

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»
Профиль подготовки: «Программное обеспечение средств вычислительной
техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

**ИНФОРМАЦИОННО-СПРАВОЧНАЯ СИСТЕМА
СТАТЕЙ ЭНЦИКЛОПЕДИЧЕСКОГО ХАРАКТЕРА**

Студент

30.05.2024

(подпись, дата)

Макарова Анна Юрьевна
(ФИО полностью)

Руководитель

30.05.2024

(подпись, дата)

М.П. Синев
(фамилия, инициалы)

Нормоконтролёр

30.05.2024

(подпись, дата)

А.С. Бычков
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 27.05.2024 г. № 11)

Заведующий кафедрой

(подпись)

М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой
04.06.2024 № 4

Отлично

(протокол заседания ГЭК от

Секретарь ГЭК

(подпись)

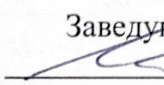
Е.И. Гурин
(фамилия, инициалы)

Пенза 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
 М.А. Митрохин
«19» апреля 2024 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Макарова Анна Юрьевна гр. 20ВВП1 факультета вычислительной техники
2. Тема работы Информационно-справочная система статей энциклопедического характера

Тема утверждена приказом ПГУ № 405/0 от "19" апреля 2024 г.

3. Руководитель бакалаврской работы Синев М.П., к.т.н., доцент кафедры ВТ
4. Задание на работу (назначение разработки, исходные данные и т.п.)
Разработать информационную систему, предназначенную для просмотра, хранения и изменения базы текстовых статей. Провести тестирование серверной и клиентской частей системы.
5. Перечень подлежащих разработке вопросов
Требуется произвести анализ предметной области, аналогов, выявить целевые сценарии использования системы. По результатам анализа необходимо спроектировать архитектуру базы данных и REST API. Также требуется сравнить достоинства и недостатки современных средств разработки, провести анализ и выбрать наиболее подходящие для решаемой задачи. Разработать серверное приложение, связанное с базой данных, и мобильный клиент.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	17	19.04.2024 – 21.04.2024	
2.	Анализ аналогов	19	21.04.2024 – 23.04.2024	
3.	Постановка задачи	23	23.04.2024 – 26.04.2024	
4.	Разработка архитектуры приложения	33	27.04.2024 – 30.04.2024	
5.	Выбор инструментов разработки	10	01.05.2024 – 02.05.2024	
6.	Разработка серверной части	56	02.05.2024 – 09.05.2024	
7.	Разработка мобильного клиента	66	10.05.2024 – 19.05.2024	
8.	Отладка и тестирование системы	43	19.05.2024 – 24.05.2024	
9.	Оформление пояснительной записки	57	24.05.2024 – 31.05.2024	

Дата выдачи задания "19" апреля 2024г.

Руководитель бакалаврской работы  19.04.2024 Синев М.П.
(подпись, дата)

Задание к исполнению принял студент  19.04.2024 Макарова А.Ю.
(подпись, дата)

Работу к защите допустить
Декан факультета  05.06.2024 Фионова Л.Р.
(подпись, дата)

Реферат

Бакалаврская работа содержит 162 листа, 33 рисунка, 38 таблиц, 10 источников и 4 приложения

СПРАВОЧНАЯ СИСТЕМА, БАЗА ДАННЫХ, СУБД, МОБИЛЬНАЯ
РАЗРАБОТКА, JAVA, KOTLIN, POSTGRESQL, SQLITE, REST, SPRING
FRAMEWORK, ANDROID

Целью работы является разработка системы, предназначенной для хранения и создания статей в формате справочника-энциклопедии.

Для этого необходимо разработать удалённую базу данных, используемую для хранения глобальной базы статей, мобильный клиент – устройство, предлагаемое конечным пользователям в качестве основного инструмента взаимодействия с системой, а также веб-сервер, позволяющий организовать доступ к удаленной базе данных через REST API.

В результате проделанной работы проведен анализ требований к системе, выполнено проектирование и разработка системы, состоящей из указанных компонентов и выполняющей функции, описанные при анализе.

Спроектировано REST API, реализованное на стороне серверного десктопного приложения, связанного с базой данных.

Реализовано мобильное приложение для устройств на операционной системе Android, позволяющее пользователям просматривать, дополнять и редактировать базу статей, ориентированное преимущественно на работу в автономном режиме, но вместе с тем поддерживающее возможность синхронизации с удаленной базой данных. Выполнено тестирование всех возможностей приложения.

					ПГУ 1.090301.16.001 ПЗ				
Изм.	Лист	№ докум.	Подпись	Дата	Информационно-справочная система статей энциклопедического характера Пояснительная записка	Лит.	Лист	Листов	
Разраб.		Макарова А.Ю.		30.05.2021					
Провер.		Синев М.П.		30.05.2021			2	162	
Н. Кофр		Бычков А.С.		30.05.2021		ФВТ, каф. ВТ гр. 20ВВП1			
Утверд.		Митрохин М.А.		30.05.2021					

Содержание

Перечень сокращений и обозначений.....	4
Введение.....	5
1 Анализ темы.....	6
1.1 Анализ предметной области	7
1.2 Анализ аналогов	8
1.3 Постановка задачи.....	9
1.3.1 Функциональные требования	10
1.3.2 Нефункциональные требования	16
2 Реализация.....	18
2.1 Архитектура системы	18
2.1.1 Компоненты системы.....	18
2.1.2 Архитектура базы данных	19
2.1.3 Описание REST API.....	22
2.2 Выбор инструментов разработки	31
2.2.1 Выбор СУБД.....	31
2.2.2 Выбор инструментов разработки серверной части	32
2.2.3 Выбор инструментов разработки клиентской части	34
2.3 Разработка серверной части	35
2.4 Разработка мобильного клиента	38
3 Методы испытаний	46
3.1 Тестирование REST API.....	46
3.2 Тестирование мобильного клиента	53
3.2.1 Тест-кейс «Регистрация».....	53
3.2.2 Тест-кейс «Авторизация»	54
3.2.2 Тест-кейс «Поиск и просмотр статьи».....	55

3.2.3 Тест-кейс «Работа со списком избранных статей».....	55
3.2.6 Тест-кейс «Создание новой статьи».....	56
3.2.7 Тест-кейс «Редактирование статьи в локальной базе данных».....	57
3.2.8 Тест-кейс «Отправка статьи в глобальную базу данных»	58
3.2.9 Тест-кейс «Удаление статьи в глобальной базе данных».....	59
3.2.10 Тест-кейс «Изменение прав пользователя».....	60
3.2.11 Тест-кейс «Удаление пользователя»	61
4 Безопасность жизнедеятельности.....	62
4.1 Классификация и категорирование помещения для проведения работ	62
4.2 Санитарно-гигиеническая характеристика.....	64
4.3 Электробезопасность	66
4.4 Пожарная безопасность	67
4.5 Расчет общего искусственного освещения помещения с ПЭВМ	68
Заключение	70
Список использованных источников	71
Приложение А – Графическая часть	73
Приложение Б – Листинг	80
Приложение В – Листинг серверного приложения	95
Приложение Г – Листинг мобильного приложения	156

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

БД – база данных

СУБД – система управления базами данных

API (Application Programming Interface) – прикладной программный интерфейс

HTTP (Hypertext Transfer Protocol) – протокол передачи гипертекста

IDE (Integrated Development Enviroment) – интегрированная среда разработки

MVC («Model-View-Controller») – «Модель-Представление-Контроллер»

REST (Representational State Transfer) – передача состояния представления

SQL (Structured Query Language) – язык структурированных запросов

Введение

На протяжении многих лет люди стремятся организовать и систематизировать знания, собранные об окружающем мире. Для этой цели создаются различные справочники, базы знаний, энциклопедии, что позволяет структурировать данные и хранить их в удобном формате, позволяющем быстро ознакомиться с основными понятиями темы. Использование таких систем позволяет сохранить и задокументировать знания, уточнить определенные данные, найти сведения о нужном предмете.

Подобные справочные системы могут содержать как узкоспециализированную информацию – сведения, интересные для конкретной аудитории, так и информацию, полезную для широкого круга читателей. Часто справочная литература используется для расширения кругозора и в исследовательских целях.

В современном мире люди всё меньше обращаются к книжным энциклопедиям, предпочитая электронные аналоги.

Целью данной работы является разработка информационной системы для хранения статей в формате справочника-энциклопедии, обеспечивающей удобное управление базой статей, хранение и эффективный поиск по базе знаний. Основная идея разработки перекликается с устройством, описанным в цикле научно-фантастических романов английского писателя Дугласа Адамса «Автостопом по галактике» (The Hitchhiker's Guide to the Galaxy). Таким образом, создание аналога описанного устройства носит не только познавательный, но и развлекательный характер.

1 Анализ темы

Справочные системы предназначены для хранения и предоставления по запросу пользователей наиболее точной информации по той или иной теме. Часто можно заметить комбинирование справочных систем с поисковыми, где выборка релевантных результатов определяется по заданным ключевым словам или (при полнотекстовом поиске) части предложения.

Справочная система должна соответствовать следующим требованиям:

- полнота базы справочной информации;
- интуитивно понятный пользовательский интерфейс;
- поддержка удобного и быстрого поиска;
- бесперебойная работа в условиях эксплуатации, определенных при проектировании системы.

База знаний состоит из статей, содержание которых определяется предметной областью. Основными структурными элементами статьи являются, как правило, заголовок, отражающий основную тему, и собственно текст со справочной информацией. Могут присутствовать блоки ключевых слов, упрощающие процесс поиска релевантных результатов, ссылки на статьи по смежным темам, содержащиеся в данной справочной системе, а также ссылки на иные информационные ресурсы.

В части требований к пользовательскому интерфейсу справочных систем можно отметить необходимость учитывать формат представления данных – основная часть данных является текстом достаточно большого объема. Соответственно, необходимо обеспечить его легкое восприятие. С этой целью текст разделяется на блоки. Кроме того, в тексте также могут присутствовать части, требующие повышенного внимания пользователя, выделяемые в пользовательском интерфейсе особым образом.

Изм.	Лист	№ докум.	Подпись	Дата

База справочной информации может быть как статичной, содержащий неизменный набор данных, так и изменяемой коллективом редакторов. Во втором случае информационно-справочная система должна иметь функционал, позволяющий создавать и редактировать статьи.

Помимо редакторов, ответственных за написание статей контентом могут управлять администраторы или модераторы, главным образом осуществляющие контроль за публикуемыми материалами. Также администраторам может предоставляться возможность управления пользователями - выдача прав на редактирование базы статей или удаление из системы недобросовестных пользователей. В этом случае в систему также включается административный интерфейс, недоступный для обычных пользователей, позволяющий осуществлять выполнение операций по администрированию системы.

1.1 Анализ предметной области

Предметная область данной работы перекликается с устройством, описанным в научно-фантастическом романе "Автостопом по галактике" английского писателя Дугласа Адамса. Устройство в романе описывается как мобильный гаджет, содержащий статьи свободного содержания о каких-либо предметах или явлениях во вселенной. Статьи формируются коллективом редакторов, ведущих повествование в достаточно вольном стиле. Можно сделать вывод о том, что данный сборник статей наряду с информационной выполняет и развлекательную функцию.

Таким образом, результатом работы станет информационная система, объединяющая вокруг себя людей, заинтересованных одной и той же литературной вселенной, которая позволит создать некий аналог познавательно-развлекательного справочника в физическом варианте.

1.2 Анализ аналогов

При поиске аналогов обнаружены сведения о реализованном ранее решении в данной предметной области – веб-сайте xource.co.uk [1]. Однако, в настоящий момент данный Интернет-ресурс недоступен. Интерфейс веб-сайта показан на рисунке 1.

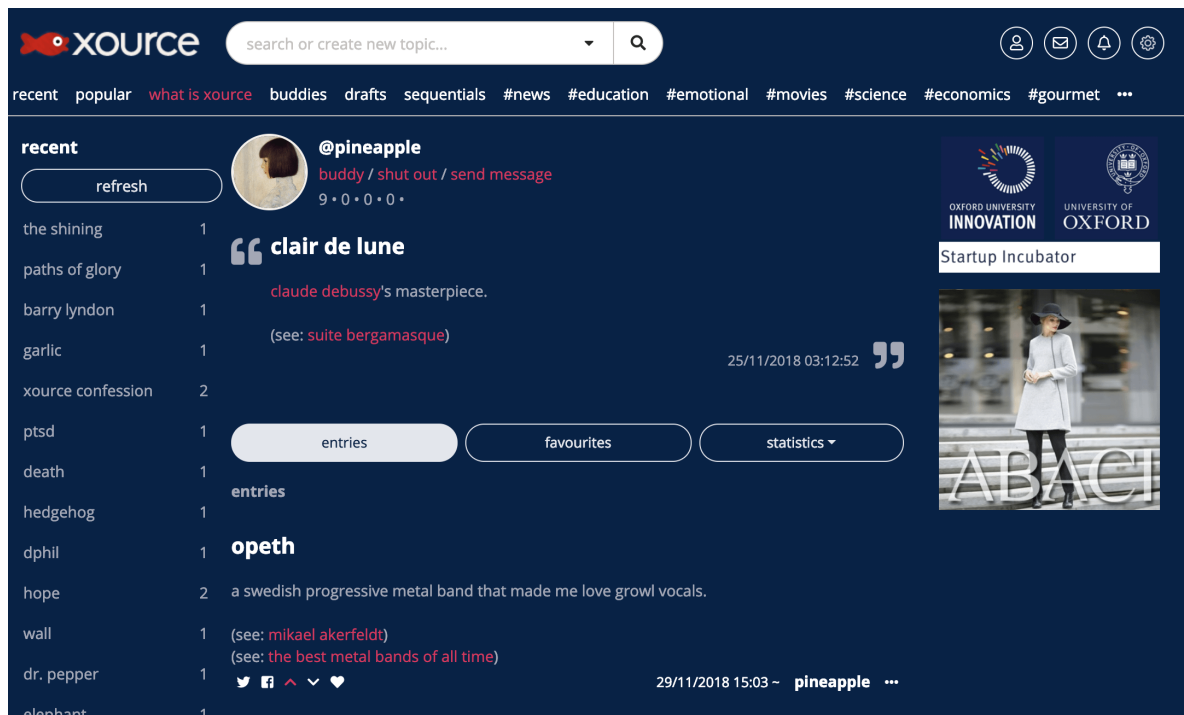


Рисунок 1 - Интерфейс веб-сайта xource

Исходя из описания продукта, реализованы были основные функции:

- регистрация и авторизация пользователей;
- создание статьи;
- просмотр статьи, поиск по базе статей.

Также можно отметить некоторые дополнительные функциональности:

- возможность поделиться статьей в социальных сетях;
- возможность выставления реакции к статье;
- поддержка списка «избранных» статей;
- поддержка списка недавно просмотренных статей;

- механизм хеш-тегов;
- добавление ссылок в текст статьи;
- особый вид статей – цитаты.

К недостаткам данного решения можно отнести формат реализации. Веб-сайт не может обеспечить доступ к базе справочной информации при отсутствии на устройстве пользователя доступа в Интернет.

Получить сведения о наличии адаптивной вёрстки под мобильные устройства невозможно в связи с недоступностью ресурса на данный момент. Однако важно отметить, что отсутствие возможности использовать справочную систему на мобильном устройстве является существенным недостатком в данной предметной области.

1.3 Постановка задачи

Требуется разработать информационно-справочную систему, основными объектами в которой являются текстовые статьи. Основные сущности, входящие в систему, описаны ниже.

Статья – минимальная структурная единица информационной базы справочной системы. Обязательно содержит заголовок, отражающий суть описываемого объекта/явления, и, собственно, текст описания. Может содержать небольшие блоки текста, требующие особого внимания пользователя и выделяемые специальным образом в пользовательском интерфейсе.

Пользователи – лица, эксплуатирующие систему. Должны иметь возможность зарегистрироваться и авторизоваться в системе для сохранения данных. Основной функционал должен быть доступен и без регистрации. Таким образом, неавторизованный пользователь имеет возможность только просмотра и обновления базы статей, загруженной на устройство.

Читатель – роль пользователя с минимальным набором прав и доступом только к базовым функциям приложения – просмотру базы статей, загруженной на устройство, её обновлению, а также возможности создания списка «Избранных» статей. Не имеет прав на добавление/редактирование статей или взаимодействие с другими пользователями.

Редактор – роль, поддерживающая, помимо функций читателя, дополнительные возможности – добавление и редактирование статей в глобальной базе данных при наличии доступа в Интернет.

Модератор – роль, предоставляющая административные функции по управлению статьями и пользователями.

1.3.1 Функциональные требования

В результате анализа выявлены группы пользователей и прецеденты взаимодействия с системой, отображенные на диаграмме вариантов использования на рисунке 2.

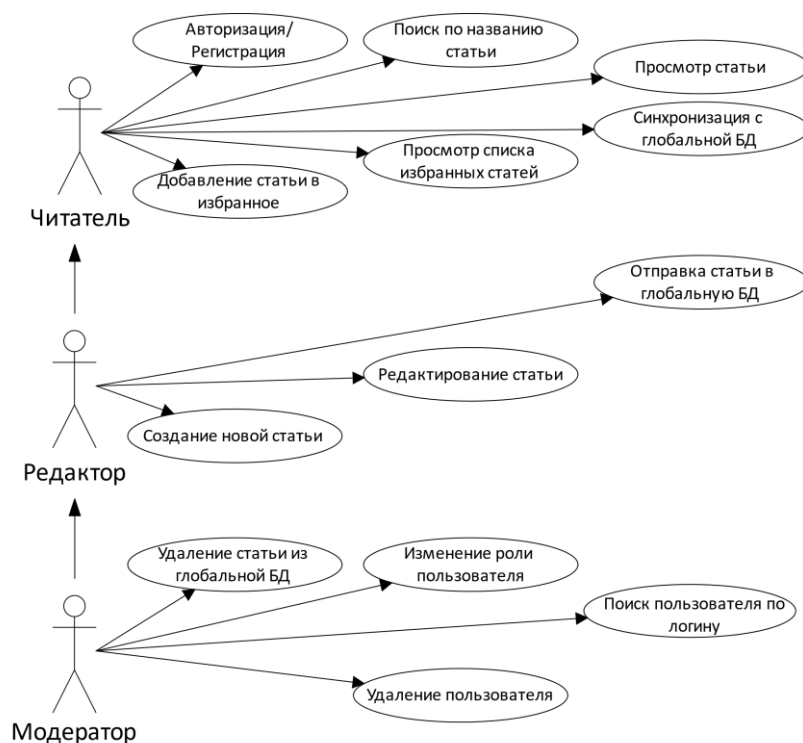


Рисунок 2 - Диаграмма вариантов использования

Ниже приведено подробное описание данных сценариев взаимодействия.

1.3.1.1 Регистрация

Регистрация выполняется на отдельной экранной форме, которая должна содержать поля для ввода логина и пароля и кнопку подтверждения регистрации. Необходимо проверять корректность вводимых пользователем данных. В случае ошибок отображать диалоговое окно с поясняющим текстом.

Регистрация возможно только при наличии устойчивого Интернет-соединения. В случае его отсутствия в момент подтверждения регистрации, отображать окно с сообщением об ошибке.

В случае ошибки сохранения в базе данных по причине конфликта ключей, сообщать пользователю о недоступности указанного логина. Очищать поля ввода на форме и ожидать повторения регистрации с новыми данными.

При успешном создании новой записи пользователя переходить к основной странице приложения.

Переход к форме регистрации осуществляется с основного экрана приложения или с экрана авторизации, если он был открыт ранее.

1.3.1.2 Авторизация

Форма авторизации должна содержать элементы, позволяющие зарегистрированному ранее пользователю войти в систему. Таким образом, необходимы поля для ввода логина и пароля пользователя и кнопка подтверждения авторизации. Также необходимо разместить кнопку перехода к форме регистрации.

После подтверждения пользователем данных для авторизации должна выполняться проверка наличия в базе записи о пользователе с указанным на форме логином. Если такого пользователя нет, отображается окно ошибки.

Иначе, выполняется проверка правильности введенного пароля. Если указанный на форме пароль не совпадает с данными в базе, выводится окно ошибки и очищается поле ввода пароля. Затем система ожидает повторения авторизации с новыми данными.

Если пользователь найден в базе и пароль указан верно, открывается основная страница приложения.

1.3.1.3 Поиск статьи по названию

На основной странице приложения должна быть размещена строка поиска, позволяющая пользователю ввести запрос и выполнить поиск по базе статей.

По запросу пользователя должна осуществляться выборка статей, содержащих в названии строку, указанную в запросе. Результаты должны быть представлены в виде списка, каждый элемент которого содержит название найденной статьи. По нажатию на элемент списка должна открываться страница просмотра выбранной статьи.

Если подходящие под запрос статьи не найдены, должно быть показано уведомление об отсутствии релевантных результатов.

1.3.1.4 Просмотр статьи

Форма просмотра статьи отображает полную информацию о статье и обязательно содержит следующие элементы:

- название статьи;
- текст статьи;
- дата создания;
- логин автора;
- признак наличия статьи в списке «Избранное» для данного пользователя.

Если есть сведения о том, что статья была отредактирована, также присутствуют дата изменения и логин редактора.

Если для статьи найдены блоки дополнительной информации, они также отображаются на форме после основного текста статьи. Если блоков одного типа несколько, отображать их следует списком. Больший приоритет имеют блоки «Важное».

1.3.1.5 Взаимодействие со списком «Избранных» статей

Признак наличия статьи в списке «Избранное» размещается в виде кнопки на странице просмотра статьи и выставляется в зависимости от текущего состояния статьи. По нажатию на кнопку должно осуществляться добавление/удаление статьи из «Избранного».

Кнопка перехода к форме просмотра списка «Избранное» должна быть размещена на основной странице приложения. На форме требуется разместить список, содержащий названия статей, отмеченных пользователем как «Избранное». По нажатию на элемент списка должна открываться страница просмотра выбранной статьи.

Предоставление возможности редактирования списка избранных статей на данной форме не предполагается.

1.3.1.6 Синхронизация с удалённой базой данных

Кнопку, предназначенную для запуска процесса синхронизации, должна быть размещена на основной странице приложения. По нажатию требуется выполнять проверку наличие доступа в Интернет. При его отсутствии должно отображаться окно об ошибке, а процесс синхронизации завершаться на данном этапе.

При наличии доступа в Интернет, должно выводиться окно с предупреждением о том, что синхронизацию следует выполнять только при уверенности в стабильном соединении. Запрос на получение всех статей в глобальной базе следует выполнять только после получения подтверждения от пользователя. По результатам запроса должно осуществляться обновление

статей в локальной базе. По завершении процесса должно быть показано окно с уведомлением об окончании синхронизации.

1.3.1.7 Создание новой статьи

Для пользователей с ролью «Редактор» и «Модератор» на основной странице приложения должна отображаться кнопка перехода к форме создания статьи, по нажатию на которую открывается страница создания/редактирования статьи. На этой странице присутствуют поля для ввода заголовка статьи и основного текста, кнопка сохранения, а также кнопки добавления дополнительных блоков.

По нажатию на кнопку добавления дополнительного блока любого типа открывается окно для ввода текста блока. Нажатие на блок дополнительной информации, ранее размещенный на странице создания/редактирования статьи, приведет к отображению того же окна, но с предзаполнением текстом данного блока. Окно содержит текстовое поле, кнопки «Отменить», «Удалить» и «Сохранить». По нажатию на кнопку «Удалить» происходит удаление блока со страницы. По кнопке «Сохранить» – применение изменений и сокрытие данного окна. По кнопке «Отменить» – сокрытие данного блока без применения изменений, внесенных на окне.

