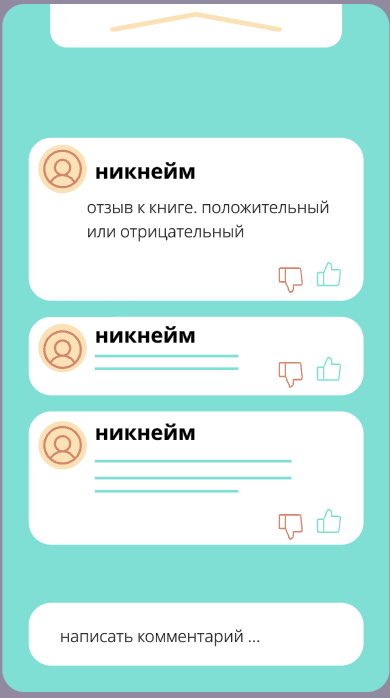


Рисунок 11

1. Оформить разделы документа SDD, относящиеся к предварительному проектированию (приложение 2.1).

**1.Введение**

* **Название приложения:** Онлайн-библиотека “Читай-ка”
* **Цель:** предоставить пользователям возможность читать книги онлайн, вести читательский дневник, участвовать в обсуждениях книг и делиться впечатлениями.
* **Целевая аудитория:** любители чтения, желающие иметь доступ к книгам онлайн и общаться с другими читателями.
* **Обзор:** Приложение состоит из клиентской части (для отображения интерфейса и взаимодействия с пользователем) и серверной части (для хранения данных и обработки запросов).

**2.Компоненты и модули**

* **2.1 Серверная часть (Backend)**
  + **Аутентификация и авторизация:**
    - Предоставляет API для регистрации, входа и выхода пользователей.
    - Генерирует и управляет токенами авторизации.
    - Проверяет права доступа к ресурсам (например, доступ к чату только для зарегистрированных пользователей).
  + **Управление каталогом книг:**
    - Предоставляет API для поиска, просмотра книг.
    - Позволяет добавлять, изменять и удалять книги (только для администраторов).
  + **Управление комментариями:**
    - Предоставляет API для получения списка комментариев к книге.
    - Позволяет добавлять новые комментарии к книге (для всех пользователей).
  + **Управление читательским дневником:**
    - Предоставляет API для добавления, просмотра, изменения и удаления записей дневника.
  + **Чат для книг:**
    - Использует WebSocket (или аналогичную технологию) для обмена сообщениями в реальном времени.
    - Предоставляет API для получения истории сообщений.
    - Ограничивает доступ к чату только для зарегистрированных пользователей.
  + **Доступ к данным:**
    - Интерфейс для взаимодействия с базой данных.
    - Выполняет операции CRUD (создание, чтение, обновление, удаление) для всех сущностей.
* **2.2 Клиентская часть (Frontend)**
  + **Каталог книг:**
    - Отображает список книг (с разбивкой на страницы, сортировкой и фильтрацией).
    - Предоставляет возможность просмотра детальной информации о книге.
    - Предоставляет возможность поиска книг.
  + **Чтение книг:**
    - Отображает текст книги.
    - Позволяет перелистывать страницы.
  + **Авторизация и управление аккаунтом:**
    - Предоставляет формы для регистрации и входа.
    - Позволяет просматривать и редактировать профиль пользователя.
  + **Читательский дневник:**
    - Отображает список записей дневника.
    - Предоставляет формы для добавления, изменения и удаления записей.
  + **Просмотр комментариев:**
    - Отображает список комментариев к книге.
    - Позволяет добавлять новые комментарии.
  + **Чат для книг:**
    - Отображает чат для обсуждения книги.
    - Позволяет отправлять сообщения в реальном времени.

**3.База данных**

* **Схема (пример):**
  + **Пользователи:** (идентификатор пользователя, имя пользователя, адрес электронной почты, хэш пароля, дата регистрации)
  + **Книги:** (идентификатор книги, название, автор, жанр, год публикации, изображение обложки, содержание)
  + **Комментарии:** (идентификатор комментария, идентификатор пользователя, идентификатор книги, содержимое, дата)
  + **ReadingLog:** (logEntryId, userId, bookId, дата, текст, рейтинг)
  + **Сообщения в чате:** (идентификатор сообщения, идентификатор книги, идентификатор пользователя, сообщение, дата)

**4.Безопасность**

* Использование HTTPS для шифрования трафика.
* Хранение паролей в виде хэшей.
* Проверка прав доступа к ресурсам (на основе токенов авторизации).

**5.Нефункциональные требования**

* **Производительность:** Быстрый отклик, обработка большого количества запросов.
* **Масштабируемость:** возможность добавления новых пользователей и контента без потери производительности.
* **Надежность:** Бесперебойная работа приложения.