Нажатие на кнопку сохранения статьи приводит к её обновлению в локальной базе данных. Перед сохранением требуется выполнять валидацию – проверку наличия заголовка и названия.

По нажатию на кнопку перехода к предыдущей странице должна выполняться проверка сохранения изменений в локальной базе данных. В случае, если изменения не сохранены, следует отображать окно с предупреждением и кнопками «Сохранить» и «Выйти без сохранения».

1.3.1.8 Редактирование статьи в локальной базе данных

Для пользователей с ролями «Редактор» и «Модератор» требуется отображать на странице просмотра статьи кнопку «Редактировать». По

нажатию открывается страница редактирования статьи. Данная страница по функционалу повторяет страницу создания, описанную в предыдущем пункте, за исключением того, что страница редактирования должна быть предзаполнена данными выбранной статьи.

1.3.1.9 Отправка статьи в глобальную базу данных

Для пользователя с ролью «Редактор» или «Модератор» на странице просмотра статьи должна размещаться кнопка «Обновить в глобальной базе данных» для статей, отредактированных текущим пользователем.

По нажатию должно отображаться окно с предупреждением о необходимости стабильного Интернет-соединения для выполнения данной операции, ожидающее подтверждения от пользователя. Получив подтверждение, следует выполнить запрос обновления статьи в глобальной базе данных с указанием нового текста, заголовка, времени изменения и редактора. По завершению операции пользователь должен увидеть окно подтверждения успешного обновления.

1.3.1.10 Удаление статьи из удаленной базы данных

Для пользователя с ролью «Модератор» на странице просмотра статьи должна отображаться кнопка «Удалить». По нажатию следует выводить окно подтверждения удаления. Получив подтверждение, отправлять запрос удаления статьи из глобальной базы данных. По завершению операции пользователь должен увидеть окно подтверждения успешного выполнения удаления.

1.3.1.11 Изменение прав пользователя

Для пользователя с ролью «Модератор» при активации строки поиска необходимо отображать параметры поиска – «статьи» и «пользователи». При выборе варианта поиска по пользователям использовать строку, указанную в запросе как логин пользователя. Результат поиска должен быть представлен в виде список пользователей, каждый элемент которого содержит логин и роль

пользователя и кнопки для управления ролью. Для каждого пользователя отображаются кнопки выдачи ролей, которыми в данный момент он не обладает. Так, для читателя должны отображаться кнопки выдачи роли редактора и модератора.

По нажатию на кнопку выдачи роли отправлять запрос изменения записи о пользователе в глобальной базе данных с указанной ролью. Получив результат запроса, отображать окно подтверждения выдачи роли или уведомления о возникшей ошибке.

В соответствии с выявленными требованиями и определенными пользовательскими сценариями серверная часть должна обеспечивать возможность выполнения следующих операций:

- создание записи пользователя;
- изменение записи пользователя;
- получение записи пользователя;
- удаление записи пользователя;
- добавление статьи;
- изменение статьи;
- получение списка всех статей;
- удаление статьи;
- добавление статьи в список «Избранное»;
- удаление статьи из списка «Избранное»;
- получение списка «Избранное» для пользователя.

1.3.2 Нефункциональные требования

Система состоит из серверной и клиентской части. Серверная часть представляет собой базу данных и веб-сервер, связанный с ней. Клиентская часть – мобильное приложение для устройств на платформе Android, не ранее версии 10.

Серверная часть должна позволять взаимодействовать с базой пользователей и статей в системе, поддерживая операции хранения, предоставления по запросу, добавления, редактирования и удаления объектов.

Базовый функционал мобильного приложения должен быть доступен без подключения к Интернет. В связи с этим следует использовать локальную базу статей, актуальную на момент выпуска клиента [2]. Реализовать возможность обновления локальной базы статей до состояния глобальной по запросу пользователя при условии наличия устойчивого Интернет-соединения. Взаимодействие с серверной частью осуществлять по запросу к веб-серверу. Также взаимодействия с сервером требуются при регистрации, авторизации, добавлении, удалении и редактировании статьи, а также при изменении прав пользователей.

2 Реализация

2.1 Архитектура системы

2.1.1 Компоненты системы

Архитектуру приложения составляют три основных модуля:

- мобильный клиент;
- удаленная база данных;
- веб-сервер.

Диаграмма разворачивания, отражающая связи между частями системы, показана на рисунке 3.

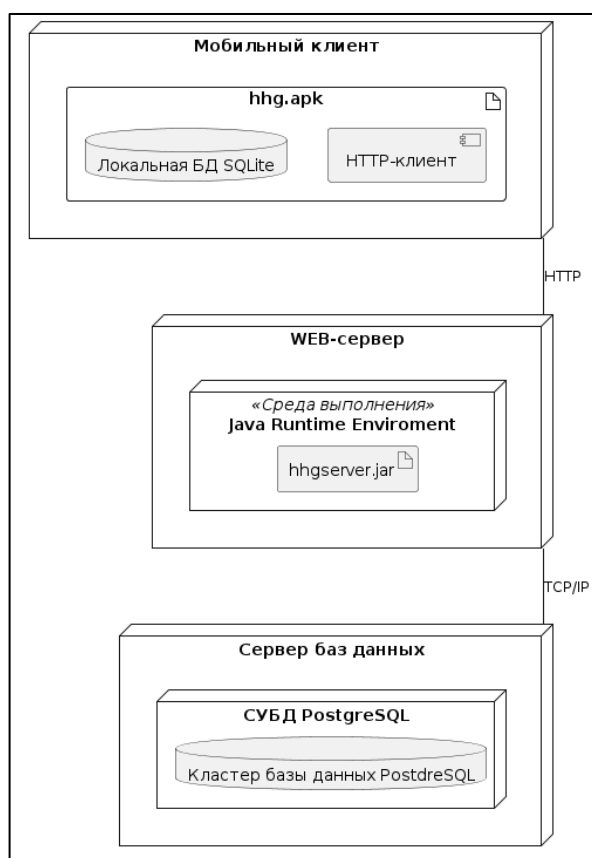


Рисунок 3 - Диаграмма разворачивания

Мобильный клиент — приложение для мобильного устройства, с которым непосредственно взаимодействует пользователь. Клиент реализует графический интерфейс, посредством которого пользователь инициирует

запросы к системе и получает результаты выполнения этих запросов. Мобильный клиент использует локальную базу данных, однако имеет возможность взаимодействия по запросу с глобальной базой данных.

Глобальная база данных предназначена для описания и хранения информации об объектах системы. Для взаимодействия с базой данных используется система управления базами данных, обеспечивающая возможности создания баз данных, поддержания их в актуальном состоянии, выполнения операций над данными.

Связующим звеном между мобильным клиентом и удаленной базой данных выступает веб-сервер. Запросы от клиента поступают на сервер, где выполняется их обработка и взаимодействие с данными в глобальной базе – получение, добавление, модификация или удаление того или иного объекта. В качестве результата обработки сервер возвращает ответ на запрос клиенту, который, в свою очередь, предоставляет сведения о результате операции пользователю посредством графического интерфейса.

2.1.2 Архитектура базы данных

В связи с необходимостью обеспечения автономного режима работы структура локальной базы данных мобильного клиента должна соответствовать глобальной базе данных. Основные сущности: Пользователи и Статьи, включающие в себя особые Блоки.

У каждой статьи есть один автор – Пользователь. В свою очередь Пользователь может быть автором нескольких статей. Таким образом, Пользователь и Статья связаны отношением «один-ко-многим». Для хранения такой связи в реляционной базе данных необходимо в таблице статей выделить поле для ссылки на таблицу пользователей, являющееся внешним ключом.

Статья может содержать один или несколько блоков типа «Важное» или «Требуется уточнения». То есть, Статья и Блок любого типа связаны

отношением «один-ко-многим». Соответственно, в таблицах блоков необходимо добавить дополнительные поля – внешние ключи для ссылки на таблицу статей.

Кроме того, при удалении Статьи должны быть удалены и Блоки, связанные с ней. Для этого необходимо создать соответствующее ограничение на уровне базы данных.

В контексте списка «Избранное» Пользователи и Статьи связаны отношением «многие-ко-многим». Для организации такой связи потребуется служебная таблица, каждая запись которой ссылается на запись в таблице пользователей и на запись в таблице статей.

Таким образом, для хранения данных системы необходимы следующие таблицы:

- user – таблица пользователей, структура которой описана ниже в таблице 1;
- article – таблица статей, описанная в таблице 2;
- clarification_block – таблица блоков «Требуется уточнения», описание структуры приведено в таблице 3;
- important_block – таблица блоков «Важное», описанная в таблице 4;
- favourites – таблица, необходимая для организации списков «Избранных» статей, структура которой приведена в таблице 5.

Таблица 1 – Структура таблицы «user»

Атрибут	Тип данных	Описание
id	serial	Идентификатор записи Первичный ключ
login	character varying(32)	Логин пользователя Уникальное значение
hash	character varying(256)	Хэш пароля
role	character varying(32)	Роль пользователя

Таблица 2 – Структура таблицы «article»

Атрибут	Тип данных	Описание
id	serial	Идентификатор записи Первичный ключ
name	text	Заголовок статьи
text	text	Текст статьи
created_at	timestamp with time zone	Время создания
created_by	integer	Автор Внешний ключ – Ссылка на таблицу пользователей
updated_at	timestamp with time zone	Время редактирования
updated_by	integer	Редактор Внешний ключ – Ссылка на таблицу пользователей
has_important_block	boolean	Признак наличия блока «Важное»
has_clarification_block	boolean	Признак наличия блока «Требуется уточнения»

Таблица 3 – Структура таблицы «important_block»

Атрибут	Тип данных	Описание
id	serial	Идентификатор записи Первичный ключ
article_id	integer	Статья, к которой относится блок Внешний ключ
text	text	Текст блока

Таблица 4 – Структура таблицы «clarification_block»

Атрибут	Тип данных	Описание
id	serial	Идентификатор записи Первичный ключ
article_id	integer	Статья, к которой относится блок Внешний ключ
text	text	Текст блока

Таблица 5 – Структура таблицы «favourite»

Атрибут	Тип данных	Описание
id	serial	Идентификатор записи Первичный ключ
article_id	integer	Ссылка на статью Внешний ключ
user_id	integer	Ссылка на пользователя Внешний ключ

Физическая схема базы данных показана на рисунке 4.

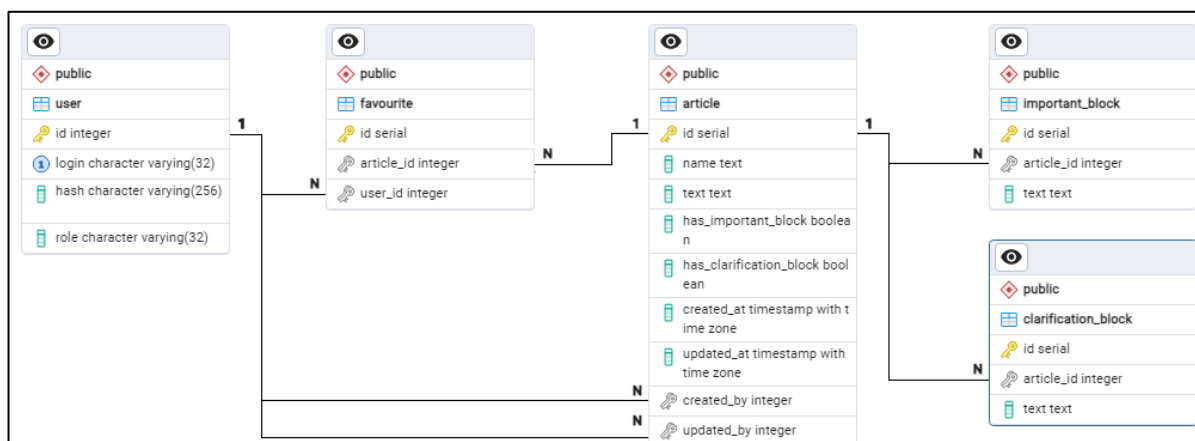


Рисунок 4 – Схема базы данных

2.1.3 Описание REST API

2.1.3.1 Создание профиля пользователя

Запрос: POST {BASE_URL}/api/v1/users

Таблица 6 - Структура тела запроса создания профиля пользователя

№	Атрибут	Тип данных	Обязательность	Описание
1	login	String	✓	Логин пользователя
2	hash	String	✓	Хэш пароля
3	role	String	✓	Роль пользователя

Ответ: Код 201 → Профиль успешно создан

Таблица 7 - Структура тела ответа на запрос создания профиля пользователя

№	Атрибут	Тип данных	Обязательность	Описание
1	USER_INFO	Object	✓	Информация о пользователе
1.1	id	Int	✓	Идентификатор пользователя
1.2	login	String	✓	Логин пользователя
1.3	hash	String	✓	Хэш пароля
1.4	role	String	✓	Роль пользователя

2.1.3.2 Получение профиля пользователя по логину

Запрос: GET {BASE_URL}/api/v1/users/login/{login}

Таблица 8 – Описание атрибута запроса получения профиля пользователя

№	Атрибут	Тип данных	Обязательность	Описание
1	login	String	✓	Логин пользователя

Ответ: Код 200 → Пользователь найден

Таблица 9 - Структура тела ответа на запрос получения профиля пользователя

№	Атрибут	Тип данных	Обязательность	Описание
1	USER_INFO	Object		Информация о пользователе
1.1	id	Int	✓	Идентификатор пользователя
1.2	login	String	✓	Логин пользователя
1.3	hash	String	✓	Хэш пароля
1.4	role	String	✓	Роль пользователя

2.1.3.3 Изменение профиля пользователя

Запрос: PUT {BASE_URL}/api/v1/users

Таблица 10 – Структура тела запроса изменения профиля пользователя

№	Атрибут	Тип данных	Обязательность	Описание
1	USER_INFO	Object	✓	Информация о пользователе
1.1	id	Int	✓	Идентификатор пользователя
1.2	login	String	✓	Логин пользователя
1.3	hash	String	✓	Хэш пароля
1.4	role	String	✓	Роль пользователя

Ответ: Код 200 → Информация о пользователе изменена

Таблица 11 – Структура тела ответа на запрос изменения профиля пользователя

№	Атрибут	Тип данных	Обязательность	Описание
1	USER_INFO	Object	✓	Информация о пользователе
1.1	id	Int	✓	Идентификатор пользователя
1.2	login	String	✓	Логин пользователя
1.3	hash	String	✓	Хэш пароля
1.4	role	String	✓	Роль пользователя

2.1.3.4 Удаление профиля пользователя по идентификатору

Запрос: DELETE {BASE_URL}/api/v1/users/{id}

Таблица 12 – Описание атрибута запроса удаления профиля пользователя

№	Атрибут	Тип данных	Обязательность	Описание
1	id	Int	✓	Идентификатор профиля пользователя

Ответ: Код 200 → Пользователь удален

2.1.3.5 Получение списка статей

Запрос: GET {BASE_URL}/api/v1/articles

Таблица 13 – Структура тела ответа на запрос получения списка статей

№	Атрибут	Тип данных	Обязательность	Описание
1	ARTICLE_INFO	Object		Информация о статье
1.1	id	Int	✓	Идентификатор записи
1.2	title	String	✓	Название статьи
1.3	text	String	✓	Текст статьи
1.4	createdAt	String	✓	Время создания
1.5	author	Object	✓	Сведения об авторе
1.5.1	id	Int	✓	Идентификатор пользователя
1.5.2	login	String	✓	Логин пользователя
1.5.3	hash	String	✓	Хэш пароля
1.5.4	role	String	✓	Роль пользователя
1.6	updatedAt	String	✓	Время обновления
1.7	editor		✓	Сведения о редакторе
1.7.1	id	Int	✓	Идентификатор пользователя
1.7.2	login	String	✓	Логин пользователя
1.7.3	hash	String	✓	Хэш пароля
1.7.4	role	String	✓	Роль пользователя
1.8	hasImportantBlock	Boolean	✓	Признак наличия блока «Важное»
1.9	hasClarificationBlock	Boolean	✓	Признак наличия блока «Требуется уточнение»
1.10	importantBlocks	Object		Блоки «Важное»
1.10.1	id	Int	✓	Идентификатор записи
1.10.2	text	String	✓	Текст

№	Атрибут	Тип данных	Обязательность	Описание
1.10.3	articleId	Int	✓	Ссылка на статью
1.11	clarificationBlocks	Object		Блоки «Требуется уточнение»
1.11.1	id	Int	✓	Идентификатор записи
1.11.2	text	String	✓	Текст
1.11.3	article_id	Int	✓	Ссылка на статью

2.1.3.6 Удаление статьи по идентификатору

Запрос: DELETE {BASE_URL}/api/v1/articles/{id}

Таблица 14 – Описание атрибута запроса удаления профиля пользователя

№	Атрибут	Тип данных	Обязательность	Описание
1	id	Int	✓	Идентификатор статьи

Ответ: Код 200 → Статья удалена

2.1.3.7 Изменение статьи

Запрос: PUT {BASE_URL}/api/v1/articles

Таблица 15 – Структура тела запроса изменения статьи

№	Атрибут	Тип данных	Обязательность	Описание
1	ARTICLE_INFO	Object		Информация о статье
1.1	id	Int	✓	Идентификатор записи
1.2	title	String	✓	Название статьи
1.3	text	String	✓	Текст статьи
1.4	createdAt	String	✓	Время создания
1.5	author	Object	✓	Сведения об авторе
1.5.1	id	Int	✓	Идентификатор пользователя
1.5.2	login	String	✓	Логин пользователя
1.5.3	hash	String	✓	Хэш пароля
1.5.4	role	String	✓	Роль пользователя
1.6	updatedAt	String	✓	Время обновления
1.7	editor	Object	✓	Сведения о редакторе
1.7.1	id	Int	✓	Идентификатор пользователя
1.7.2	login	String	✓	Логин пользователя
1.7.3	hash	String	✓	Хэш пароля
1.7.4	role	String	✓	Роль пользователя
1.8	hasImportantBlock	Boolean	✓	Признак наличия блока «Важное»
1.9	hasClarificationBlock	Boolean	✓	Признак наличия

№	Атрибут	Тип данных	Обязательность	Описание
				блока «Требуется уточнение»
1.10	importantBlocks	Object		Блоки «Важное»
1.10.1	id	Int	✓	Идентификатор записи
1.10.2	text	String	✓	Текст
1.10.3	articleId	Int	✓	Ссылка на статью
1.11	clarificationBlocks	Object		Блоки «Требуется уточнение»
1.11.1	id	Int	✓	Идентификатор записи
1.11.2	text	String	✓	Текст
1.11.3	article_id	Int	✓	Ссылка на статью

Ответ: Код 200 → Статья отредактирована

Таблица 16 – Структура тела ответа на запрос обновления статьи

№	Атрибут	Тип данных	Обязательность	Описание
1	ARTICLE_INFO	Object		Информация о статье
1.1	id	Int	✓	Идентификатор записи
1.2	title	String	✓	Название статьи
1.3	text	String	✓	Текст статьи
1.4	createdAt	String	✓	Время создания
1.5	author	Object	✓	Сведения об авторе
1.5.1	id	Int	✓	Идентификатор пользователя
1.5.2	login	String	✓	Логин пользователя
1.5.3	hash	String	✓	Хэш пароля
1.5.4	role	String	✓	Роль пользователя
1.6	updatedAt	String	✓	Время обновления
1.7	editor		✓	Сведения о редакторе
1.7.1	id	Int	✓	Идентификатор пользователя
1.7.2	login	String	✓	Логин пользователя
1.7.3	hash	String	✓	Хэш пароля
1.7.4	role	String	✓	Роль пользователя
1.8	hasImportantBlock	Boolean	✓	Признак наличия блока «Важное»
1.9	hasClarificationBlock	Boolean	✓	Признак наличия блока «Требуется уточнение»
1.10	importantBlocks	Object		Блоки «Важное»
1.10.1	id	Int	✓	Идентификатор записи
1.10.2	text	String	✓	Текст
1.10.3	articleId	Int	✓	Ссылка на статью
1.11	clarificationBlocks	Object		Блоки «Требуется уточнение»

№	Атрибут	Тип данных	Обязательность	Описание
1.11.1	id	Int	✓	Идентификатор записи
1.11.2	text	String	✓	Текст
1.11.3	article_id	Int	✓	Ссылка на статью

2.1.3.8 Добавление статьи

Запрос: POST {BASE_URL}/api/v1/articles

Таблица 17 – Структура тела запроса добавления статьи

№	Атрибут	Тип данных	Обязательность	Описание
1	ARTICLE_INFO	Object		Информация о статье
1.1	id	Int	✓	Идентификатор записи
1.2	title	String	✓	Название статьи
1.3	text	String	✓	Текст статьи
1.4	createdAt	String	✓	Время создания
1.5	author	Object	✓	Сведения об авторе
1.5.1	id	Int	✓	Идентификатор пользователя
1.5.2	login	String	✓	Логин пользователя
1.5.3	hash	String	✓	Хэш пароля
1.5.4	role	String	✓	Роль пользователя
1.6	updatedAt	String	✓	Время обновления
1.7	editor		✓	Сведения о редакторе
1.7.1	id	Int	✓	Идентификатор пользователя
1.7.2	login	String	✓	Логин пользователя
1.7.3	hash	String	✓	Хэш пароля
1.7.4	role	String	✓	Роль пользователя
1.8	hasImportantBlock	Boolean	✓	Признак наличия блока «Важное»
1.9	hasClarificationBlock	Boolean	✓	Признак наличия блока «Требуется уточнение»
1.10	importantBlocks	Object		Блоки «Важное»
1.10.1	id	Int	✓	Идентификатор записи
1.10.2	text	String	✓	Текст
1.10.3	articleId	Int	✓	Ссылка на статью
1.11	clarificationBlocks	Object		Блоки «Требуется уточнение»
1.11.1	id	Int	✓	Идентификатор записи
1.11.2	text	String	✓	Текст
1.11.3	article_id	Int	✓	Ссылка на статью

Ответ: Код 200 → Статья отредактирована

Таблица 18 – Структура тела ответа на запрос обновления статьи

№	Атрибут	Тип данных	Обязательность	Описание
1	ARTICLE_INFO	Object		Информация о статье
1.1	id	Int	✓	Идентификатор записи
1.2	title	String	✓	Название статьи
1.3	text	String	✓	Текст статьи
1.4	createdAt	String	✓	Время создания
1.5	author	Object	✓	Сведения об авторе
1.5.1	id	Int	✓	Идентификатор пользователя
1.5.2	login	String	✓	Логин пользователя
1.5.3	hash	String	✓	Хэш пароля
1.5.4	role	String	✓	Роль пользователя
1.6	updatedAt	String	✓	Время обновления
1.7	editor		✓	Сведения о редакторе
1.7.1	id	Int	✓	Идентификатор пользователя
1.7.2	login	String	✓	Логин пользователя
1.7.3	hash	String	✓	Хэш пароля
1.7.4	role	String	✓	Роль пользователя
1.8	hasImportantBlock	Boolean	✓	Признак наличия блока «Важное»
1.9	hasClarificationBlock	Boolean	✓	Признак наличия блока «Требуется уточнение»
1.10	importantBlocks	Object		Блоки «Важное»
1.10.1	id	Int	✓	Идентификатор записи
1.10.2	text	String	✓	Текст
1.10.3	articleId	Int	✓	Ссылка на статью
1.11	clarificationBlocks	Object		Блоки «Требуется уточнение»
1.11.1	id	Int	✓	Идентификатор записи
1.11.2	text	String	✓	Текст
1.11.3	article_id	Int	✓	Ссылка на статью

2.1.3.9 Добавление блока «Важное»

Запрос: POST {BASE_URL}/api/v1/important

Таблица 19 – Структура тела запроса добавления блока «Важное»

№	Атрибут	Тип данных	Обязательность	Описание
1	text	String	✓	Текст
2	articleId	Int	✓	Ссылка на статью

Ответ: Код 201 → Блок «Важное» создан

Таблица 20 – Структура тела ответа на запрос добавления блока «Важное»

№	Атрибут	Тип данных	Обязательность	Описание
1	id	Int	✓	Идентификатор записи
2	text	String	✓	Текст
3	articleId	Int	✓	Ссылка на статью

2.1.3.10 Изменение блока «Важное»

Запрос: PUT {BASE_URL}/api/v1/important

Таблица 21 – Структура тела запроса изменения блока «Важное»

№	Атрибут	Тип данных	Обязательность	Описание
1	id	Int	✓	Идентификатор записи
2	text	String	✓	Текст
3	articleId	Int	✓	Ссылка на статью

Ответ: Код 200 → Блок «Важное» изменен

Таблица 22 – Структура тела ответа на запрос изменения блока «Важное»

№	Атрибут	Тип данных	Обязательность	Описание
1	id	Int	✓	Идентификатор записи
2	text	String	✓	Текст
3	articleId	Int	✓	Ссылка на статью

2.1.3.11 Удаление блока «Важное»

Запрос: DELETE {BASE_URL}/api/v1/important/{id}

Таблица 23 – Описание атрибута запроса удаления блока «Важное»

№	Атрибут	Тип данных	Обязательность	Описание
1	id	Int	✓	Идентификатор записи

Ответ: Код 200 → Запись удалена

2.1.3.12 Добавление блока «Требуется уточнение»

Запрос: POST {BASE_URL}/api/v1/clarification

Таблица 24 – Структура тела запроса добавления блока «Требуется уточнение»

№	Атрибут	Тип данных	Обязательность	Описание
1	text	String	✓	Текст
2	articleId	Int	✓	Ссылка на статью

Ответ: Код 201 → Блок «Требуется уточнение» создан

Таблица 25 – Структура тела ответа на запрос добавления блока «Требуется уточнение»

№	Атрибут	Тип данных	Обязательность	Описание
1	id	Int	✓	Идентификатор записи
2	text	String	✓	Текст
3	articleId	Int	✓	Ссылка на статью

2.1.3.13 Изменение блока «Требуется уточнение»

Запрос: PUT {BASE_URL}/api/v1/clarification

Таблица 26 – Структура тела запроса изменения блока «Важное»

№	Атрибут	Тип данных	Обязательность	Описание
1	id	Int	✓	Идентификатор записи
2	text	String	✓	Текст
3	articleId	Int	✓	Ссылка на статью

Ответ: Код 200 → Блок «Требуется уточнение» изменен

Таблица 27 – Структура тела ответа на запрос изменения блока «Требуется уточнение»

№	Атрибут	Тип данных	Обязательность	Описание
1	id	Int	✓	Идентификатор записи
2	text	String	✓	Текст
3	articleId	Int	✓	Ссылка на статью

2.1.3.14 Удаление блока «Требуется уточнение»

Запрос: DELETE {BASE_URL}/api/v1/clarification/{id}

Таблица 28 – Описание атрибута запроса удаления блока «Требуется уточнение»

№	Атрибут	Тип данных	Обязательность	Описание
1	id	Int	✓	Идентификатор записи

2.2 Выбор инструментов разработки

2.2.1 Выбор СУБД

Одним из наиболее важных этапов разработки информационной системы является выбор системы управления базами данных. Наибольшей популярностью пользуются СУБД, основанные на реляционной модели – способе представления данных в виде таблиц. К таким СУБД относятся, например, Oracle Database, PostgreSQL и SQLite [3].

Все указанные решения зарекомендовали себя как достаточно надежные, производительные и мощные инструменты по управлению данными. Наличие исчерпывающей документации делает данные продукты удобными в использовании и сопровождении.

Среди особых достоинств объектно-реляционной СУБД Oracle Database выделяется возможность взаимодействия с облачными сервисами и встроенная поддержка языка программирования JavaScript. Однако необходимо указать и недостатки, среди которых выявляется требовательность системы к аппаратным ресурсам и высокая стоимость программного продукта.

Объектно-реляционная СУБД PostgreSQL, в отличие от Oracle Database, распространяется как свободное программное обеспечение и лишена недостатка, связанного с высокой стоимостью. Другое достоинство – наличие поддержки широкого списка типов данных, который также можно дополнять собственными типами, высокую надежность, а также наличие графического инструмента администрирования pgAdmin, поставляемого в базовом комплекте устанавливаемых компонентов [3].

СУБД SQLite является менее мощной по сравнению с аналогами и используется главным образом в мобильных устройствах, веб-сайтах с низкой и средней нагрузкой, приложениях, запускаемых локально [4]. Главное преимущество данной СУБД – встраиваемость. Поскольку SQLite является не полноценным клиент-серверным приложением, а библиотекой, подключаемой к программе, процесс интеграции базы данных с другими компонентами системы, расположенными на том же устройстве, что и БД, значительно упрощается.

Таким образом, наиболее подходящей СУБД для мобильного клиента является легковесная, но при этом обеспечивающая полную поддержку языка SQL, СУБД SQLite. В качестве удалённой СУБД выбрана PostgreSQL, предоставляющая все необходимые функции взаимодействия с данными и обладающая наиболее удобным инструментом администрирования с графическим интерфейсом.

2.2.2 Выбор инструментов разработки серверной части

Одним из наиболее популярных фреймворков для создания веб-сервисов является Spring Boot – универсальный инструмент с открытым исходным кодом для Java-платформы. Spring Boot является расширением фреймворка Spring, разработанным специально для создания веб-приложений и поддерживающим ряд встроенных инструментов, которые упрощают написание кода именно под эти задачи. Spring Framework включает в себя механизмы внедрения зависимостей и инверсии управления, доступа к базе данных, возможность разработки в парадигме аспектно-ориентированного программирования, поддержку веб-инфраструктуры MVC и многие другие функциональности, ускоряющие и упрощающие процесс создания серверных приложений [5].

Использование данного фреймворка для создания веб-сервера предполагает использование одного из двух языков программирования – Java или Kotlin.

Java является строго типизированным кроссплатформенным объектно-ориентированным языком программирования общего назначения, нашедшем применение в разработке практически любого вида программного обеспечения. Это быстрый, безопасный и надежный язык программирования, собравший за долгие годы своего существования огромное сообщество разработчиков.

Kotlin – более молодой язык программирования, разработанный в качестве новой версии Java, лишенной её недостатков. Kotlin запускается на виртуальной машине Java и поддерживает полную совместимость с Java. Соответственно, применять Kotlin можно в тех же областях, что и Java – бэкенд, веб-приложения, клиент-серверные системы и множество других задач [6]. Однако наибольшее распространение Kotlin получил в мобильной разработке как официальный язык программирования Google отчасти из-за ликвидации недостатков Java, отчасти из-за менее консервативного сообщества мобильных разработчиков.

Сообщество разработчиков, использующих Spring Boot, на данный момент отдает предпочтение Java. Хотя компания JetBrains, разработавшая Kotlin, активно содействует распространению языка именно в контексте серверной разработки, основную нишу так или иначе занимает Java.

Таким образом, для разработки серверной части системы выбран язык программирования Java, поддерживающий взаимодействие со Spring Boot и библиотекой Hibernate. Также используется библиотека Lombok – содержащая ряд аннотаций, разработанных для исключения повторяющегося или шаблонного кода, который содержат большинство Java-объектов –

конструкторов с различным набором параметров, геттеров и сеттеров, методов сравнения и приведения.

Среди интегрированных сред разработки, позволяющих работать со Spring Framework, выделяют IntelliJ IDEA и Eclipse, а также редактор кода VSCode. Наиболее современным и удобным решением среди указанных является IntelliJ IDEA, включающая Spring Framework в основной состав компонентов Ultimate версии, бесплатный доступ к которой возможно получить на период 30 дней.

2.2.3 Выбор инструментов разработки клиентской части

Разработка приложений для устройств на платформе Android традиционно ведётся на языках программирования Java и Kotlin. Каждый из этих языков программирования имеет свои преимущества и недостатки.

Разработка Android-приложений на Java дает и разработчикам, и пользователям широкий перечень преимуществ. Java сообщество – одно из наиболее влиятельных во всем мире, а сам язык программирования универсальный, простой в изучении, кроссплатформенный, предоставляющий гарантии безопасности и поддерживающий обратную совместимость версий. Трактовки конструкций Java преимущественно простые, имеют однозначное прочтение. Однако при написании приложений на Java код получается многословным и длинным, что нравится не всем.

Android-разработка на Kotlin активно развивается благодаря поддержке рабочих отношений с облачной платформой Gradle, Spring, Google, следовательно, и наличии средств для взаимодействия с указанными продуктами. К основным преимуществам Kotlin можно отнести совместимость со многими фреймворками и библиотеками Java, высокий уровень безопасности, повышенную производительность, интуитивный и последовательный синтаксис, а также поддержку от IDE.

Однако нужно помнить, что Kotlin является надстройкой над Java, соответственно, требует больших затрат ресурсов при сборке и компиляции программ.

Таким образом, для разработки клиентской части системы выбран язык программирования Kotlin как наиболее современный и простой в использовании.

В качестве среды разработки выбрана Android Studio, обеспечивающая максимальную поддержку средств языка Kotlin и предоставляющая все инструменты, необходимые при разработке Android-приложений [7].

2.3 Разработка серверной части

Архитектура серверной части включает в себя классы нескольких типов: контроллеры, сервисы, репозитории, представления и сущности.

Сущности представляют собой структуры базы данных, устанавливают связи и между собой и задают ограничения. Репозитории осуществляют операции взаимодействия с СУБД – чтение данных по различным условиям, запись, изменение и удаление записей в базе данных. Контроллеры отвечают за выполнение REST API запросов. Сервисы связывают репозитории и контроллеры, выполняя обработку данных, поступивших из REST-запроса, для записи в базу данных через репозиторий и подготавливая данные, полученные из базы данных от репозитория, для передачи их в качестве REST-ответа. Представления позволяют описать сущности системы, в формате, подходящем для использования за пределами системы.

Принцип взаимодействия описанных компонентов во время обработки запроса в общем виде продемонстрирован на диаграмме последовательности на рисунке 5.

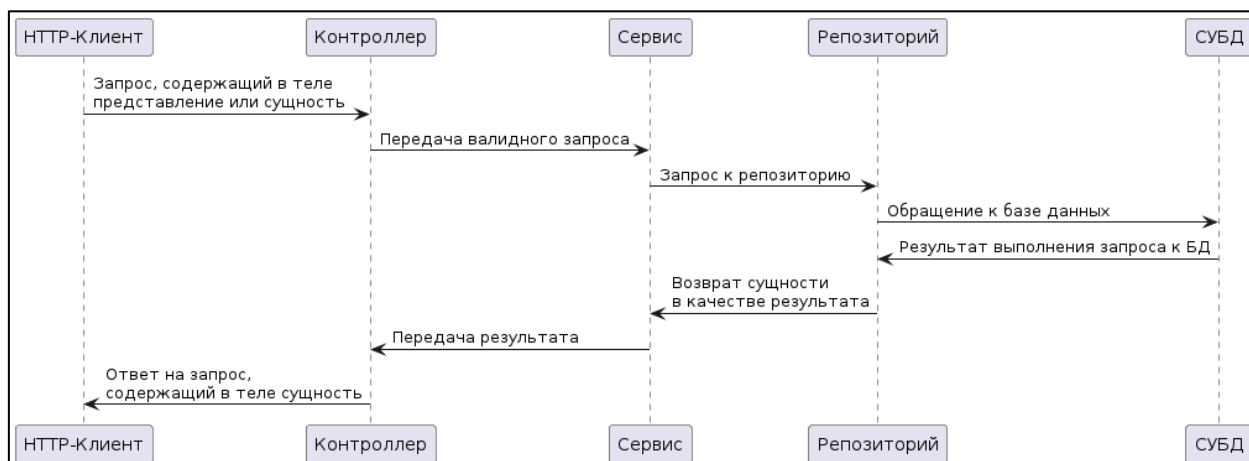


Рисунок 5 - Диаграмма последовательности обработки запроса сервером

Таким образом для реализации спроектированного REST-сервиса для каждой сущности были созданы следующие программные модули: сущность, представление, репозиторий, сервис и контроллер. Диаграмма классов, отображающая разработанные модули сервиса и их взаимосвязи, показана на рисунке 6.

2.4 Разработка мобильного клиента

В результате постановки задачи были определены основные экранные формы и обозначены условия переходов между ними. На основе этих данных построена карта экранов, показанная на рисунке 7.



Рисунок 7 - Карта экранов

Реализация каждого из экранов требует файл вёрстки, описывающий визуальное представление располагаемых на экране элементов, и файл активности, задающий поведение системы при взаимодействии пользователя с приложением. Некоторые компоненты требуют дополнительного описания. Например, для размещения на форме компонента *RecyclerView*, используемого для отображения динамических списков, требуется также создать файл верстки элемента списка и класс-адаптер, связывающий визуальное представление с цифровыми данными. Для реализации обработчиков обращений пользователя к элементу такого списка требуется интерфейс, объявляющий методы использования частей элемента и позволяющий определять их реализацию в рамках требуемой активности.

Диаграмма классов мобильного приложения показана на рисунке 8.

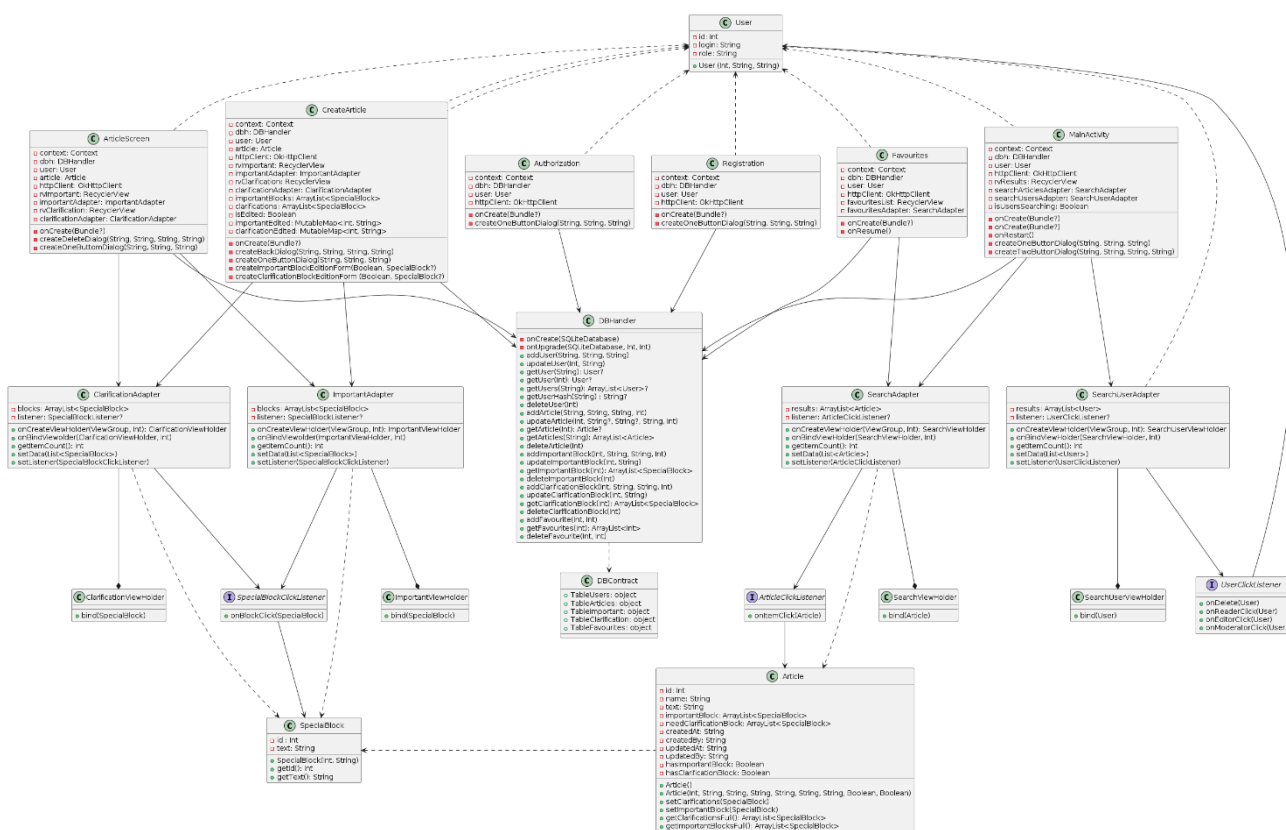


Рисунок 8 – Диаграмма классов мобильного приложения

Основной экран приложения, показанный на рисунке 9 содержит кнопки для перехода к регистрации, обновления локальной БД, просмотра списка «Избранное» и добавления статьи, а также поле для выполнения поиска по справочнику статей или по пользователям.

Набор отображаемых элементов меняется в зависимости от роли пользователя. Так, неавторизованные пользователи не имеют возможности

добавить статью, а выполнять поиск по пользователям могут только модераторы. Кнопка «Зарегистрироваться» отображается только для неавторизованных пользователей. Для остальных она заменяется на кнопку «Назад», позволяющую сменить учетную запись.



Рисунок 9 - Основной экран приложения

Формы авторизации и регистрации, показанные на рисунках 10 и 11 содержат похожие элементы – поля ввода логина и пароля, а также кнопки подтверждения операции и перехода между данными двумя процессами. Также предусмотрена обработка ошибок со стороны пользователя или системы – отображение диалогового окна с сообщением или уведомлением. Данная форма отображена на рисунке 12 и используется в рамках всего приложения в качестве стандартного решения подобных ситуаций.

Успешная регистрация/авторизация завершается переходом к основной странице и отображением на ней элементов, соответствующих роли.

Рисунок 10 - Экран авторизации

Рисунок 11 – Экран регистрации

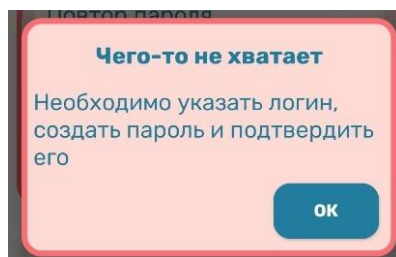


Рисунок 12 - Универсальное диалоговое окно

Выполнение поиска по статьям осуществляется с помощью поискового поля на основном экране. Если по запросу не было найдено ни одной статьи, отображается соответствующее сообщение. Иначе, если результаты есть, они представляются в виде списка, каждый элемент которого содержит название статьи. Состояние основного экрана при успешном выполнении поиска показано на рисунке 13.

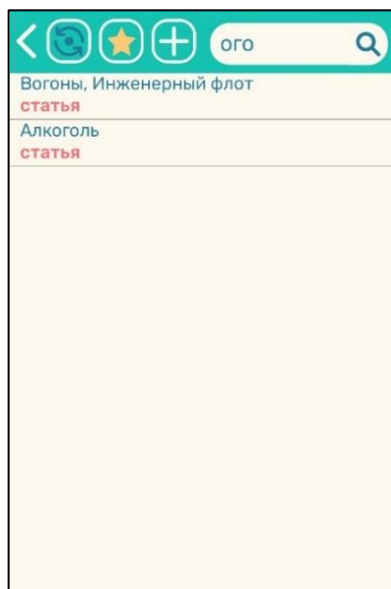


Рисунок 13 - Результат выполнения поиска по статьям

Нажатие на статью в списке результатов приводит к переходу на экран просмотра статьи, показанный на рисунке 14.

Основное пространство экрана занимают заголовок и текст статьи. Особым образом выделяются блоки «Важное» и «Требует уточнения». Также

в справочном формате отображаются логины создателя и последнего редактора статьи, а также время создания и изменения.



Рисунок 14 - Экран просмотра статьи

Кнопка «Добавить в Избранное» / «Удалить из Избранного» доступна всем авторизованным пользователям и предназначена для управления статьями в списке избранного.

Кнопка «Удалить» отображается только для модератора и позволяет удалить статью из глобальной базы данных при наличии устойчивого Интернет-соединения. Кнопка «Изменить» доступна редактору и модератору. По нажатию осуществляется переход к экрану редактирования.

Экран редактирования содержит те же функциональные элементы, что и экран создания статьи, показанный на рисунке 15. Отличие заключается в заголовке и выполнении предзаполнения данными редактируемой статьи текстовых областей «Заголовок» и «Текст статьи». На данной странице реализован функционал по заполнению статьи данными. При сохранении

предусмотрена дополнительная валидация на стороне клиента, не позволяющая добавлять статьи с некорректным содержанием.



Рисунок 15 - Экран создания/редактирования статьи

Кнопки добавления блоков «Важное» и «Требуется уточнения» приводят к вызову модального окна, показанного на рисунке 16. К аналогичному результату приводит нажатие на ранее добавленный особый блок. Данное окно позволяет ввести/изменить текст блока и подтвердить его размещение с помощью кнопки «Сохранить». Также на модальном окне располагаются кнопки «Отмена» – выход из процесса работы с особым блоком – и «Удалить» - удаление особого блока, созданного ранее.

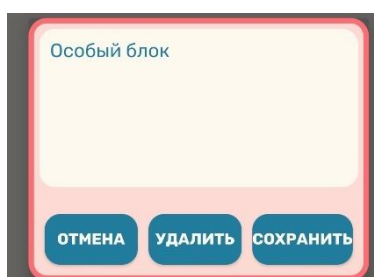


Рисунок 16 - Модальное окно добавления текста блока "Важное"/"Требуется уточнения"

Кнопка «Сохранить статью» на основном экране выполняет добавление новой статьи или изменение существующей в локальной базе данных. Затем осуществляется переход к основному экрану.

На рисунке 17 показано состояние основного экрана после выполнения поиска по пользователям. На основной части экрана располагается список пользователей с указанием логина и роли, а также кнопки для удаления или изменения роли пользователя. Нажатие на каждую из кнопок смены роли приводит к выполнению запроса на изменение профиля пользователя, после выполнения которого на экран выводится уведомление об успешном выполнении операции, показанное на рисунке 18.

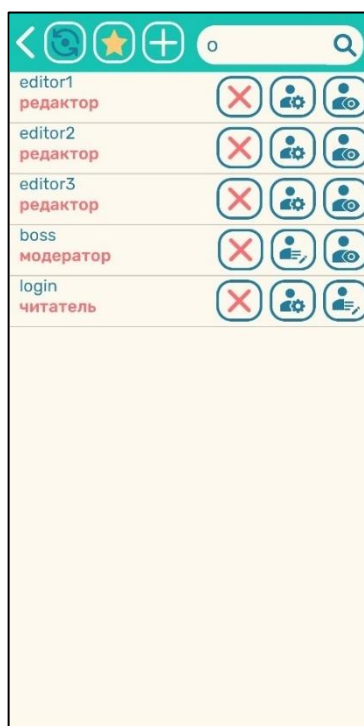


Рисунок 17 - Результат выполнения поиска по пользователям

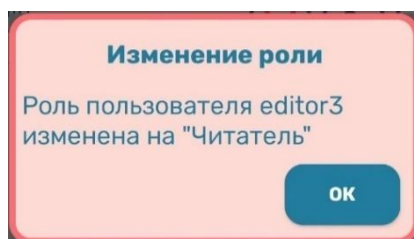


Рисунок 18 – Уведомление о смене роли пользователя

На рисунке 19 продемонстрирована последняя страница приложения – список «Избранное». Данный экран содержит список статей, которые были отмечены пользователем как важные. Нажатие на элемент в списке вызывает переход к странице просмотра выбранной статьи.