1. Программно реализовать первую версию спроектированной системы (создать программные модули и закодировать основные классы / структуры и связи между ними).

**1. Упрощенные базовые классы:**

#include <iostream>

#include <string>

#include <vector>

#include <ctime>

#include <map>

*// Класс для пользователя*

class User {

public:

int id; *// Идентификатор пользователя*

std::string name; *// Имя пользователя*

std::string email; *// Электронная почта*

std::string password; *// Пароль (в реальной жизни нужно хранить хэш!)*

*// Конструктор - метод для создания объекта User*

User(int id, std::string name, std::string email, std::string password) {

this->id = id;

this->name = name;

this->email = email;

this->password = password;

}

};

*// Класс для книги*

class Book {

public:

int id; *// Идентификатор книги*

std::string title; *// Название книги*

std::string author; *// Автор книги*

*// Конструктор - метод для создания объекта Book*

Book(int id, std::string title, std::string author) {

this->id = id;

this->title = title;

this->author = author;

}

};

*// Класс для комментария*

class Comment {

public:

int id; *// Идентификатор комментария*

int userId; *// Идентификатор пользователя, оставившего комментарий*

int bookId; *// Идентификатор книги, к которой относится комментарий*

std::string text; *// Текст комментария*

*// Конструктор - метод для создания объекта Comment*

Comment(int id, int userId, int bookId, std::string text) {

this->id = id;

this->userId = userId;

this->bookId = bookId;

this->text = text;

}

};

**2. Упрощенные классы управления:**

*// Класс для управления пользователями*

class UserManager {

public:

std::map<int, User> users; *// Хранит пользователей*

*// Метод для добавления пользователя*

void addUser(User user) {

users[user.id] = user; *// Добавляем пользователя по id*

}

*// Метод для аутентификации пользователя*

User\* authenticate(std::string email, std::string password) {

*// Проходим по всем пользователям*

for (auto const& [id, user] : users) {

if (user.email == email && user.password == password) {

return &users[id]; *// Возвращаем пользователя, если нашли совпадение*

}

}

return nullptr; *// Если не нашли - возвращаем nullptr*

}

};

*// Класс для управления книгами*

class BookCatalog {

public:

std::map<int, Book> books; *// Хранит книги*

*// Метод для добавления книги*

void addBook(Book book) {

books[book.id] = book; *// Добавляем книгу по id*

}

*// Метод для поиска книг по названию (простой поиск)*

std::vector<Book> search(std::string query) {

std::vector<Book> result;

for(auto const& [id, book] : books){

if(book.title.find(query) != std::string::npos){

result.push\_back(book);

}

}

return result;

}

};

*// Класс для управления комментариями*

class CommentManager {

public:

std::map<int, Comment> comments; *// Хранит комментарии*

*// Метод для добавления комментария*

void addComment(Comment comment) {

comments[comment.id] = comment; *// Добавляем комментарий по id*

}

*// Метод для получения комментариев к книге*

std::vector<Comment> getCommentsByBook(int bookId) {

std::vector<Comment> bookComments;

for(auto const& [id, comment] : comments){

if (comment.bookId == bookId) {

bookComments.push\_back(comment);

}

}

return bookComments;

}

};

**3. Пример использования:**

int main() {

*// Создаем менеджеров*

UserManager userManager;

BookCatalog bookCatalog;

CommentManager commentManager;

*// Создаем пользователей*

User user1(1, "Alice", "alice@example.com", "12345");

User user2(2, "Bob", "bob@example.com", "67890");

userManager.addUser(user1);

userManager.addUser(user2);

*// Создаем книги*

Book book1(1, "The Hobbit", "J.R.R. Tolkien");

Book book2(2, "1984", "George Orwell");

bookCatalog.addBook(book1);

bookCatalog.addBook(book2);

*// Создаем комментарии*

Comment comment1(1, user1.id, book1.id, "Great book!");

Comment comment2(2, user2.id, book1.id, "I agree!");

commentManager.addComment(comment1);

commentManager.addComment(comment2);

*// Поиск книг*

std::vector<Book> foundBooks = bookCatalog.search("Hobbit");

std::cout << "Found books: " << std::endl;

for (Book book : foundBooks) {

std::cout << book.title << " by " << book.author << std::endl;

}

*// Получение комментариев к книге*

std::vector<Comment> bookComments = commentManager.getCommentsByBook(book1.id);

std::cout << "Comments for book 1: " << std::endl;

for(Comment comment : bookComments) {

std::cout << comment.text << std::endl;

}

*// Проверка аутентификации*

User\* authUser = userManager.authenticate("alice@example.com", "12345");

if(authUser != nullptr) {

std::cout << "User " << authUser->name << " successfully authenticated!" << std::endl;

} else {

std::cout << "Invalid credentials." << std::endl;

}

return 0;

}