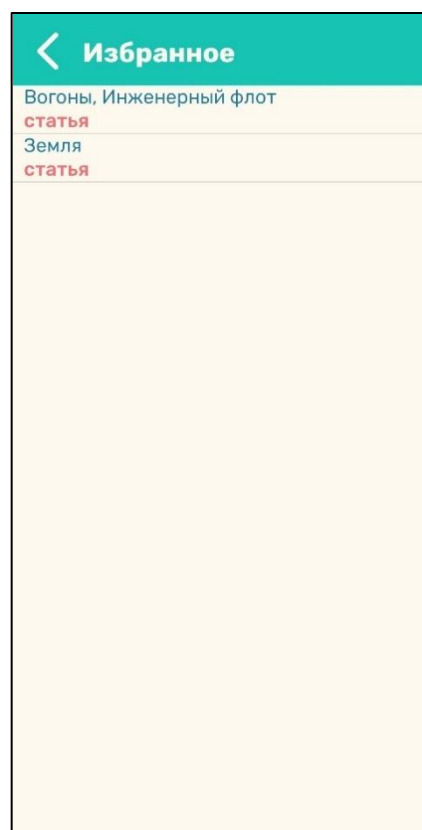


Рисунок 19 - Экран "Избранное"

3 Методы испытаний

3.1 Тестирование REST API

Проверка работоспособности серверной части системы осуществлялось с помощью инструмента для тестирования API Insomnia REST Client. Данный программный продукт позволяет выполнять REST-запросы с необходимыми параметрами и просматривать полученные с сервера ответы.

Результаты проверки работоспособности всех методов API представлены ниже на рисунках 20 – 33.

GET localhost:8081/api/v1/articles Send

Body Auth Query Headers 1 Docs

Enter a URL and send to get a response

200 OK 198 ms 26.3 KB 8 Hours Ago

Preview Headers 3 Cookies Timeline

```
1 [
2   {
25     {
26       "id": 7,
27       "title": "Земля",
28       "text": "Безвредна",
29       "createdAt": "2024-05-13 15:00:00+03",
30       "author": {
31         "id": 2,
32         "login": "administrator",
33         "hash":
"$2a$12$JnB5F57l/VBMXq9gAUQaQ.YDtEeLN.V8jmwI58Sj0xNySF2vndp2G",
34         "role": "moderator"
35       },
36       "updatedAt": "2024-05-13 15:00:00+03",
37       "editor": {
38         "id": 2,
39         "login": "administrator",
40         "hash":
"$2a$12$JnB5F57l/VBMXq9gAUQaQ.YDtEeLN.V8jmwI58Sj0xNySF2vndp2G",
41         "role": "moderator"
42       },
43       "hasImportantBlock": false,
44       "hasClarificationBlock": false,
45       "importantBlocks": [],
46       "clarificationBlocks": [
47         {
48           "id": 53,
49           "text": "В основном безвредна",
50           "articleId": 7
51         }
52       ]
53     }
54   ]
55 }
```

Рисунок 20 – Ответ на запрос получения всех статей

POST localhost:8081/api/v1/articles Send

JSON Auth Query Headers 2 Docs

```
1 {
2   "title": "Земля",
3   "text": "Безвредна",
4   "createdAt": "2024-05-13 15:00:00+03",
5   "createdBy": 2,
6   "updatedAt": "2024-05-13 15:00:00+03",
7   "updatedBy": 2,
8   "hasImportantBlock": false,
9   "hasClarificationBlock": false
10 }
```

Beautify JSON

200 OK 12.9 ms 501 B 1 Day Ago

Preview Headers 3 Cookies Timeline

```
1 {
2   "id": 404,
3   "title": "Земля",
4   "text": "Безвредна",
5   "createdAt": "2024-05-13 15:00:00+03",
6   "author": {
7     "id": 2,
8     "login": "administrator",
9     "hash":
10    "$2a$12$JnB5F57l/VBMXq9gAUQaQ.YDtEeLN.V8jmwI58Sj0xNySF2vndp2G",
11    "role": "moderator"
12  },
13  "updatedAt": "2024-05-13 15:00:00+03",
14  "editor": {
15    "id": 2,
16    "login": "administrator",
17    "hash":
18    "$2a$12$JnB5F57l/VBMXq9gAUQaQ.YDtEeLN.V8jmwI58Sj0xNySF2vndp2G",
19    "role": "moderator"
20  },
21  "hasImportantBlock": false,
22  "hasClarificationBlock": false,
23  "importantBlocks": null,
24  "clarificationBlocks": null
25 }
```

Рисунок 21 – Результат выполнения метода добавления статьи

DELETE localhost:8081/api/v1/articles/411 Send

Body Auth Query Headers 1 Docs

200 OK 17.7 ms 4 B 1 Day Ago

Preview Headers 3 Cookies Timeline

```
1 "OK"
```

Рисунок 22 – Результат удаления статьи

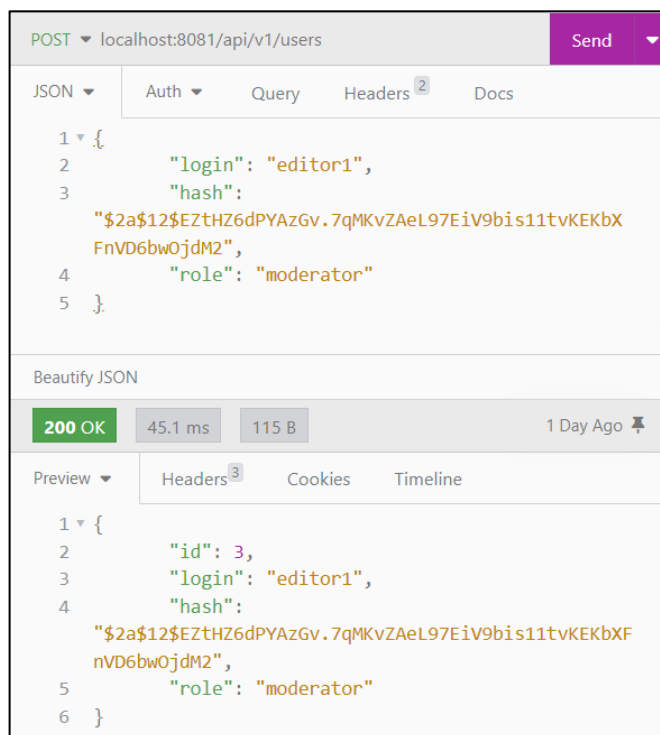


Рисунок 23 – Результат выполнения запроса добавления пользователя

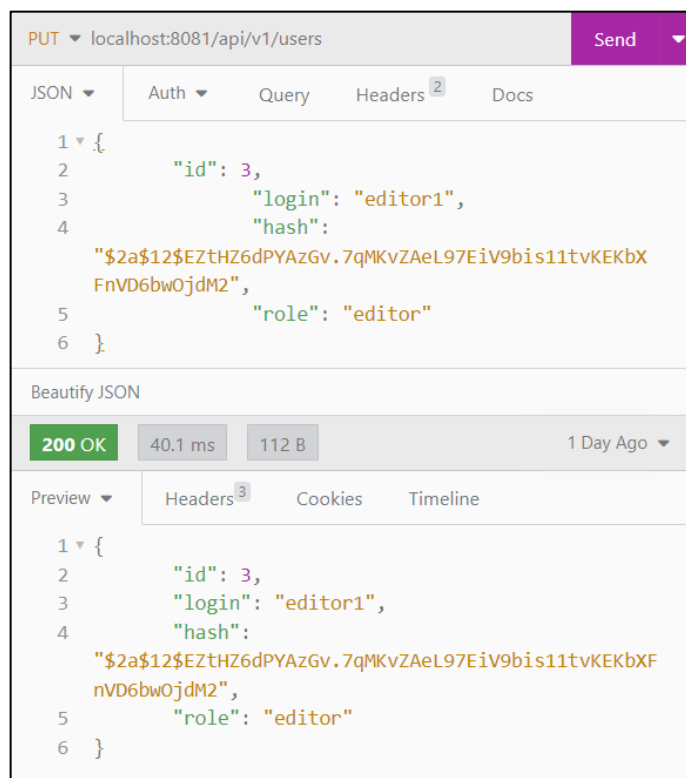


Рисунок 24 – Результат выполнения запроса изменения профиля пользователя

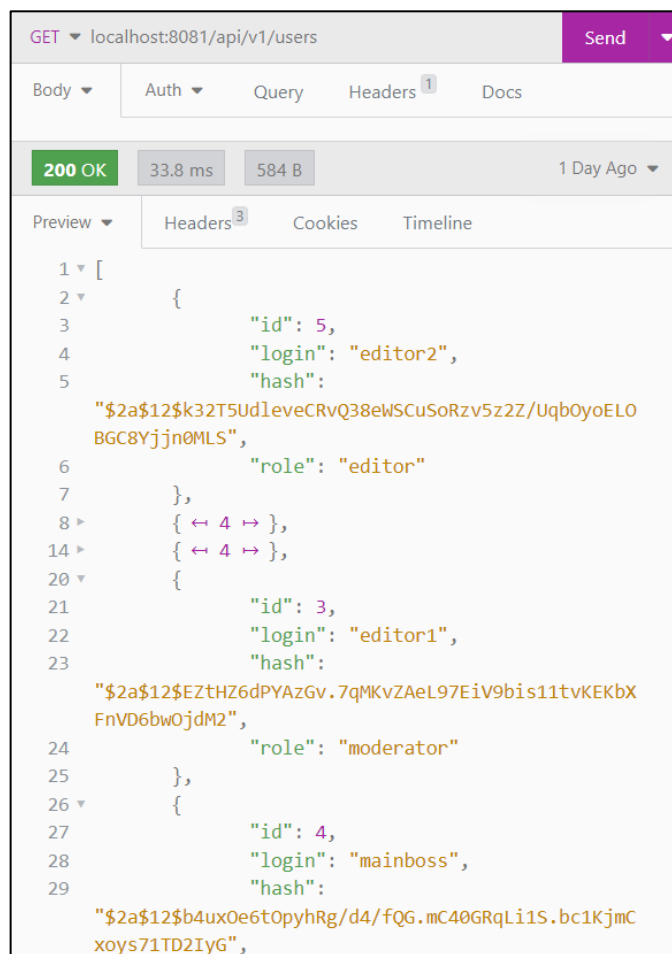


Рисунок 25 – Результат выполнения запроса на получение списка всех пользователей

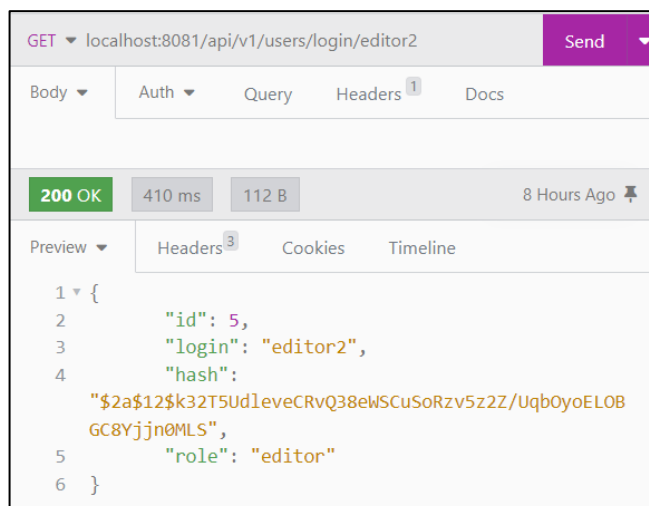


Рисунок 26 - Результат выполнения запроса на получение пользователя по логину

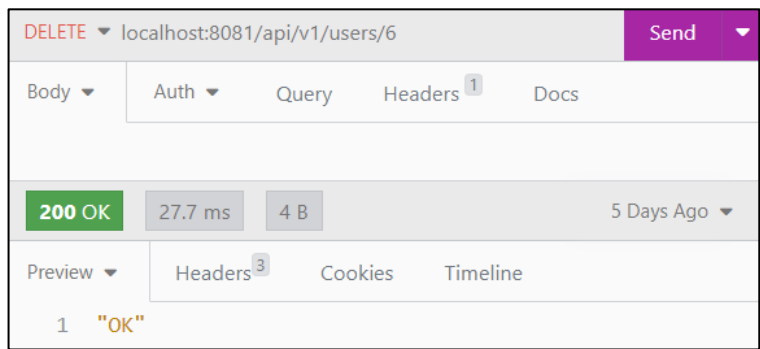


Рисунок 27 – Результат выполнения запроса удаления пользователя



Рисунок 28 – Результат выполнения запроса добавления блока «Важное»

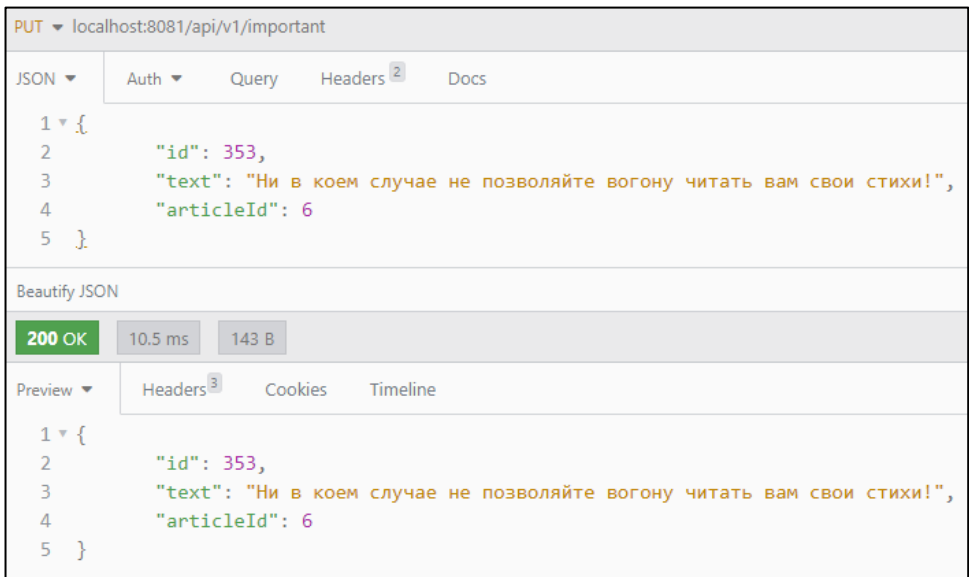


Рисунок 29 – Результат выполнения запроса на изменение блока «Важное»

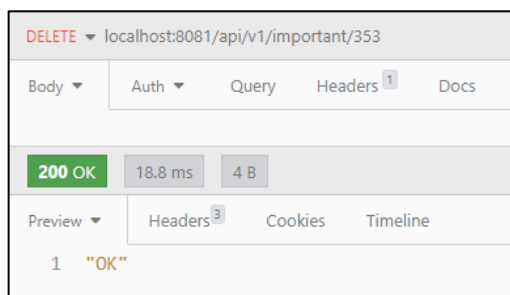


Рисунок 30 – Результат выполнения запроса удаления блока «Важное»

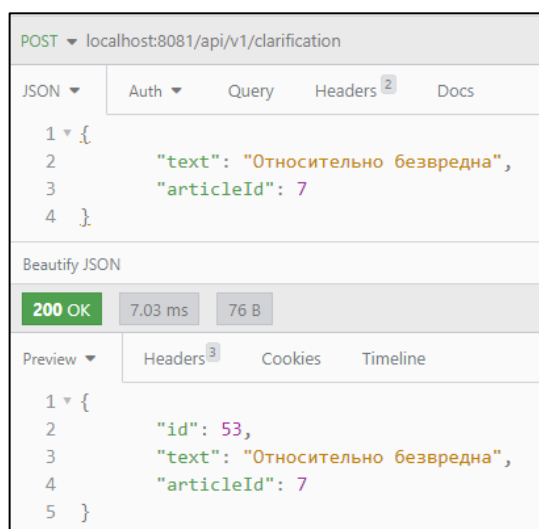


Рисунок 31 – Результат выполнения запроса добавления блока «Требуется уточнения»

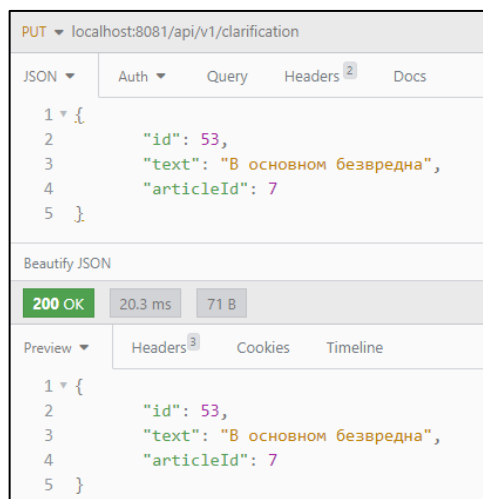


Рисунок 32 – Результат выполнения запроса изменения блока «Требуется уточнения»

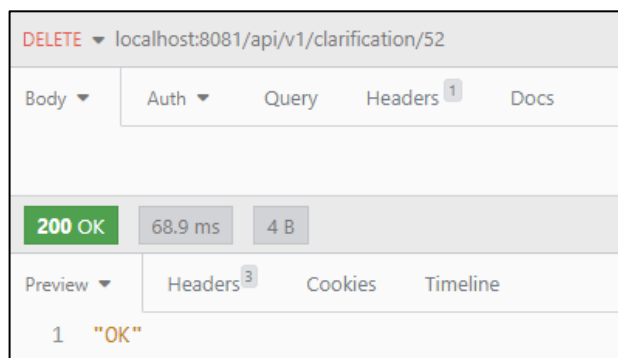


Рисунок 33 – Результат выполнения запроса удаления блока «Требуется уточнения»

3.2 Тестирование мобильного клиента

Тестирование мобильного клиента проведено в ручном режиме. Воспроизведены все пользовательские сценарии, выявленные при анализе системы и указанные в постановке задачи, и проверена их работоспособность. Подробное описание проведенных испытаний приведено ниже.

3.2.1 Тест-кейс «Регистрация»

Обязательные предусловия:

- пользователь находится на основном экране приложения;
- пользователь имеет стабильный доступ в интернет.

Таблица 29 – Шаги тест-кейса «Регистрация»

№	Описание шага	Реакция системы
1	Нажать на кнопку «Назад».	Отобразить страницу авторизации.
2	Нажать на кнопку «Регистрация», расположенную на странице авторизации.	Отобразить страницу регистрации.
3	Ввести логин «newUser» и нажать на кнопку «Регистрация».	Отобразить диалоговое окно с уведомлением «Чего-то не хватает. Необходимо указать логин, создать пароль и подтвердить его» и кнопкой «ОК».
4	Нажать на кнопку «ОК».	Заккрыть диалоговое окно.
5	Ввести логин «newUser». Ввести пароль и	Выполнить переход к основному

№	Описание шага	Реакция системы
	повтор пароля «newPassword». Нажать кнопку «Регистрация»	экрану. Отобразить следующие элементы: <ul style="list-style-type: none"> – кнопка «Назад»; – кнопка «Синхронизировать базу данных»; – кнопка перехода к списку «Избранное»; – строка поиска.

3.2.2 Тест-кейс «Авторизация»

Обязательные предусловия:

- в системе существуют записи пользователей «newUser» с правами читателя, «administrator» с правами модератора и «editor3» с правами редактора;
- неавторизованный пользователь находится на основной странице приложения.

Таблица 30 – Шаги тест-кейса «Авторизация»

№	Описание шага	Реакция системы
1	Нажать на кнопку «Назад».	Отобразить страницу авторизации.
2	Ввести логин «newUser» и пароль «newPassword», нажать на кнопку «Войти».	Отобразить основной экран со следующими элементами: <ul style="list-style-type: none"> – кнопка «Назад»; – кнопка «Синхронизировать базу данных»; – кнопка перехода к списку «Избранное»; строка поиска.
3	Нажать на кнопку «Назад».	Отобразить страницу авторизации.
4	Ввести логин «administrator», пароль «adminpsw» и нажать на кнопку «Войти».	Отобразить основной экран со следующими элементами: <ul style="list-style-type: none"> – кнопка «Назад»; – кнопка «Синхронизировать базу данных»; – кнопка перехода к списку «Избранное»; – кнопка «Добавить статью»; – строка поиска.
5	Нажать на кнопку «Назад».	Отобразить страницу авторизации.
6	Ввести логин «editor3», пароль «pswedt» и	Отобразить основной экран со

№	Описание шага	Реакция системы
	нажать на кнопку «Войти».	следующими элементами: <ul style="list-style-type: none"> – кнопка «Назад»; – кнопка «Синхронизировать базу данных»; – кнопка перехода к списку «Избранное»; – кнопка «Добавить статью»; строка поиска.

3.2.2 Тест-кейс «Поиск и просмотр статьи»

Обязательные предусловия:

- выполнена авторизация от лица пользователя с правами модератора;
- статья «Земля» не добавлена в список «Избранное».

Таблица 31 – Шаги тест-кейса «Поиск и просмотр статьи»

№	Описание шага	Реакция системы
1	Нажать на кнопку «Поиск».	Отобразить окно с уведомлением «Пустой запрос. Для выполнения поиска необходимо ввести запрос»
2	Нажать на кнопку «ОК» или на область экрана вне открытого диалогового окна.	Закрыть диалоговое окно.
3	Ввести текст «Земля» в поисковую строку и нажать кнопку «Поиск».	Отобразить результаты поиска – список из одной статьи «Земля».
4	Нажать на статью «Земля» в списке.	Отобразить страницу просмотра статьи. Страница содержит заголовок «Земля», текст «Безвредна», блок типа «Требуется уточнения» с текстом «В основном безвредна» и кнопки управления: «Назад», «Удалить», «Редактировать», «Добавить в избранное».

3.2.3 Тест-кейс «Работа со списком избранных статей»

Обязательные предусловия:

- пользователь находится на странице просмотра статьи «Земля»;
- пользователь не имеет ни одной статьи в списке «Избранное».

Таблица 32 – Шаги тест-кейса «Работа со списком избранных статей»

№	Описание шага	Реакция системы
1	Нажать на кнопку «Добавить в избранное».	Заменить кнопку «Добавить в избранное» на «Удалить из избранного»
2	Перейти к основному экрану приложения.	Отобразить основной экран.
3	Нажать на кнопку перехода к странице «Избранное».	Отобразить страницу «Избранное» и список статей, состоящий из одной статьи «Земля».
4	Нажать на статью «Земля» в списке.	Отобразить страницу просмотра статьи с элементами в соответствии с ролью и кнопку «Удалить».
5	Нажать на кнопку «Удалить из избранного».	Заменить кнопку «Удалить из избранного» на «Добавить в избранное».
6	Перейти на страницу «Избранное».	Отобразить страницу «Избранное» с сообщением «Нет избранных статей».

3.2.6 Тест-кейс «Создание новой статьи»

Обязательные предусловия:

- выполнена авторизация от лица пользователя с ролью «Редактор» или «Модератор»;
- пользователь находится на основной странице приложения.

Таблица 33 – Шаги тест-кейса «Создание новой статьи»

№	Описание шага	Реакция системы
1	Нажать на кнопку «Создать статью».	Отобразить страницу «Новая статья».
2	Нажать на кнопку «Сохранить».	Отобразить диалоговое окно с уведомлением «Чего-то не хватает. Для сохранения необходимо задать название статьи и ввести её текст» и кнопкой «ОК».
3	Нажать на кнопку «ОК» или на область вне уведомления.	Закрыть диалоговое окно.
4	В поле «Заголовок» указать текст «Тестовая статья». В поле «Текст статьи» ввести текст: «Рыбате́кст используется дизайнерами, проектировщиками и фронтендерами, когда нужно быстро	Отобразить на форме введенные пользователем данные.

– пользователь находится на странице просмотра статьи «Тестовая статья», содержащей, помимо основного текста, блоки «Важное» и «Требуется уточнения» с текстом «Lorem ipsum dolor sit amet».

Таблица 34 – Шаги тест-кейса «Редактирование статьи в локальной базе данных»

№	Описание шага	Реакция системы
1	Нажать на кнопку «Изменить статью».	Отобразить страницу «Редактирование» с предзаполнением данными статьи «Тестовая статья».
2	Ввести новые значения элементов: Название статьи – «Изменение». Текст статьи – «Новый текст». Текст блока «Важное» - «Новое важное». Текст блока «Требуется уточнения» - «Новое неточное».	Применить изменения пользователя к элементам формы.
3	Нажать на кнопку «Сохранить».	Сохранить изменения в статье. Закрыть страницу редактирования. Отобразить основной экран приложения.
4	Выполнить поиск статей по названию «Изменение».	Отобразить результаты поиска, среди которых есть элемент «Изменение».
5	Перейти на страницу просмотра статьи «Изменение».	Отобразить сведения о выбранной статье с учетом изменений, сделанных пользователем на шаге 2.

3.2.8 Тест-кейс «Отправка статьи в глобальную базу данных»

Обязательные предусловия:

- выполнена авторизация от лица пользователя с ролью «Редактор» или «Модератор»;
- пользователь находится на странице просмотра статьи «Тестовая статья», отредактированной ранее;
- имеется устойчивое Интернет-соединение.

Таблица 35 – Шаги тест-кейса «Отправка статьи в глобальную базу данных»

№	Описание шага	Реакция системы
1	Нажать на кнопку «Отправить статью».	Отобразить диалоговое окно с текстом «Отправить статью. Для выполнения данной операции необходимо убедиться в стабильности Интернет-соединения. Продолжить?» и кнопками «Да» и «Нет».
2	Нажать на кнопку «Да».	Выполнить запрос обновления статьи в глобальной БД. Закрывать диалоговое окно. Отобразить уведомление с текстом «Отправка статьи. Статья успешно обновлена» и кнопкой «ОК».
3	Нажать на кнопку «ОК» или на область экрана вне открытого диалогового окна.	Закрывать окно с уведомлением.

3.2.9 Тест-кейс «Удаление статьи в глобальной базе данных»

Обязательные предусловия:

- выполнена авторизация от лица пользователя с ролью «Модератор»;
- пользователь находится на странице просмотра статьи «Тестовая статья»;
- имеется устойчивое Интернет-соединение.

Таблица 36 – Шаги тест-кейса «Удаление статьи в глобальной базе данных»

№	Описание шага	Реакция системы
1	Нажать на кнопку «Удалить статью».	Отобразить диалоговое окно с текстом «Удаление статьи. Удаление статьи является необратимой операцией. Удалить?» и кнопками «Да» и «Нет».
2	Нажать на кнопку «Да».	Выполнить запрос удаления статьи из глобальной БД. Закрывать диалоговое окно. Отобразить уведомление с текстом «Удаление статьи. Статья успешно удалена» и кнопкой «ОК».
3	Нажать на кнопку «ОК» или на область	Закрывать окно с уведомлением.

№	Описание шага	Реакция системы
	экрана вне открытого диалогового окна.	

3.2.10 Тест-кейс «Изменение прав пользователя»

Требуемые предусловия:

- выполнена авторизация от лица пользователя с ролью «Модератор»;
- пользователь находится на основной странице приложения;
- создана запись пользователя «editor3» с ролью «Редактор».

Таблица 37 – Шаги тест-кейса «Изменение прав пользователя»

№	Описание шага	Реакция системы
1	Нажать на поисковую строку.	Отобразить параметры поиска – кнопки «статьи» и «пользователи». Активна кнопка «статьи».
2	Нажать на кнопку «пользователи».	Активировать кнопку «пользователи», сбросить кнопку «статьи».
3	Ввести текст «editor3» в строку поиска и нажать кнопку «Поиск».	Отобразить результаты поиска – список пользователей, среди которых есть элемент с записью пользователя «editor3». Элемент списка отображает логин, статус «Редактор» и содержит кнопки «Удалить пользователя», «Выдать права читателя» и «Выдать права модератора».
4	Нажать на кнопку «Выдать права читателя».	Изменить роль пользователя на «Читатель». Изменить роль пользователя в списке результатов. Скрыть кнопку «Выдать права читателя», отобразить кнопку «Выдать права редактора». Отобразить окно с уведомлением «Изменение роли. Роль пользователя editor3 изменена на «Читатель»» и кнопкой «ОК».
5	Нажать на кнопку «ОК» или на область вне открытого диалогового окна.	Закрыть диалоговое окно.
6	Нажать на кнопку «Выдать права модератора».	Изменить роль пользователя на «Модератор». Изменить роль пользователя в списке результатов. Скрыть кнопку «Выдать права модератора», отобразить кнопку «Выдать права читателя». Отобразить окно с уведомлением «Изменение роли. Роль пользователя editor3 изменена на «Модератор»» и кнопкой «ОК».
7	Нажать на кнопку «ОК» или на	Закрыть диалоговое окно.

№	Описание шага	Реакция системы
	область вне открытого диалогового окна.	

3.2.11 Тест-кейс «Удаление пользователя»

Обязательные предусловия:

- выполнена авторизация от лица пользователя с ролью «Модератор»;
- пользователь находится на основной странице приложения;
- создана запись пользователя «editor3» с ролью «Редактор».

Таблица 38 – Шаги тест-кейса «Удаление пользователя»

№	Описание шага	Реакция системы
1	Выполнить поиск пользователя с логином «editor3».	Отобразить результат поиска – список пользователей, среди которых есть элемент с записью пользователя «editor3». Элемент списка отображает логин, статус «Редактор» и содержит кнопки «Удалить пользователя», «Выдать права читателя» и «Выдать права модератора».
2	Нажать на кнопку «Удалить пользователя».	Отобразить окно подтверждения с текстом «Удаление пользователя. Пользователь editor3 удалён» и кнопкой «ОК».
3	Нажать на кнопку «ОК» или на область вне открытого диалогового окна.	Заккрыть диалоговое окно.

4 Безопасность жизнедеятельности

Воздействие персональных компьютеров на здоровье человека — это актуальная проблема, которая требует внимания и изучения.

Основными вредными факторами, влияющие на состояние здоровья людей, работающих за мобильными устройствами, являются:

- сидячее положение в течение длительного времени. Из-за стеснённой позы возникают напряжение шеи, мышц головы, рук и плеч, проблемы с позвоночником;
- воздействие электромагнитного излучения устройства. Считается, что электромагнитное излучение может вызвать расстройства нервной системы, снижение иммунитета, расстройства сердечно-сосудистой системы;
- утомление глаз, нагрузка на зрение. Изображение на экране дисплея складывается не из непрерывных линий, как на бумаге, а из отдельных точек, к тому же светящихся и мерцающих;
- влияние на психику. Длительное пребывание за экраном монитора, особенно детей, отрицательно влияет на психическое состояние здоровья.

4.1 Классификация и категорирование помещения для проведения работ

Производственные помещения по степени опасности поражения людей электрическим током подразделяются на три категории: без повышенной опасности, с повышенной опасностью, особо опасные.

Помещения без повышенной опасности — это сухие не запылённые помещения с нормальной температурой воздуха и изолирующим (не токопроводящим) полом. К ним относятся кабинеты, залы, лаборатории,

производственные участки, в которых отсутствуют признаки химически активной среды.

Помещения с повышенной опасностью характеризуются следующими признаками:

- повышенная температура (температура воздуха длительно превышает 35 °С или кратковременно — 40 °С независимо от времени года и различных тепловых излучений);
- повышенная (выше 75 %) относительная влажность воздуха;
- наличие токопроводящей пыли;
- токопроводящий пол (металлический, земляной, железобетонный и т. п.);
- возможность одновременного прикосновения человека к заземлённым металлоконструкциям сооружений, машин и механизмов, с одной стороны, и металлическим корпусам — с другой.

Помещения особо опасные — в которых выполняется одно из условий:

- особая сырость (относительная влажность около 100 %, когда потолок, стены, и предметы покрыты влагой);
- химически активная среда (в помещении постоянно или в течение длительного времени выделяются пары и образуются отложения, которые разрушающе действуют на изоляцию и токопроводящие части оборудования);
- одновременное наличие двух или более признаков, характеризующих признаки помещений с повышенной опасностью.

Офисные помещения относятся к помещениям без повышенной опасности.

По пожарной и взрывопожарной опасности помещения производственного и складского назначения подразделяются на следующие категории:

- А, повышенная взрывопожароопасность;
- Б, взрывопожароопасность;
- В1-В4, пожароопасность;
- Г, умеренная пожароопасность;
- Д, пониженная пожароопасность.

Помещения иного назначения разделению на категории не подлежат. Офисные помещения можно отнести к категории Д или не категоризировать вовсе.

4.2 Санитарно-гигиеническая характеристика

Работу можно разделить на категории. Категория I подразумевает легкую физическую деятельность, в ходе которой энергозатраты не превышают 150 кКал/ч. К категории IA относятся работы, сопровождающиеся незначительным физическим напряжением мышц. Категория IB связана с небольшими нагрузками, ходьбой.

Работа программиста относится к категории IA и включает в себя следующие требования и нормы, определенные СанПиН [8]:

- поддерживать температуру воздуха в пределах от 22 до 24 градусов;
- верхний предел относительной влажности воздуха не превышает отметку в 75%;
- скорость движения воздуха не больше 0,1 м/с.

В ходе разработки проекта человек в основном смотрит в монитор и подвергает свои глаза сильному напряжению. Для профилактики проблем со зрением при работе за компьютером рекомендуется выполнять следующие правила:

- правильно организовать рабочее место. Монитор должен быть размещён на расстоянии вытянутой руки, а его высота настроена так, чтобы

он был немного ниже уровня глаз. Выбор рабочего кресла также важен: оно должно быть удобным и с возможностью регулирования, а его спинка должна поддерживать позвоночник;

- делать короткие перерывы каждые 40–45 минут. Во время перерыва нужно полностью отвлечься от компьютера: можно сходить на свежий воздух, попить воды или просто посмотреть в окно, фокусируясь на предметах, которые находятся на дальнем расстоянии;

- делать гимнастику для глаз. Это наиболее эффективный способ защитить зрение от ухудшения. Вот несколько базовых упражнений: в течение минуты, не вращая головой, водите глазами в разные стороны (вниз, вверх, вправо, влево); сильно-сильно зажмурьтесь, а затем широко распахните глаза; быстро моргайте на протяжении 1 минуты; фиксируйте зрение поочерёдно;

- использовать очки, защищающие от компьютерного излучения;

- пить больше воды и не забывать моргать, чтобы увлажнять глаза, предотвратить ощущение сухости и «песка» в них.

Одной из причин усталости и напряжения глаз является плохое освещение рабочего места. Правила, которые необходимо соблюдать при работе за компьютером:

- не допускать работу в темном помещении без использования искусственных источников света;

- не допускать чрезмерного попадания солнечных лучей на монитор, т.к. это может затруднить чтение текста на нем, а также может привести к выцветанию цветов и искаженному отображению картинки;

- сочетать искусственное освещение и естественное.

Также в рабочих помещениях не должны нарушаться пределы допустимого уровня шума и вибраций.

4.3 Электробезопасность

В соответствии с СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы» [9] помещения, где размещаются рабочие места с ПЭВМ, должны быть оборудованы защитным заземлением в соответствии с техническими требованиями по эксплуатации электроустановок и вычислительной техники.

Рабочие места с ПЭВМ не следует размещать вблизи силовых кабелей и вводов, высоковольтных трансформаторов, технологического оборудования, создающего помехи в работе ПЭВМ.

Поскольку непосредственно на ПЭВМ должно подаваться стабилизированное электропитание подачу электроэнергии в компьютерные помещения следует осуществлять от отдельного независимого источника питания.

В дальнейшем при эксплуатации ПЭВМ необходимо соблюдать следующие рекомендации [10]:

- постоянно контролировать надежность соединения контактов трехпроводных розеток;
- подключать дисплей (при наличии только двухпроводной однофазной сети) рекомендуется через согласующее устройство;
- не ставить системный блок в зоне повышенной влажности и повышенного содержания пыли, на пол, у ног оператора;
- во избежание поражения электрическим током запрещается прикасаться к задней панели системного блока;
- необходимо устанавливать ПЭВМ только на жестко закрепленной подставке, исключающей даже случайное сотрясение системного блока;

– не рекомендуется установка ПЭВМ и его клавиатуры на поверхности, накапливающие статическое электричество (органическое стекло и полированные лаковые поверхности).

4.4 Пожарная безопасность

При работе с компьютером необходимо соблюдать правила пожарной безопасности – комплекса мер, направленных на предотвращение угрозы возникновения пожара. Это поможет избежать возгорания в помещении или поражения человека током.

Некоторые правила:

- не вставлять вилку электропровода в розетку, которая имеет неисправный вид внешне. Во время работы нужно следить за степенью нагревания вилки;
- не прикасаться к разъёмам, проводам электропитания, заземляющим устройствам, тыльной стороне монитора;
- не устранять неисправности техники самостоятельно;
- не заставлять проходы и входы/выходы из помещения, чтобы не препятствовать эвакуации людей из класса;
- не включать несколько мощных электроприборов в одну розетку.

В случае возникновения возгорания необходимо:

- отключить всё электронное оборудование;
- принять оговоренные заранее меры для ликвидации возгорания;
- эвакуировать материальные ценности по возможности;
- сообщить о пожаре в соответствующие службы — дежурному, руководству, на контрольный пункт.

Для тушения пожара необходимо использовать первичные средства пожаротушения. К ним относятся огнетушители, пожарные краны и щиты.

При выборе огнетушителя следует учитывать тип вещества или материала, который подвергся возгоранию

4.5 Расчет общего искусственного освещения помещения с ПЭВМ

Как было описано выше, в ходе длительной работы за компьютером, у человека может возникнуть усталость и слезоточивость глаз, а в дальнейшем все это может привести к синдрому сухого глаза. Одной из причин является недостаточное освещение рабочего места. Поэтому в ходе работы следует позаботиться об освещении помещения.

Для того, чтобы найти необходимое количество светильников, можно воспользоваться следующей формулой:

$$N = \frac{E_n \cdot S \cdot K_z \cdot Z}{n \cdot \Phi},$$

где N – число светильников в помещении;

E_n – минимальная освещенность помещения;

S – площадь помещения;

K_z – коэффициент запаса;

Z – коэффициент неравномерности освещения;

n – коэффициент использования светового потока;

Φ – световой поток.

По всем нормам, минимальная освещенность помещения E_n должна быть равна примерно 300 лк. Значение коэффициента запаса для сухих помещений принимается равной 1.4. Для освещения помещения будут применяться люминесцентные лампы, для которые коэффициент неравномерности будет равен 1.1. Размеры помещения – 3 на 3 метра. По этим данным, можно сразу вычислить площадь, она будет равна:

$$S = 3 \text{ м} \cdot 3 \text{ м} = 9 \text{ м}^2.$$

После этого можно приступить к расчету индекса помещения, который находится по формуле:

$$i = \frac{S}{(hp \cdot (W+L))},$$

где i – индекс помещения;

hp – высота светильников над поверхностью стола;

W – ширина помещения;

L – длина помещения.

Размеры комнаты 3 на 3 метра, высота светильников над столом составляет 1.8 метра.

$$i = \frac{3 \cdot 3}{1.8 \cdot (3+3)} = 0.8.$$

Коэффициент отражения потолка как белой поверхности R_p – 70%. Коэффициент отражения стен – светлых поверхностей – R_c равен 50%. По таблице был определен коэффициент использования светового потока n , он равен 0,27.

Световой поток для ламп ЛБ40 равен 2800 лм.

Соответственно, количество светильников типа ЛПО с двумя лампами:

$$N = \frac{300 \cdot 9 \cdot 1.4 \cdot 1.1}{0.27 \cdot 2800 \cdot 2} = 3.$$

Для освещения комнаты необходимо три светильника типа ЛПО.

Заключение

В результате выполнения бакалаврской работы решены следующие задачи: анализ предметной области, сравнительный анализ аналогов, выявление функциональных и нефункциональных требований к системе. По полученным результатам сформулированы и формализованы требования к системе, спроектирована архитектура, поставлены задачи на разработку клиентской и серверной частей системы, выбраны инструменты разработки.

Были освоены и закреплены основные понятия и технологии RESTful веб-сервисов и мобильных приложений для платформы Android. Изучены основные инструменты разработки серверных приложений на языке программирования Java, такие, как фреймворк Spring Framework, СУБД PostgreSQL и IDE IntelliJ IDEA. Освоены инструменты разработки мобильных приложений для устройств, работающих на платформе Android – IDE Android Studio и СУБД SQLite. В результате разработана справочная система, состоящая из мобильного клиента, с которым непосредственно взаимодействует пользователь, серверной части, содержащей базу данных и веб-сервера, связывающего клиент с сервером.

Список использованных источников

1. Reddit [Электронный ресурс] – URL: https://www.reddit.com/r/DontPanic/comments/ayjug7/real_life_hitch_hikers_guide_to_the_galaxy/?rdt=48742 (дата обращения: 23.08.2023)
2. Вигерс К., Битти Д. Разработка требований к программному обеспечению. 3-е изд., дополненное / Пер. с англ. – М. : Издательство «Русская редакция» ; СПб. : БХВ-Петербург, 2014 – 736 стр.
3. PostgreSQL [Электронный ресурс] – URL: <https://www.postgresql.org/> (дата обращения: 20.04.2024)
4. SQLite - что это: обзор на реляционную базу данных с примерами [Электронный ресурс] – URL: <https://blog.skillfactory.ru/glossary/sqlite/> (дата обращения: 20.04.2024)
5. Spring | Веб Applications [Электронный ресурс] – URL: <https://spring.io/веб-applications> (дата обращения: 22.04.2024)
6. Kotlin Docs | Kotlin Documentation [Электронный ресурс] – URL: <https://kotlinlang.org/docs/home.html> (дата обращения: 22.04.2024)
7. Develop for Android | Android Developers [Электронный ресурс] – URL: <https://developer.android.com/develop> (дата обращения: 10.05.2024)
8. СанПиН 1.2.3685-21 "Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания" [Электронный ресурс] – URL: <https://www.opengost.ru/> (дата обращения: 23.05.2024 г)
9. СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к ПЭВМ и организации работы» [Электронный ресурс] – URL: <http://www.consultant.ru/> (дата обращения: 25.05.2024 г)
10. Электробезопасность при работе оператора ПЭВМ: методические указания по выполнению в дипломных проектах и работах раздела

«Безопасность объектов» /сост.:И.О. Протодяконов, В.И. Сарже, О.И. Протодяконова – СПб.: СПбГТУРП, 2013. –13 с.

					ПГУ 1.090301.16.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		72

Приложение А

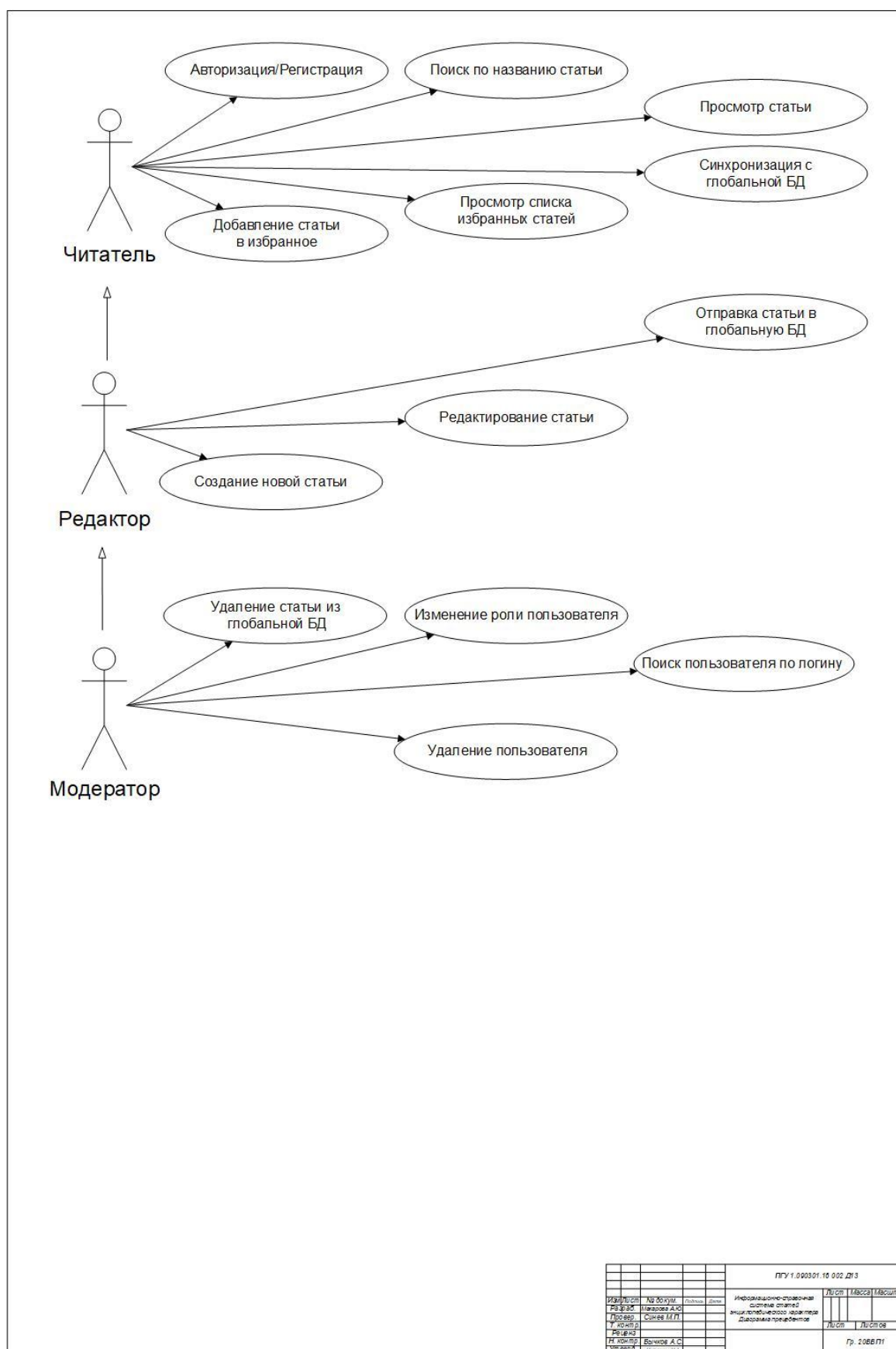
(Обязательное)

Информационно-справочная система статей энциклопедического характера

Графическая часть

					ПГУ 1.090301.16.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		73

Приложение А.1 – Диаграмма прецедентов



				ПГУ 1.090301.16.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Информационно-справочная система статей	Лист	Листов
Ред.	1	1	Иванов А.С.		инициализация системы	1	1
Уд.			Сидоров М.П.		Диаграмма прецедентов		
Ред.							
Изм.			Бочков А.С.				
Уд.			Михайлов И.В.				

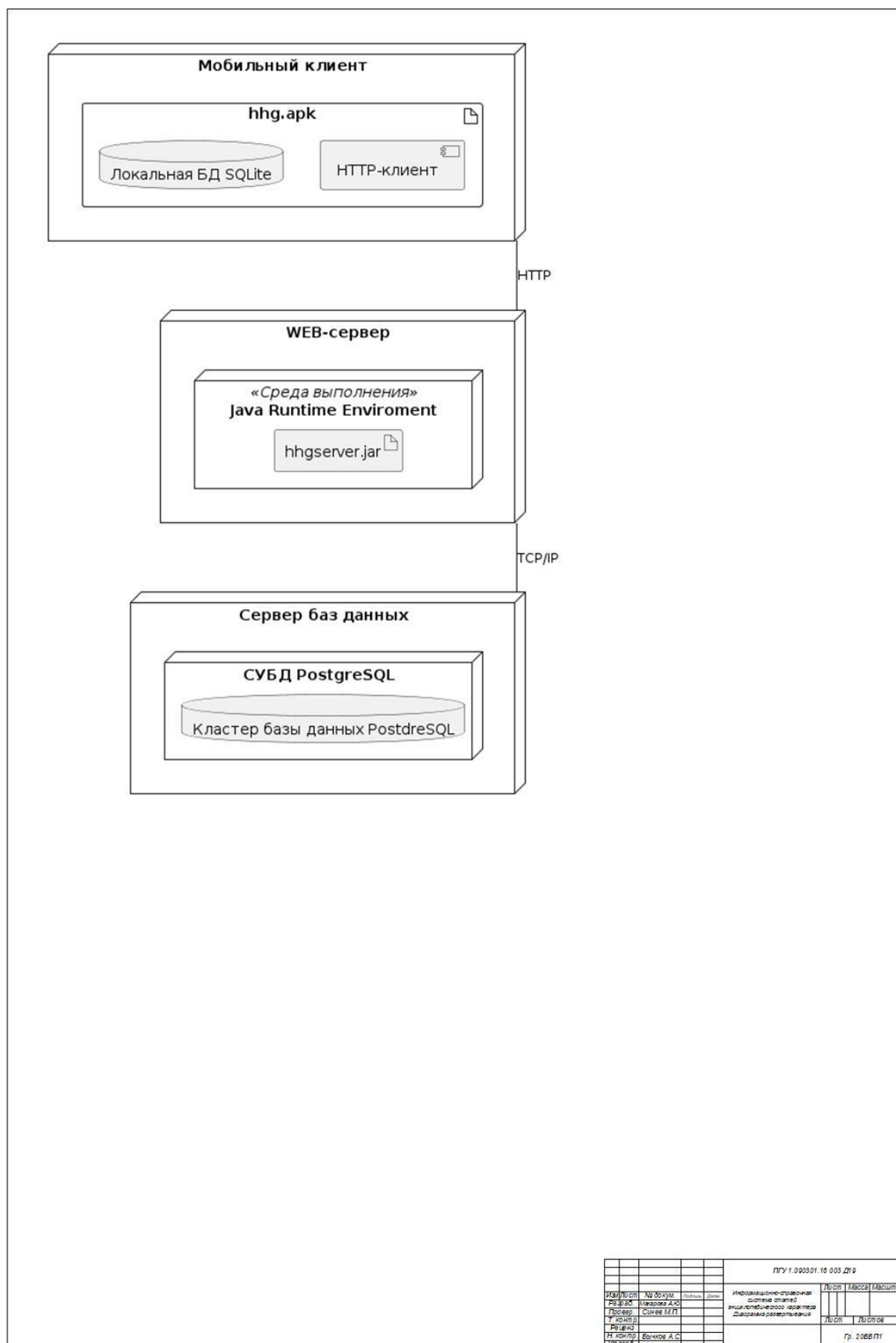
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.16.001 ПЗ

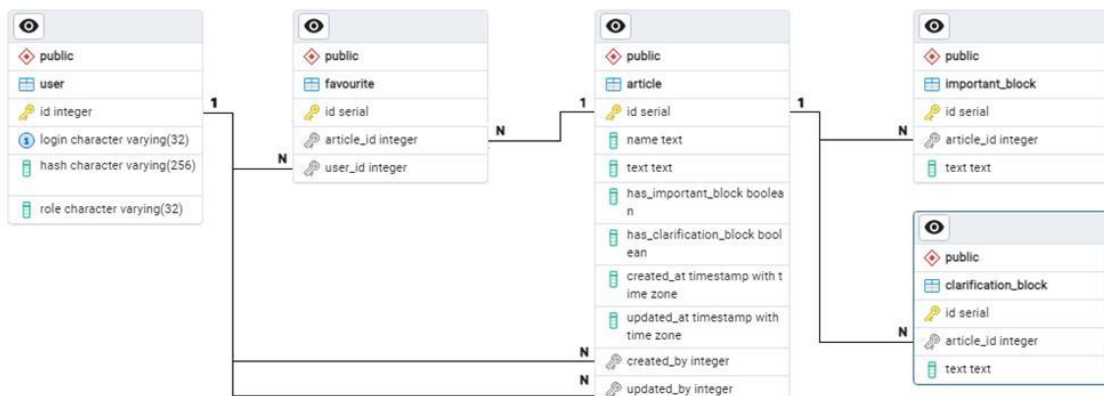
Лист

74

Приложение А.2 – Диаграмма развертывания



Приложение А.3 – Физическая схема базы данных



				ПГУ 1.090301.16.001 ПЗ			
ИЗДАТЕЛЬ	№ докум.	Подпись	Дата	Информационно-справочная служба «ПГУ»			
Проверено	Составлено	Исполнено		Информационно-справочная служба «ПГУ»			
Исполнено	Составлено	Исполнено		Информационно-справочная служба «ПГУ»			
Исполнено	Составлено	Исполнено		Информационно-справочная служба «ПГУ»			
Исполнено	Составлено	Исполнено		Информационно-справочная служба «ПГУ»			
Исполнено	Составлено	Исполнено		Информационно-справочная служба «ПГУ»			
Исполнено	Составлено	Исполнено		Информационно-справочная служба «ПГУ»			
Исполнено	Составлено	Исполнено		Информационно-справочная служба «ПГУ»			
Исполнено	Составлено	Исполнено		Информационно-справочная служба «ПГУ»			

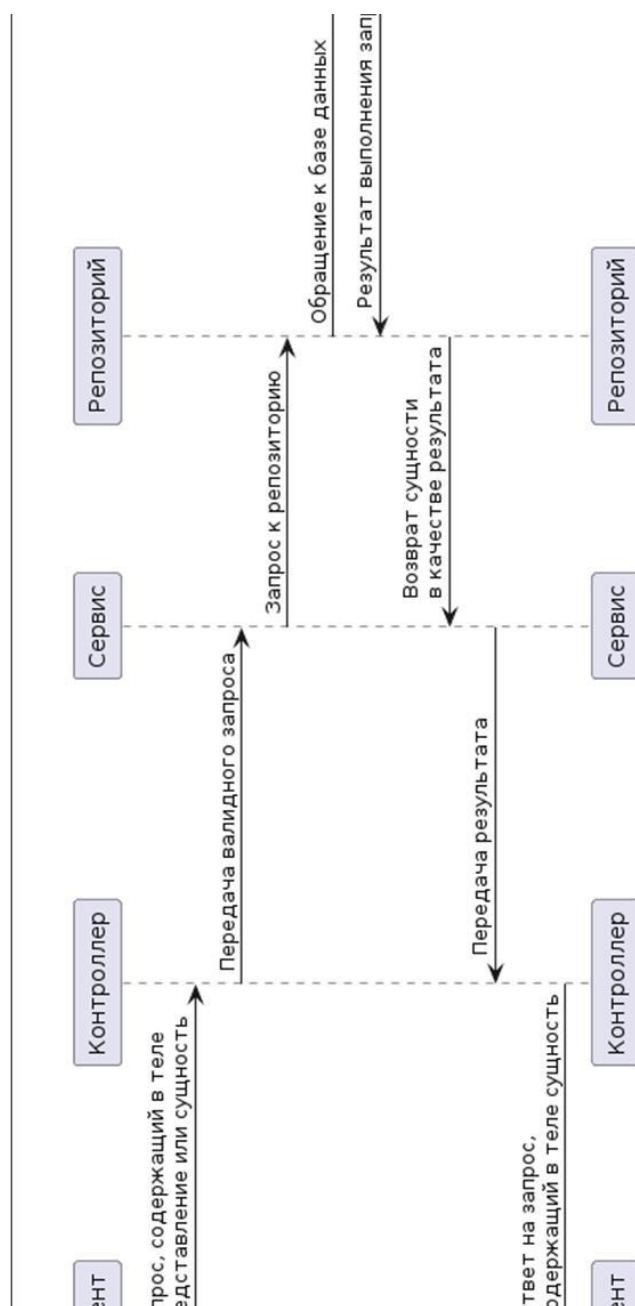
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.16.001 ПЗ

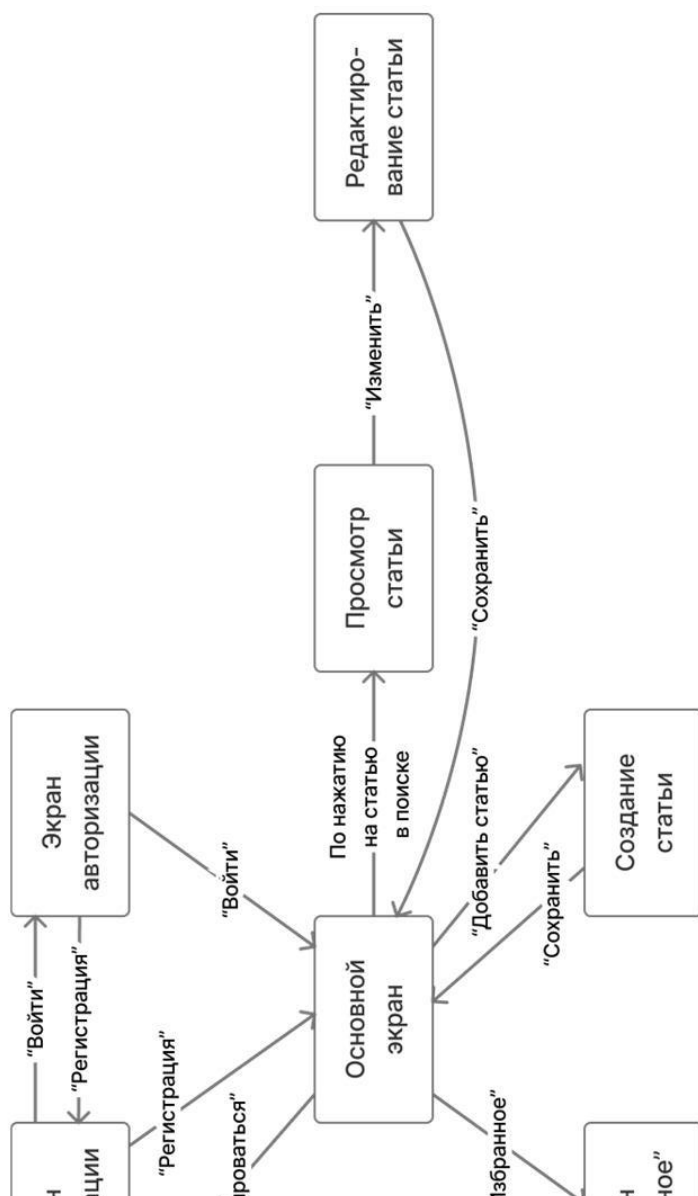
Лист

76

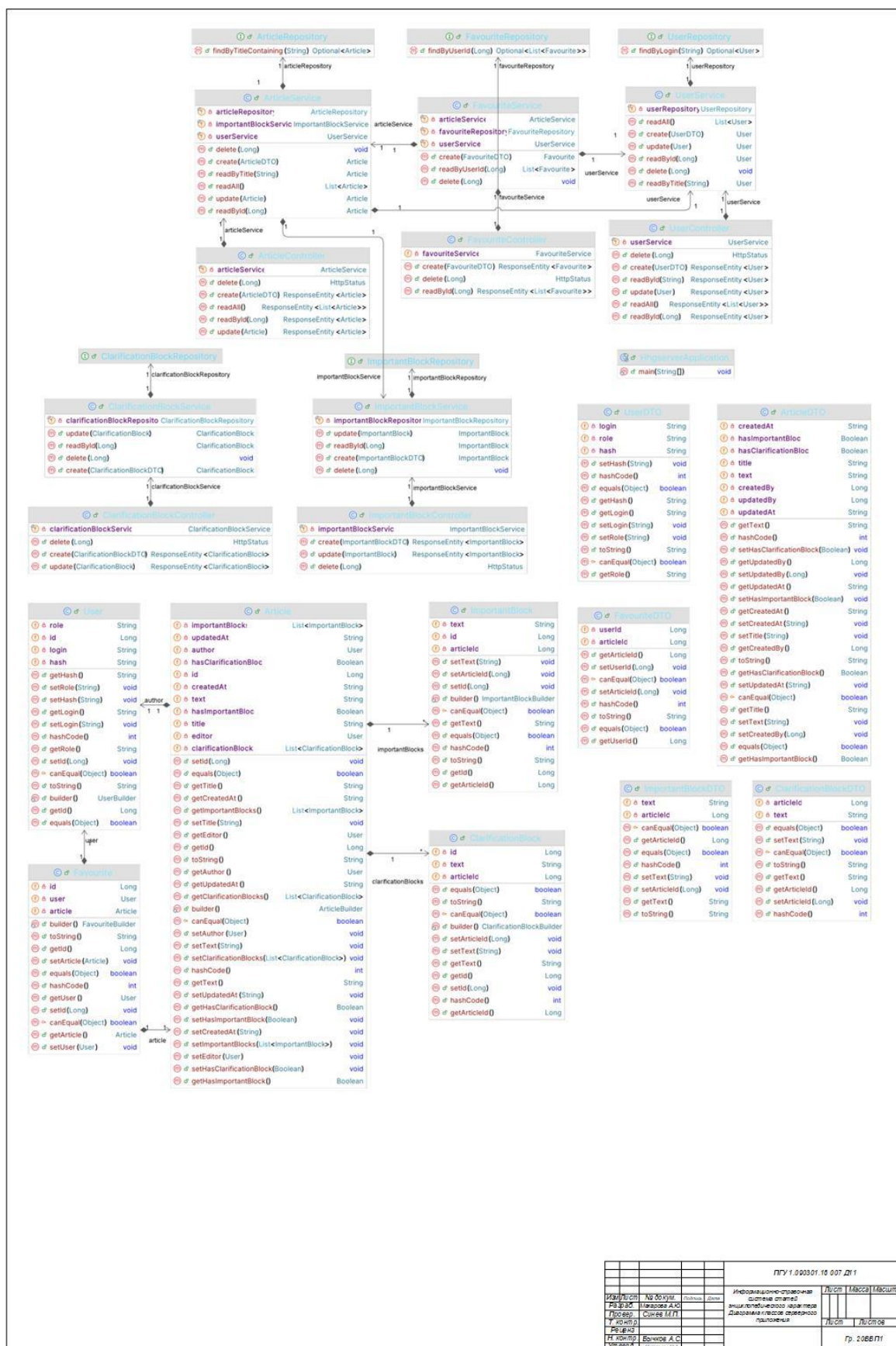
Приложение А.4 – Диаграмма последовательности



Приложение А.5 – Карта экранов



Приложение А.6 – Диаграмма классов сервера



Приложение Б.1

Файл User.java

```
package com.example.hhgserver.entity;
import jakarta.persistence.*;
import lombok.*;
@Entity
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
@Table(name = "users")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private Long id;
    @Column(unique = true)
    private String login;
    private String hash;
    private String role;}
```

Приложение Б.2

Файл UserDTO.java

```
package com.example.hhgserver.dto;
import lombok.Data;
@Data
public class UserDTO {
    private String login;
    private String hash;
    private String role;}
```

Приложение Б.3

Файл UserRepository.java

```
package com.example.hhgserver.repository;
import com.example.hhgserver.entity.Article;
import com.example.hhgserver.entity.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import java.util.List;
import java.util.Optional;
@Repository
public interface UserRepository extends JpaRepository<User,
Long> {
    Optional<User> findByLogin(String login);}
```

Приложение Б.4

Файл UserService.java

```
package com.example.hhgserver.service;
import com.example.hhgserver.dto.UserDTO;
import com.example.hhgserver.entity.User;
import com.example.hhgserver.repository.UserRepository;
import lombok.AllArgsConstructor;
import org.springframework.stereotype.Service;
import java.util.List;
@Service
@AllArgsConstructor
public class UserService {
    private final UserRepository userRepository;
    public User create(UserDTO dto) {
        return userRepository.save(User.builder()
            .login(dto.getLogin())
            .hash(dto.getHash())
            .role(dto.getRole())
            .build());
    }
    public List<User> readAll() {
        return userRepository.findAll();
    }
    public User readById(Long id) {
        return userRepository.findById(id).orElseThrow(() -> new
RuntimeException("Пользователь не найден"));
    }
    public User readByTitle(String login) {
        return userRepository.findByLogin(login).orElseThrow(()
-> new RuntimeException("Пользователь не найден"));
    }

    public User update(User user) {
        return userRepository.save(user);
    }

    public void delete(Long id) {
        userRepository.deleteById(id);
    }
}
```

Приложение Б.5

Файл UserController.java

```
package com.example.hhgserver.controller;
import com.example.hhgserver.dto.UserDTO;
import com.example.hhgserver.entity.User;
import com.example.hhgserver.service.UserService;
import lombok.AllArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.util.List;
```

```

@RestController
@RequestMapping("/api/v1/users")
@AllArgsConstructor
public class UserController {
    private final UserService userService;
    @PostMapping
    public ResponseEntity<User> create(@RequestBody UserDTO dto)
    {
        return new ResponseEntity<>(userService.create(dto),
        HttpStatus.OK);
    }

    @GetMapping
    public ResponseEntity<List<User>> readAll() {
        return new ResponseEntity<>(userService.readAll(),
        HttpStatus.OK);
    }

    @GetMapping("/{id}")
    public ResponseEntity<User> readById(@PathVariable Long id)
    {
        return new ResponseEntity<>(userService.readById(id),
        HttpStatus.OK);
    }

    @GetMapping("/login/{login}")
    public ResponseEntity<User> readById(@PathVariable String
    login) {
        try {
            return new
        ResponseEntity<>(userService.readByTitle(login), HttpStatus.OK);
        } catch (RuntimeException ex) {
            return new ResponseEntity<>(null, HttpStatus.OK);
        }
    }

    @PutMapping
    public ResponseEntity<User> update(@RequestBody User user) {
        return new ResponseEntity<>(userService.update(user),
        HttpStatus.OK);
    }

    @DeleteMapping("/{id}")
    public HttpStatus delete(@PathVariable Long id) {
        userService.delete(id);
        return HttpStatus.OK;
    }
}

```

Приложение Б.6

Файл Article.java

```

package com.example.hhgserver.entity;
import jakarta.persistence.*;

```

					ПГУ 1.090301.16.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		83

```

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import java.util.List;
@Entity
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
@Table(name = "article")
public class Article {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private Long id;
    private String title;
    private String text;
    @Column(name = "created_at")
    private String createdAt;
    @ManyToOne
    @JoinColumn(name = "created_by")
    private User author;
    @Column(name = "updated_at")
    private String updatedAt;
    @ManyToOne
    @JoinColumn(name = "updated_by")
    private User editor;
    @Column(name = "has_important_block")
    private Boolean hasImportantBlock;
    @Column(name = "has_clarification_block")
    private Boolean hasClarificationBlock;
    @OneToMany(mappedBy = "articleId")
    private List<ImportantBlock> importantBlocks;
    @OneToMany(mappedBy = "articleId")
    private List<ClarificationBlock> clarificationBlocks;
}

```

Приложение Б.7

Файл ArticleDTO.java

```

package com.example.hhgserver.dto;
import lombok.Data;
@Data
public class ArticleDTO {
    private String title;
    private String text;
    private String createdAt;
    private Long createdBy;
    private String updatedAt;
}

```

```
private Long updatedBy;
private Boolean hasImportantBlock;
private Boolean hasClarificationBlock;}
```

Приложение Б.8

Файл ArticleRepository.java

```
package com.example.hhgserver.repository;
import com.example.hhgserver.entity.Article;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import java.util.Optional;
@Repository
public interface ArticleRepository extends
JpaRepository<Article, Long> {
    Optional<Article> findByTitleContaining(String title);}
```

Приложение Б.9

Файл ArticleService.java

```
package com.example.hhgserver.service;
import com.example.hhgserver.dto.ArticleDTO;
import com.example.hhgserver.entity.Article;
import com.example.hhgserver.repository.ArticleRepository;
import lombok.AllArgsConstructor;
import org.springframework.stereotype.Service;
import java.util.List;
@Service
@AllArgsConstructor
public class ArticleService {
    private final ArticleRepository articleRepository;
    private final UserService userService;
    public Article create(ArticleDTO dto) {
        return articleRepository.save(Article.builder()
            .title(dto.getTitle())
            .text(dto.getText())
            .createdAt(dto.getCreatedAt())

            .author(userService.readById(dto.getCreatedBy()))
            .updatedAt(dto.getUpdatedAt())

            .editor(userService.readById(dto.getCreatedBy()))
            .hasClarificationBlock(dto.getHasClarificationBlock())
            .hasImportantBlock(dto.getHasImportantBlock())
            .build());
    }
    public List<Article> readAll(){
        return articleRepository.findAll();}
```

```

    public Article readById(Long id) {
        return articleRepository.findById(id).orElseThrow(() ->
new RuntimeException("Статья не найдена"));
    }

    public Article update(Article article) {
        return articleRepository.save(article);
    }

    public void delete(Long id) {
        articleRepository.deleteById(id);
    }

```

Приложение Б.10

Файл ArticleController.java

```

package com.example.hhgserver.controller;
import com.example.hhgserver.dto.ArticleDTO;
import com.example.hhgserver.entity.Article;
import com.example.hhgserver.service.ArticleService;
import lombok.AllArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.util.List;

@RestController
@RequestMapping("/api/v1/articles")
@AllArgsConstructor
public class ArticleController {
    private final ArticleService articleService;

    @PostMapping
    public ResponseEntity<Article> create(@RequestBody
ArticleDTO dto) {
        return new ResponseEntity<>(articleService.create(dto),
HttpStatus.OK);
    }

    @GetMapping
    public ResponseEntity<List<Article>> readAll() {
        return new ResponseEntity<>(articleService.readAll(),
HttpStatus.OK);
    }

    @GetMapping("/{id}")
    public ResponseEntity<Article> readById(@PathVariable Long
id) {
        return new ResponseEntity<>(articleService.readById(id),
HttpStatus.OK);
    }

    @PutMapping
    public ResponseEntity<Article> update(@RequestBody Article
article) { return new
ResponseEntity<>(articleService.update(article), HttpStatus.OK);
    }

```

```
@DeleteMapping("/{id}")
public HttpStatus delete(@PathVariable Long id) {
    articleService.delete(id);
    return HttpStatus.OK;}}
```

Приложение Б.11

Файл Favourite.java

```
package com.example.hhgserver.entity;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.AllArgsConstructor;
@Table(name = "favourite")
public class Favourite {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private Long id;
    @ManyToOne
    @JoinColumn(name = "user_id")
    private User user;
    @ManyToOne
    @JoinColumn(name = "article_id")
    private Article article;}
```

Приложение Б.12

Файл FavouriteDTO.java

```
package com.example.hhgserver.dto;
import lombok.Data;
@Data
public class FavouriteDTO {
    private Long userId;
    private Long articleId;}
```

Приложение Б.13

Файл FavouriteRepository.java

```
package com.example.hhgserver.repository;
```

```

import com.example.hhgserver.entity.Favourite;
import com.example.hhgserver.entity.User;
import org.springframework.data.jpa.repository.JpaRepository;
import java.util.List;
import java.util.Optional;

public interface FavouriteRepository extends
JpaRepository<Favourite, Long> {
    Optional<List<Favourite>> findByUserId(Long userId);}

```

Приложение Б.14

Файл FavouriteService.java

```

package com.example.hhgserver.service;
import com.example.hhgserver.dto.FavouriteDTO;
import com.example.hhgserver.entity.Favourite;
import com.example.hhgserver.repository.FavouriteRepository;
import lombok.AllArgsConstructor;
import org.springframework.stereotype.Service;
import java.util.List;
@Service
@AllArgsConstructor
public class FavouriteService {
    private final FavouriteRepository favouriteRepository;
    private final UserService userService;
    private final ArticleService articleService;
    public Favourite create(FavouriteDTO dto) {
        return favouriteRepository.save(Favourite.builder()
            .user(userService.readById(dto.getUserId()))
            .article(articleService.readById(dto.getArticleId()))
            .build());}

    public List<Favourite> readByUserId(Long userId) {
        return
favouriteRepository.findByUserId(userId).orElseThrow(() -> new
RuntimeException("Избранное не найдено"));}

    public void delete(Long id) {
        favouriteRepository.deleteById(id);
    }
}

```

Приложение Б.15

Файл FavouriteController.java

```

package com.example.hhgserver.controller;
import com.example.hhgserver.dto.FavouriteDTO;
import com.example.hhgserver.entity.Favourite;

```

					ПГУ 1.090301.16.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		88

```

import com.example.hhgserver.service.FavouriteService;
import lombok.AllArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.util.List;

@RestController
@RequestMapping("/api/v1/favourite")
@AllArgsConstructor
public class FavouriteController {
    private FavouriteService favouriteService;

    @PostMapping
    public ResponseEntity<Favourite> create(@RequestBody
FavouriteDTO dto) {
        return new
ResponseEntity<>(favouriteService.create(dto), HttpStatus.OK);}

    @GetMapping("/user/{userId}")
    public ResponseEntity<List<Favourite>>
readById(@PathVariable Long userId) {
        return new
ResponseEntity<>(favouriteService.readByUserId(userId),
HttpStatus.OK);}

    @DeleteMapping("/{id}")
    public HttpStatus delete(@PathVariable Long id) {
        favouriteService.delete(id);
        return HttpStatus.OK;}}

```

Приложение Б.16

Файл ClarificationBlock.java

```

package com.example.hhgserver.entity;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
@Entity
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class ClarificationBlock {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private Long id;
    private String text;
}

```

					ПГУ 1.090301.16.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		89

```
@Column(name = "article_id")
private Long articleId;}
```

Приложение Б.17

Файл ClarificationBlockDTO.java

```
package com.example.hhgserver.dto;
import lombok.Data;
@Data
public class ClarificationBlockDTO {
    private String text;
    private Long articleId;}
```

Приложение Б.18

Файл ClarificationBlockRepository.java

```
package com.example.hhgserver.repository;
import com.example.hhgserver.entity.ClarificationBlock;
import org.springframework.data.jpa.repository.JpaRepository;
public interface ClarificationBlockRepository extends
JpaRepository<ClarificationBlock, Long> {}
```

Приложение Б.19

Файл ClarificationBlockService.java

```
package com.example.hhgserver.service;
import com.example.hhgserver.dto.ClarificationBlockDTO;
import com.example.hhgserver.entity.ClarificationBlock;
import
com.example.hhgserver.repository.ClarificationBlockRepository;
import lombok.AllArgsConstructor;
import org.springframework.stereotype.Service;
@Service
@AllArgsConstructor
public class ClarificationBlockService {
    private final ClarificationBlockRepository
clarificationBlockRepository;

    public ClarificationBlock create(ClarificationBlockDTO dto)
{
    return
clarificationBlockRepository.save(ClarificationBlock.builder()
        .text(dto.getText())
        .articleId(dto.getArticleId())
        .build());}
```

```

    public ClarificationBlock update(ClarificationBlock block) {
        return clarificationBlockRepository.save(block);
    }

    public void delete(Long id) {
        clarificationBlockRepository.deleteById(id);
    }
}

```

Приложение Б.20

Файл ClarificationBlockController.java

```

package com.example.hhgserver.controller;
import com.example.hhgserver.dto.ClarificationBlockDTO;
import com.example.hhgserver.entity.ClarificationBlock;
import com.example.hhgserver.service.ClarificationBlockService;
import lombok.AllArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api/v1/clarification")
@AllArgsConstructor
public class ClarificationBlockController {
    private final ClarificationBlockService
    clarificationBlockService;

    @PostMapping
    public ResponseEntity<ClarificationBlock>
    create(@RequestBody ClarificationBlockDTO dto) {
        return new
        ResponseEntity<>(clarificationBlockService.create(dto),
        HttpStatus.OK);
    }

    @PutMapping
    public ResponseEntity<ClarificationBlock>
    update(@RequestBody ClarificationBlock block) {
        return new
        ResponseEntity<>(clarificationBlockService.update(block),
        HttpStatus.OK);
    }

    @DeleteMapping("/{id}")
    public HttpStatus delete(@PathVariable Long id) {
        clarificationBlockService.delete(id);
        return HttpStatus.OK;
    }
}

```

Приложение Б.21

Файл ImportantBlock.java

```
package com.example.hhgserver.entity;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.AllArgsConstructor;
public class ImportantBlock {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private Long id;
    private String text;
    @Column(name = "article_id")
    private Long articleId;}

```

Приложение Б.22

Файл ImportantBlockDTO.java

```
package com.example.hhgserver.dto;
import lombok.Data;
@Data
public class ImportantBlockDTO {
    private String text;
    private Long articleId;}

```

Приложение Б.23

Файл ImportantBlockRepository.java

```
package com.example.hhgserver.repository;
import com.example.hhgserver.entity.ImportantBlock;
import org.springframework.data.jpa.repository.JpaRepository;
public interface ImportantBlockRepository extends
JpaRepository<ImportantBlock, Long> {}

```

Приложение Б.24

Файл ImportantBlockService.java

```
package com.example.hhgserver.service;

```

					ПГУ 1.090301.16.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		92

```

import com.example.hhgserver.dto.ImportantBlockDTO;
import com.example.hhgserver.entity.ImportantBlock;
import
com.example.hhgserver.repository.ImportantBlockRepository;
import lombok.AllArgsConstructor;
import org.springframework.stereotype.Service;
@Service
@AllArgsConstructor
public class ImportantBlockService {
    private final ImportantBlockRepository
importantBlockRepository;
    public ImportantBlock create(ImportantBlockDTO dto) {
        return
importantBlockRepository.save(ImportantBlock.builder()
                .text(dto.getText())
                .articleId(dto.getArticleId())
                .build());}

    public ImportantBlock update(ImportantBlock block) {
        return importantBlockRepository.save(block);}

    public void delete(Long id) {
        importantBlockRepository.deleteById(id);}
}

```

Приложение Б.25

Файл ImportantBlockController.java

```

package com.example.hhgserver.controller;
import com.example.hhgserver.dto.ImportantBlockDTO;
import com.example.hhgserver.entity.ImportantBlock;
import com.example.hhgserver.service.ImportantBlockService;
import lombok.AllArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
@RestController
@RequestMapping("/api/v1/important")
@AllArgsConstructor
public class ImportantBlockController {
    private final ImportantBlockService importantBlockService;
    @PostMapping
    public ResponseEntity<ImportantBlock> create(@RequestBody
ImportantBlockDTO dto) {
        return new
ResponseEntity<>(importantBlockService.create(dto),
HttpStatus.OK);}

    @PutMapping

```

```

        public ResponseEntity<ImportantBlock> update(@RequestBody
ImportantBlock block) {
            return new
ResponseEntity<>(importantBlockService.update(block),
HttpStatus.OK);}

        @DeleteMapping("/{id}")
        public HttpStatus delete(@PathVariable Long id) {
            importantBlockService.delete(id);
            return HttpStatus.OK;
        }
    }
}

```

Приложение В

(Обязательное)

Информационно-справочная система статей энциклопедического характера

Листинг мобильного приложения

					ПГУ 1.090301.16.001 ПЗ	Лист
						95
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение В.1

Файл article_screen.xml

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    android:background="@color/hg_beige">
    <RelativeLayout
        android:id="@+id/appbarArticleScreen"
        android:background="@color/hg_teal"
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true">
        <ImageButton
            android:id="@+id/backButton"
            android:layout_width="45dp"
            android:layout_height="45dp"
            android:layout_alignParentStart="true"
            android:layout_marginStart="4dp"
            android:layout_marginTop="4dp"
            android:layout_marginEnd="4dp"
            android:layout_marginBottom="4dp"
            android:background="@color/hg_teal"
            android:padding="4dp"
            android:src="@drawable/back_arrow_icon" />
        <ImageButton
            android:id="@+id/deleteArticleButton"
            android:layout_width="45dp"
            android:layout_height="45dp"
            android:layout_marginStart="4dp"
            android:layout_marginTop="4dp"
            android:layout_marginEnd="4dp"
            android:layout_marginBottom="4dp"
            android:layout_toLeftOf="@+id/sendArticleButton"
            android:background="@drawable/appbar_button"
            android:padding="4dp"
            android:src="@drawable/delete_icon" />
        <ImageButton
            android:id="@+id/sendArticleButton"
            android:layout_width="45dp"
            android:layout_height="45dp"
            android:layout_toLeftOf="@+id/editArticleButton"
            android:background="@drawable/appbar_button"
            android:src="@drawable/send_icon"

```

```

        android:layout_margin="4dp"
        android:padding="4dp" />
<ImageButton
    android:id="@+id/editArticleButton"
    android:layout_width="45dp"
    android:layout_height="45dp"
    android:layout_toLeftOf="@+id/addToFavouritesButton"
    android:background="@drawable/appbar_button"
    android:src="@drawable/edit_icon"
    android:layout_margin="4dp"
    android:padding="4dp" />
<ImageButton
    android:id="@+id/addToFavouritesButton"
    android:layout_width="45dp"
    android:layout_height="45dp"
    android:layout_alignParentEnd="true"
    android:background="@drawable/appbar_button"
    android:src="@drawable/favorites_passive_icon"
    android:layout_margin="4dp"
    android:padding="4dp" />
</RelativeLayout>
<RelativeLayout
    android:id="@+id/textBlock"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:scrollbars="vertical"
    android:scrollbarStyle="insideInset"
    android:layout_below="@id/appbarArticleScreen"
    android:layout_above="@id/editionInfo"
    android:background="@color/hg_beige">
    <TextView
        android:id="@+id/articleTitle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_margin="5dp"
        android:paddingHorizontal="10dp"
        android:fontFamily="@font/rubik_bold"
        android:textColor="@color/hg_blue"
        android:textSize="30sp" />
    <TextView
        android:id="@+id/articleText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/articleTitle"
        android:paddingHorizontal="10dp"
        android:fontFamily="@font/rubik"
        android:textColor="@color/hg_blue"
        android:textSize="20sp" />
</androidx.recyclerview.widget.RecyclerView

```

```

        android:id="@+id/importantList"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/articleText">
</androidx.recyclerview.widget.RecyclerView>
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/clarificationList"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/importantList">
</androidx.recyclerview.widget.RecyclerView>
</RelativeLayout>
<TextView
    android:id="@+id/editionInfo"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginHorizontal="5dp"
    android:layout_marginTop="10dp"
    android:layout_above="@id/creationInfo"
    android:fontFamily="@font/rubik"
    android:textColor="@color/hg_blue"
    android:textSize="18sp" />
<TextView
    android:id="@+id/creationInfo"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginHorizontal="5dp"
    android:layout_alignParentBottom="true"
    android:fontFamily="@font/rubik"
    android:textColor="@color/hg_blue"
    android:textSize="18sp" />
</RelativeLayout>

```

Приложение В.2

Файл authorization_screen.xml

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    android:background="@color/hg_beige">
<LinearLayout
    android:id="@+id/appbarAuthorizationScreen"
    android:background="@color/hg_teal"
    android:orientation="horizontal"
    android:layout_width="match_parent"

```

```

        android:layout_height="wrap_content">
        <ImageButton
            android:id="@+id/backToMainFromAuthorizationButton"
            android:layout_width="55dp"
            android:layout_height="55dp"
            android:layout_alignParentStart="true"
            android:background="@color/hg_teal"
            android:src="@drawable/back_arrow_icon"
            android:layout_margin="5dp"
            android:padding="5dp" />
    </LinearLayout>
    <LinearLayout
        android:id="@+id/authorizationBlock"
        android:background="@drawable/red_background"
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="10dp"
        android:layout_marginTop="70dp"
        android:layout_marginHorizontal="30dp"
        android:gravity="center">
        <EditText
            android:id="@+id/loginField"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:inputType="text"
            android:background="@drawable/beige_field"
            android:padding="10dp"
            android:layout_margin="5dp"
            android:hint="Логин"
            android:textColorHint="@color/hg_blue"
            android:fontFamily="@font/rubik"
            android:textSize="20sp"
            android:textColor="@color/hg_blue"
            android:maxLength="32" />
        <EditText
            android:id="@+id/passwordField"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:background="@drawable/beige_field"
            android:padding="10dp"
            android:layout_margin="5dp"
            android:inputType="textPassword"
            android:hint="Пароль"
            android:textColorHint="@color/hg_blue"
            android:fontFamily="@font/rubik"
            android:textSize="20sp"
            android:textColor="@color/hg_blue"
            android:maxLength="32" />
    <Button

```

```

        android:id="@+id/authorizeButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/action_button"
        android:paddingHorizontal="10dp"
        android:layout_margin="5dp"
        android:text="Войти"
        android:fontFamily="@font/rubik_bold"
        android:textSize="20sp"
        android:textColor="@color/hg_beige" />
    <Button
        android:id="@+id/toRegistrationButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/action_button"
        android:paddingHorizontal="10dp"
        android:layout_marginTop="10dp"
        android:layout_marginBottom="5dp"
        android:text="Регистрация"
        android:fontFamily="@font/rubik_bold"
        android:textSize="17sp"
        android:textColor="@color/hg_beige" />
</LinearLayout>
</LinearLayout>

```

Приложение В.3

Файл favourites_screen.xml

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    android:background="@color/hg_beige">
    <LinearLayout
        android:id="@+id/appbarRegistrationScreen"
        android:background="@color/hg_teal"
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <ImageButton
            android:id="@+id/backToMainFromFavourites"
            android:layout_width="55dp"
            android:layout_height="55dp"
            android:background="@color/hg_teal"
            android:src="@drawable/back_arrow_icon"
            android:layout_margin="5dp"

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.16.001 ПЗ

Лист

100

```

        android:padding="5dp" />
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="9dp"
    android:paddingTop="10dp"
    android:text="Избранное"
    android:fontFamily="@font/rubik_bold"
    android:textColor="@color/hg_beige"
    android:textSize="25sp" />
</LinearLayout>
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/favouritesList"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
<TextView
    android:id="@+id/emptyListText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginHorizontal="60dp"
    android:layout_marginTop="150dp"
    android:fontFamily="@font/rubik"
    android:gravity="center"
    android:text="Нет избранных статей"
    android:textColor="@color/hg_salmon"
    android:textSize="25sp" />
</LinearLayout>

```

Приложение В.4

Файл important_block.xml

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/important_block_border"
    android:layout_margin="10dp">
    <TextView
        android:id="@+id/importantBlockText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="10dp"

        android:textSize="20sp"
        android:textColor="@color/hg_salmon"
        android:fontFamily="@font/rubik_medium" />
</LinearLayout>

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.16.001 ПЗ

Лист

101

Приложение В.5

Файл main_screen.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    android:background="@color/hg_beige">
    <RelativeLayout
        android:id="@+id/appbarMainScreen"
        android:background="@color/hg_teal"
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <ImageButton
            android:id="@+id/backButton"
            android:layout_width="25dp"
            android:layout_height="45dp"
            android:layout_alignParentStart="true"

            android:layout_marginStart="4dp"
            android:layout_marginTop="4dp"
            android:layout_marginEnd="4dp"
            android:layout_marginBottom="4dp"
            android:background="@color/hg_teal"
            android:padding="4dp"
            android:src="@drawable/back_arrow_icon" />
        <ImageButton
            android:id="@+id/registerButton"
            android:layout_width="45dp"
            android:layout_height="45dp"
            android:layout_toRightOf="@id/backButton"
            android:background="@drawable/appbar_button"
            android:src="@drawable/register_icon"
            android:layout_margin="4dp"
            android:padding="4dp" />
        <ImageButton
            android:id="@+id/updateDBButton"
            android:layout_width="45dp"
            android:layout_height="45dp"
            android:layout_toRightOf="@id/registerButton"
            android:background="@drawable/appbar_button"
            android:src="@drawable/update_db_icon"
            android:layout_margin="4dp"
            android:padding="4dp" />
        <ImageButton
```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.16.001 ПЗ

Лист

102

```

        android:id="@+id/favoritesListButton"
        android:layout_width="45dp"
        android:layout_height="45dp"
        android:layout_toRightOf="@id/updateDBButton"
        android:background="@drawable/appbar_button"
        android:src="@drawable/favorites_active_icon"
        android:layout_margin="4dp"
        android:padding="4dp" />
    <ImageButton
        android:id="@+id/addArticleButton"
        android:layout_width="45dp"
        android:layout_height="45dp"
        android:layout_toRightOf="@id/favoritesListButton"
        android:background="@drawable/appbar_button"
        android:src="@drawable/plus_beige_icon"
        android:layout_margin="4dp"
        android:padding="4dp" />

    <EditText
        android:id="@+id/searchField"
        android:layout_width="140dp"
        android:layout_height="wrap_content"
        android:layout_marginVertical="8dp"
        android:layout_marginLeft="5dp"
        android:layout_toLeftOf="@id/searchButton"
        android:background="@drawable/search_field_beige"
        android:ellipsize="start"
        android:fontFamily="@font/rubik"
        android:inputType="textCapSentences"
        android:maxLines="1"
        android:paddingVertical="8dp"
        android:paddingLeft="10dp"
        android:singleLine="true"
        android:hint="Поиск"
        android:textColorHint="@color/hg_blue"
        android:textAlignment="viewStart"
        android:textColor="@color/hg_blue"
        android:textSize="22sp" />
    <ImageButton
        android:id="@+id/searchButton"
        android:layout_width="40dp"
        android:layout_height="43dp"
        android:background="@drawable/search_button_beige"
        android:src="@drawable/search_icon"
        android:layout_marginRight="10dp"
        android:layout_marginVertical="8dp"
        android:layout_alignParentEnd="true"
        android:padding="3dp" />
</RelativeLayout>

```

```

<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/lowAppBarMainScreen"
    android:background="@drawable/low_appbar"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <android.widget.Button
        android:id="@+id/articleFilter"
        android:layout_width="wrap_content"
        android:layout_height="30dp"
        android:background="@drawable/filter_button_active"
        android:layout_marginLeft="100dp"
        android:layout_marginVertical="5dp"
        android:paddingHorizontal="10dp"
        android:layout_weight="2"
        android:text="статьи"
        android:fontFamily="@font/rubik"
        android:textSize="20dp"
        android:textAlignment="center"
        android:textAllCaps="false"
        android:textColor="@color/hg_blue" />
    <android.widget.Button
        android:id="@+id/usersFilter"
        android:fontFamily="@font/rubik"
        android:layout_width="wrap_content"
        android:layout_height="30dp"
        android:background="@drawable/filter_button"
        android:layout_margin="5dp"
        android:paddingHorizontal="10dp"
        android:layout_weight="3"
        android:text="пользователи"
        android:textSize="20sp"
        android:textAlignment="center"
        android:textAllCaps="false"
        android:textColor="@color/hg_blue" />
</LinearLayout>
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/searchResults"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
<TextView
    android:id="@+id/dontPanicText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginHorizontal="60dp"
    android:layout_marginTop="150dp"
    android:fontFamily="@font/rubik_black"
    android:gravity="center"
    android:text="DON'T\nPANIC"

```

```

        android:textColor="@color/hg_salmon"
        android:textSize="80sp" />
</LinearLayout>

```

Приложение В.6

Файл new_article_screen.xml

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    android:background="@color/hg_beige">
    <RelativeLayout
        android:id="@+id/appbarNewArticleScreen"
        android:background="@color/hg_teal"
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <ImageButton
            android:id="@+id/backButton"
            android:layout_width="45dp"
            android:layout_height="45dp"
            android:layout_alignParentStart="true"
            android:background="@color/hg_teal"
            android:src="@drawable/back_arrow_icon"
            android:layout_margin="4dp"
            android:padding="4dp" />
        <TextView
            android:id="@+id/newArticleTitle"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="15dp"
            android:layout_toRightOf="@id/backButton"
            android:layout_toLeftOf="@+id/saveArticleButton"
            android:text="Новая статья"
            android:fontFamily="@font/rubik_bold"
            android:textColor="@color/hg_beige"
            android:textSize="25sp" />
        <ImageButton
            android:id="@+id/saveArticleButton"
            android:layout_width="45dp"
            android:layout_height="45dp"
            android:layout_alignParentEnd="true"
            android:background="@drawable/appbar_button"
            android:src="@drawable/save_icon"
            android:layout_marginRight="10dp"

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.16.001 ПЗ

Лист

105

```

        android:layout_marginVertical="7dp"
        android:padding="4dp" />
    </RelativeLayout>
    <RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/lowAppBarNewArticleScreen"
        android:background="@drawable/low_appbar"
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
        <ImageButton
            android:id="@+id/addImportantButton"
            android:layout_width="45dp"
            android:layout_height="45dp"
            android:layout_toLeftOf="@+id/addClarificationButton"
            android:background="@drawable/add_important_button"
            android:src="@drawable/plus_salmon_icon"
            android:layout_margin="4dp"
            android:padding="10dp" />
        <ImageButton
            android:id="@+id/addClarificationButton"
            android:layout_width="45dp"
            android:layout_height="45dp"
            android:layout_alignParentEnd="true"
            android:background="@drawable/add_clarification_button"
            android:src="@drawable/plus_mustard_icon"
            android:layout_margin="4dp"
            android:padding="10dp" />
    </RelativeLayout>
    <EditText
        android:id="@+id/articleTitle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:inputType="textCapSentences"
        android:hint="Заголовок"
        android:textColorHint="@color/hg_blue"
        android:fontFamily="@font/rubik_bold"
        android:textColor="@color/hg_blue"
        android:textSize="35sp" />
    <EditText
        android:id="@+id/articleText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginHorizontal="5dp"
        android:layout_marginBottom="5dp"
        android:scrollbars="vertical"
        android:scrollbarStyle="insideInset"
        android:gravity="start"
        android:inputType="textCapSentences|textMultiLine"

```

```

        android:hint="Текст статьи"
        android:fontFamily="@font/rubik"
        android:textColorHint="@color/hg_blue"
        android:textColor="@color/hg_blue"
        android:textSize="20sp" />
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/editImportantList"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/editClarificationList"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
</LinearLayout>

```

Приложение В.7

Файл registration_screen.xml

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    android:background="@color/hg_beige">
    <LinearLayout
        android:id="@+id/appbarRegistrationScreen"
        android:background="@color/hg_teal"
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <ImageButton
            android:id="@+id/backToMainFromRegistrationButton"
            android:layout_width="55dp"
            android:layout_height="55dp"
            android:layout_gravity="start"
            android:background="@color/hg_teal"
            android:src="@drawable/back_arrow_icon"
            android:layout_margin="5dp"
            android:padding="5dp" />
    </LinearLayout>

    <LinearLayout
        android:id="@+id/registrationBlock"
        android:background="@drawable/red_background"
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.16.001 ПЗ

Лист

107

```

android:padding="10dp"
android:layout_marginTop="70dp"
android:layout_marginHorizontal="30dp"
android:gravity="center">
<EditText
    android:id="@+id/newLoginField"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/beige_field"
    android:padding="10dp"
    android:layout_margin="5dp"
    android:inputType="text"
    android:hint="Логин"
    android:textColorHint="@color/hg_blue"
    android:fontFamily="@font/rubik"
    android:textSize="20sp"
    android:textColor="@color/hg_blue"
    android:maxLength="32" />
<EditText
    android:id="@+id/newPasswordField"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/beige_field"
    android:padding="10dp"
    android:layout_margin="5dp"
    android:inputType="textPassword"
    android:hint="Пароль"
    android:textColorHint="@color/hg_blue"
    android:fontFamily="@font/rubik"
    android:textSize="20sp"
    android:textColor="@color/hg_blue"
    android:maxLength="32" />
<EditText
    android:id="@+id/repeatPasswordField"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/beige_field"
    android:padding="10dp"
    android:layout_margin="5dp"
    android:inputType="textPassword"
    android:hint="Повтор пароля"
    android:textColorHint="@color/hg_blue"
    android:fontFamily="@font/rubik"
    android:textSize="20sp"
    android:textColor="@color/hg_blue"
    android:maxLength="32" />
<Button
    android:id="@+id/registrationButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

```

```

        android:background="@drawable/action_button"
        android:paddingHorizontal="10dp"
        android:layout_margin="5dp"
        android:text="Регистрация"
        android:fontFamily="@font/rubik_bold"
        android:textSize="20sp"
        android:textColor="@color/hg_beige" />
    <Button
        android:id="@+id/toAuthorizationButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/action_button"
        android:paddingHorizontal="10dp"
        android:layout_marginTop="15dp"
        android:layout_marginBottom="5dp"
        android:text="Войти"
        android:fontFamily="@font/rubik_bold"
        android:textSize="17sp"
        android:textColor="@color/hg_beige" />
</LinearLayout>
</LinearLayout>

```

Приложение В.8

Файл clarification_block.xml

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/clarification_block_border"
    android:layout_margin="10dp">
    <TextView
        android:id="@+id/clarificationBlockText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="10dp"
        android:textSize="18sp"
        android:textColor="@color/hg_blue"
        android:fontFamily="@font/rubik" />
</LinearLayout>

```

Приложение В.9

Файл custom_dialog.xml

```

<?xml version="1.0" encoding="utf-8"?>

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.16.001 ПЗ

Лист

109

```

<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@color/hg_beige_dark">
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="10dp"
        android:background="@drawable/red_background"
        android:orientation="vertical">
        <TextView
            android:id="@+id/dialogHeader"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginVertical="10dp"
            android:fontFamily="@font/rubik_bold"
            android:textAllCaps="false"
            android:gravity="center"
            android:textColor="@color/hg_blue"
            android:textSize="20sp" />
        <TextView
            android:id="@+id/alertText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginVertical="5dp"
            android:fontFamily="@font/rubik"
            android:textAllCaps="false"
            android:gravity="start"
            android:textColor="@color/hg_blue"
            android:textSize="20sp" />
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="horizontal">
            <Button
                android:id="@+id/negativeButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:background="@drawable/action_button"
                android:layout_margin="5dp"
                android:layout_weight="1"
                android:textColor="@color/hg_beige"
                android:fontFamily="@font/rubik_bold"
                android:textSize="15sp"
                android:maxLength="8"
                android:textAllCaps="true"
                android:visibility="invisible" />
            <Button
                android:id="@+id/neutralButton"

```

```

        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/action_button"
        android:layout_margin="5dp"
        android:layout_weight="1"
        android:textColor="@color/hg_beige"
        android:fontFamily="@font/rubik_bold"
        android:textSize="15sp"
        android:textAllCaps="true"
        android:maxLength="8"
        android:visibility="invisible" />
    <Button
        android:id="@+id/positiveButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/action_button"
        android:layout_margin="5dp"
        android:layout_weight="1"
        android:textColor="@color/hg_beige"
        android:fontFamily="@font/rubik_bold"
        android:textSize="15sp"
        android:textAllCaps="true"
        android:maxLength="8"
        android:visibility="invisible" />
</LinearLayout>
</LinearLayout>
</LinearLayout>

```

Приложение В.10

Файл edit_special_block.xml

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@color/hg_beige_dark">
    <LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/red_background">

        <EditText
            android:id="@+id/definitelyUniqueId"
            android:layout_width="match_parent"
            android:layout_height="150dp"

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.16.001 ПЗ

Лист

111

```

        android:background="@drawable/beige_field"
        android:padding="10dp"
        android:layout_margin="10dp"
        android:gravity="start"
        android:hint="Особый блок"
        android:inputType="textCapSentences|textMultiLine"
        android:textColorHint="@color/hg_blue"
        android:fontFamily="@font/rubik"
        android:textColor="@color/hg_blue"
        android:textSize="18sp" />
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp">
    <Button
        android:id="@+id/negativeButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/action_button"
        android:layout_margin="5dp"
        android:layout_weight="1"
        android:text="Отмена"
        android:textColor="@color/hg_beige"
        android:fontFamily="@font/rubik_bold"
        android:textSize="15sp"
        android:maxLength="8" />
    <Button
        android:id="@+id/neutralButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/action_button"
        android:layout_margin="5dp"
        android:layout_weight="1"
        android:text="Удалить"
        android:textColor="@color/hg_beige"
        android:fontFamily="@font/rubik_bold"
        android:textSize="15sp"
        android:textAllCaps="true" />
    <Button
        android:id="@+id/positiveButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/action_button"
        android:layout_margin="5dp"
        android:layout_weight="1"

        android:text="Сохранить"
        android:textColor="@color/hg_beige"
        android:fontFamily="@font/rubik_bold"
        android:textSize="15sp"

```

```

        android:textAllCaps="true" />
    </LinearLayout>
</LinearLayout>
</LinearLayout>

```

Приложение В.11

Файл search_item_article.xml

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <TextView
        android:id="@+id/resultHeader"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingHorizontal="10dp"
        android:text="тестовый текст"
        android:fontFamily="@font/rubik"
        android:textColor="@color/hg_blue"
        android:textSize="18sp"
        android:gravity="start" />
    <TextView
        android:id="@+id/resultType"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingHorizontal="10dp"
        android:text="статья"
        android:fontFamily="@font/rubik_medium"
        android:textColor="@color/hg_salmon"
        android:textSize="18sp"
        android:gravity="start" />
</LinearLayout>

```

Приложение В.12

Файл Article.kt

```

package com.example.hgg
class Article {
    var id: Int = -1
    var name: String = ""
    var text: String = ""
    var importantBlock: ArrayList<SpecialBlock> =
ArrayList<SpecialBlock>()
    var needClarificationBlock: ArrayList<SpecialBlock> =
ArrayList<SpecialBlock>()

```

```

var createdAt: String = ""
var createdBy: String = ""
var updatedAt: String = ""
var updatedBy: String = ""
var hasImportantBlock: Boolean = false
var hasClarificationBlock: Boolean = false

constructor(_id: Int, _name: String, _text: String,
            _createdAt: String, _createdBy: String,
            _updatedAt: String, _updatedBy: String,
            _hasImportantBlock: Boolean,
            _hasClarificationBlock: Boolean) {
    this.id = _id
    this.name = _name
    this.text = _text
    this.createdAt = _createdAt
    this.createdBy = _createdBy
    this.updatedAt = _updatedAt
    this.updatedBy = _updatedBy
    this.hasImportantBlock = _hasImportantBlock
    this.hasClarificationBlock = _hasClarificationBlock}

fun setClarifications (_block: SpecialBlock) {
    needClarificationBlock.add(_block)
    this.hasClarificationBlock = true}

fun setImportantBlock (_block: SpecialBlock) {
    importantBlock.add(_block)
    this.hasImportantBlock = true}

fun getImportantBlocksFull(): ArrayList<SpecialBlock> {
    return this.importantBlock}

fun getClarificationsFull(): ArrayList<SpecialBlock> {
    return this.needClarificationBlock}}

```

Приложение В.13

Файл User.kt

```

package com.example.hgg
import kotlinx.serialization.Serializable
import kotlinx.serialization.json.Json
import kotlinx.serialization.encodeToString

@Serializable
class User {
    var id: Int = -1
    var login: String = ""
    var role: String = ""
}

```

```

constructor() {}
constructor(_id: Int, _login: String, _role: String) {
    this.id = _id
    this.login = _login
    this.role = _role}}

```

Приложение В.14

Файл ArticleScreen.kt

```

package com.example.hgg
import android.app.Activity
import android.app.Dialog
import android.content.Context
import android.content.Intent
import android.os.Bundle
import android.text.method.ScrollingMovementMethod

import android.widget.Button
import android.widget.ImageButton
import android.widget.LinearLayout
import android.widget.RelativeLayout
import android.widget.TextView
import androidx.core.view.isVisible
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.example.lab7.DBHandler
import kotlinx.serialization.json.Json
import kotlinx.serialization.decodeFromString

class ArticleScreen: Activity() {
    val context: Context = this
    private val dbh: DBHandler = DBHandler(context)
    private var user: User = User()
    private lateinit var article: Article
    private lateinit var rvImportant: RecyclerView
    private val importantAdapter = ImportantAdapter()
    private lateinit var rvClarification: RecyclerView
    private val clarificationAdapter = ClarificationAdapter()
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.article_screen)

        val backButton: ImageButton =
            findViewById(R.id.backButton)
        val deleteButton: ImageButton =
            findViewById(R.id.deleteArticleButton)
        val sendButton: ImageButton =
            findViewById(R.id.sendArticleButton)

```

```

        val editButton: ImageButton =
findViewById(R.id.editArticleButton)
        val favouriteButton: ImageButton =
findViewById(R.id.addToFavouritesButton)
        val title: TextView = findViewById(R.id.articleTitle)
        val text: TextView = findViewById(R.id.articleText)
        val editionInfo: TextView =
findViewById(R.id.editionInfo)
        val creationInfo: TextView =
findViewById(R.id.creationInfo)
        var favourites: ArrayList<Int> = ArrayList<Int>()
        var isFavourite: Boolean = false
        text.movementMethod = ScrollingMovementMethod()
        val arguments: Bundle? = intent.extras
        if (arguments != null) {
            val userId = arguments.getInt("userId")
            if (userId > 0) {
                user = dbh.getUser(userId)!!
                favourites = dbh.getFavourites(user.id)
            }
            article =
dbh.getArticle(arguments.getInt("articleId"))!!
        }
        deleteButton.isVisible = false
        sendButton.isVisible = false
        editButton.isVisible = false
        when (user.role) {
            "editor" -> {
                editButton.isVisible = true
            }
            "moderator" -> {
                deleteButton.isVisible = true
                editButton.isVisible = true
            }
        }
        title.text = article.name
        text.text = article.text
        editionInfo.text = article.updatedAt + " " +
article.updatedBy
        creationInfo.text = article.createdAt + " " +
article.createdBy
        if (favourites.contains(article.id)) {

favouriteButton.setImageResource(R.drawable.favorites_active_ico
n)

            isFavourite = true}

        if (article.hasImportantBlock) {
importantAdapter.setData(article.getImportantBlocksFull())
        }

```

```

        if (article.hasClarificationBlock) {
            clarificationAdapter.setData(article.getClarificationsFull())
        }
        rvImportant = findViewById(R.id.importantList)
        rvImportant.layoutManager = LinearLayoutManager(this)
        rvImportant.adapter = importantAdapter
        rvClarification = findViewById(R.id.clarificationList)
        rvClarification.layoutManager =
            LinearLayoutManager(this)
        rvClarification.adapter = clarificationAdapter
        editButton.setOnClickListener() {
            val toEditArticleIntent: Intent = Intent(this,
                CreateArticle::class.java)
            toEditArticleIntent.putExtra("isEdition", true)
            toEditArticleIntent.putExtra("userId", user.id)
            toEditArticleIntent.putExtra("articleId",
                article.id)
            startActivity(toEditArticleIntent)
            finish()
        }
        deleteButton.setOnClickListener() {
            createDeleteDialog("Удаление статьи", "Отменить это
                действие будет невозможно.\nУдалить статью?", "Да", "Нет")
        }
        backButton.setOnClickListener() {
            finish()
        }
        favouriteButton.setOnClickListener() {
            if (user.id == -1) {
                createOneButtonDialog("Действие недоступно",
                    "Необходимо авторизоваться, чтобы сохранять избранные статьи",
                    "OK")
            } else if (isFavourite) {
                favouriteButton.setImageResource(R.drawable.favorites_passive_ico
                    on)
                dbh.deleteFavourite(user.id, article.id)
            } else {
                favouriteButton.setImageResource(R.drawable.favorites_active_ico
                    n)
                dbh.addFavourite(user.id, article.id)}}}

        private fun createDeleteDialog (_header: String,
            _alertText:String, _buttonText: String, _negativeButtonText:
                String) {
            val dialog: Dialog = Dialog(context)
            dialog setContentView(R.layout.custom_dialog)
            val header: TextView =
                dialog.findViewById(R.id.dialogHeader)
            val text: TextView = dialog.findViewById(R.id.alertText)
            val positiveButton: Button =
                dialog.findViewById(R.id.positiveButton)

```

```

        val negativeButton: Button =
dialog.findViewById(R.id.negativeButton)
        header.setText(_header)
        text.setText(_alertText)
        positiveButton.setText(_buttonText)
        positiveButton.isVisible = true
        dialog.show()
        negativeButton.setOnClickListener() {
            dialog.dismiss()
        }
        positiveButton.setOnClickListener() {
            dbh.deleteArticle(article.id)
            dialog.dismiss()
            finish()
        }

private fun createOneButtonDialog (_header: String,
_alertText:String, _buttonText: String) {
    val dialog: Dialog = Dialog(context)
    dialog setContentView(R.layout.custom_dialog)
    val header: TextView =
dialog.findViewById(R.id.dialogHeader)
    val text: TextView = dialog.findViewById(R.id.alertText)
    val okButton: Button =
dialog.findViewById(R.id.positiveButton)

    header.setText(_header)
    text.setText(_alertText)
    okButton.setText(_buttonText)
    okButton.isVisible = true
    dialog.show()
    okButton.setOnClickListener() {
        dialog.dismiss()
    }}

```

Приложение В.15

Файл Authorization.kt

```

package com.example.hgg
import android.app.Activity
import android.app.Dialog
import android.content.Context
import android.content.Intent
import android.content.SharedPreferences
import android.os.Bundle

import android.widget.Button
import android.widget.EditText
import android.widget.ImageButton
import android.widget.TextView
import androidx.core.view.isVisible
import at.favre.lib.crypto.bcrypt.BCrypt

```

```

import com.example.lab7.DBHandler
import kotlinx.serialization.json.Json
import okhttp3.OkHttpClient
import okhttp3.Request
import org.json.JSONObject
import java.io.IOException

class Authorization : Activity () {
    val context: Context = this
    private val dbh: DBHandler = DBHandler(context)
    private var user: User = User()
    val httpClient = OkHttpClient()
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.authorization_screen)
        val backButton: ImageButton =
            findViewById(R.id.backToMainFromAuthorizationButton)
        val loginField: EditText = findViewById(R.id.loginField)
        val passwordField: EditText =
            findViewById(R.id.passwordField)
        val authorizeButton: Button =
            findViewById(R.id.authorizeButton)
        val toRegistrationButton: Button =
            findViewById(R.id.toRegistrationButton)
        backButton.setOnClickListener() {
            val toMainScreenIntent: Intent = Intent(this,
                MainActivity::class.java)
            startActivity(toMainScreenIntent)
            finish()
        }
        authorizeButton.setOnClickListener() {
            val login = loginField.text.toString()
            val password = passwordField.text.toString()
            if (login.isNullOrEmpty() ||
                password.isNullOrEmpty()) {
                createOneButtonDialog(
                    "Чего-то не хватает",
                    "Необходимо указать логин и пароль",
                    "OK"
                )
            } else {
                val dbHash: String? = dbh.getUserHash(login)
                if (!dbHash.isNullOrEmpty()) {
                    val result: BCrypt.Result =
                        BCrypt.verifier().verify(password.toCharArray(), dbHash)
                    if (result.verified) {
                        user = dbh.getUser(login)!!
                        val sharedPreferences =

```

```

        getSharedPreferences("settings",
MODE_PRIVATE)

        val sharedPrefEditor:
SharedPreferences.Editor = sharedPref.edit()
        sharedPrefEditor.putInt("userId",
user.id)

        sharedPrefEditor.apply()
        val toMainScreenIntent: Intent =
Intent(this, MainActivity::class.java)
        startActivity(toMainScreenIntent)
        finish()
    } else {
        createOneButtonDialog(
            " Неверный пароль",
            "Введён неверный пароль", "OK")
        loginField.text.clear()
        passwordField.text.clear()
    } else {
        createOneButtonDialog(
            " Пользователь не найден",
            "Пользователь с таким логином не
найден", "OK") }}}
        toRegistrationButton.setOnClickListener() {
            val toRegistrationIntent: Intent = Intent(this,
Registration::class.java)
            startActivity(toRegistrationIntent)
            finish()}}

    private fun createOneButtonDialog (_header: String,
_alertText:String, _buttonText: String) {
        val dialog: Dialog = Dialog(context)
        dialog setContentView(R.layout.custom_dialog)
        val header: TextView =
dialog.findViewById(R.id.dialogHeader)
        val text: TextView = dialog.findViewById(R.id.alertText)
        val okButton: Button =
dialog.findViewById(R.id.positiveButton)
        header.setText(_header)
        text.setText(_alertText)
        okButton.setText(_buttonText)
        okButton.isVisible = true
        dialog.show()
        okButton.setOnClickListener() {
            dialog.dismiss()}}}

```

Приложение В.16

Файл CreateArticle.kt

```
package com.example.hgg
```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.16.001 ПЗ

Лист

120

```

import android.app.Activity
import android.app.Dialog
import android.content.Context
import android.content.Intent
import android.os.Bundle

import android.widget.Button
import android.widget.EditText
import android.widget.ImageButton
import android.widget.RelativeLayout
import android.widget.TextView
import androidx.core.view.isVisible
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.example.lab7.DBHandler
import java.time.OffsetDateTime
import java.time.ZoneOffset
import java.time.format.DateTimeFormatter
import java.util.Locale

class CreateArticle: Activity() {
    private var user: User = User()
    private var context: Context = this
    private var dbh: DBHandler = DBHandler(this)
    private lateinit var article: Article
    private lateinit var rvImportant: RecyclerView
    private val importantAdapter = ImportantAdapter()
    private lateinit var rvClarification: RecyclerView
    private val clarificationAdapter = ClarificationAdapter()
    private var importantBlocks: ArrayList<SpecialBlock> =
        ArrayList<SpecialBlock>()
    private var clarifications: ArrayList<SpecialBlock> =
        ArrayList<SpecialBlock>()

    private var isEdited: Boolean = false
    private var importantEdited = mutableMapOf<Int, String>()
    private var clarificationEdited = mutableMapOf<Int,
        String>() )

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.new_article_screen)
        val backButton: ImageButton =
            findViewById(R.id.backButton)
        val appBarTitle: TextView =
            findViewById(R.id.newArticleTitle)
        val saveButton: ImageButton =
            findViewById(R.id.saveArticleButton)
        val addImportantBlockButton: ImageButton =
            findViewById(R.id.addImportantButton)
    }

```

```

        val addClarificationBlockButton: ImageButton =
findViewById(R.id.addClarificationButton)
        val editTitle: EditText =
findViewById(R.id.articleTitle)
        val editText: EditText = findViewById(R.id.articleText)
        rvImportant = findViewById(R.id.editImportantList)
        rvImportant.layoutManager = LinearLayoutManager(this)
        rvImportant.adapter = importantAdapter
        rvClarification =
findViewById(R.id.editClarificationList)
        rvClarification.layoutManager =
LinearLayoutManager(this)
        rvClarification.adapter = clarificationAdapter
        val arguments: Bundle? = intent.extras
        var isEdition: Boolean = false
        if (arguments != null) {
            user = dbh.getUser(arguments.getInt("userId"))!!
            isEdition = arguments.getBoolean("isEdition")
            if (isEdition) {
                article =
dbh.getArticle(arguments.getInt("articleId"))!!
                importantBlocks =
article.getImportantBlocksFull()
                clarifications = article.getClarificationsFull()
                importantAdapter.setData(importantBlocks)
                clarificationAdapter.setData(clarifications)
                for (ib in importantBlocks)
                    importantEdited.put(ib.getId(), "old")
                for (cb in clarifications)
                    clarificationEdited.put(cb.getId(), "old")

                editText.setText(article.text)
                editTitle.setText(article.name)}}
        if (isEdition)
            appBarTitle.setText("Редактирование")
        editText.setOnClickListener() {
            isEdited = true}
        editTitle.setOnClickListener() {
            isEdited = true}
        addImportantBlockButton.setOnClickListener() {
            Log.i("CreateArticle",
"addImportantBlockButton.onClick()")
            createImportantBlockEditionForm(false, null)}

        addClarificationBlockButton.setOnClickListener() {
            Log.i("CreateArticle",
"addClarificationBlockButton.onClick()")
            createClarificationBlockEditionForm(false, null)}

```

```

        importantAdapter.setListener (object:
SpecialBlockClickListener {
            override fun onBlockClick(_block: SpecialBlock) {
                Log.i("CreateArticle", "редактирование блока
\"Важное\"")
                createImportantBlockEditionForm(true, _block)
            }
        })
        clarificationAdapter.setListener(object:
SpecialBlockClickListener{
            override fun onBlockClick(_block: SpecialBlock) {
                Log.i("CreateArticle",
"addClarificationBlockButton.onClick()")
                createClarificationBlockEditionForm(true,
_block) } })
        saveButton.setOnClickListener() {
            Log.i("CreateArticle", "saveButton.onClick()")
            if (editText.text.isNullOrEmpty() ||
editTitle.text.isNullOrEmpty()) {
                createOneButtonDialog("Чего-то не хватает", "Для
сохранения необходимо задать название статьи и ввести её текст",
"OK")
            } else {
                val offset = ZoneOffset.UTC
                val odt = OffsetDateTime.now(offset)
                val resTime =
odt.format(DateTimeFormatter.ISO_OFFSET_DATE_TIME)

                if (!isEdition) {
                    dbh.addArticle(editTitle.text.toString(),
editText.text.toString(), resTime, user.id)
                    article =
dbh.getArticles(editTitle.text.toString())[0]
                    for (i in importantBlocks)
                        dbh.addImportantBlock(article.id,
i.getText(), resTime, user.id)
                    for (c in clarifications)
                        dbh.addClarificationBlock(article.id,
c.getText(), resTime, user.id)
                } else {
                    dbh.updateArticle(article.id,
editTitle.text.toString(), editText.text.toString(), resTime,
user.id)

                    for (e in importantEdited.entries)
                        if (e.value == "removed")
                            dbh.deleteImportantBlock(e.key)
                    for (e in clarificationEdited.entries)
                        if (e.value == "removed")
                            dbh.deleteClarificationBlock(e.key)

                    for (i in importantBlocks)

```

```

        if (importantEdited.get(i.getId()) ==
"new")
            if (i.getId() <= 0)

dbh.addImportantBlock(article.id, i.getText(), resTime, user.id)
            else

dbh.updateImportantBlock(i.getId(), i.getText())

        for (c in clarifications)
            if (clarificationEdited.get(c.getId())
== "new")
                if (c.getId() <= 0)

dbh.addClarificationBlock(article.id, c.getText(), resTime,
user.id)
                else
dbh.updateClarificationBlock(c.getId(), c.getText())

        article = dbh.getArticle(article.id)!!
    }
    finish() }}

    backButton.setOnClickListener() {
        Log.i("CreateArticle", "backButton.onClick()")
        if (isEdited)
            createBackDialog("Выход без сохранения", "Все
изменения будут потеряны.\nЗаккрыть страницу?", "Да", "Нет")
        else {
            val toArticleIntent: Intent = Intent(this,
CreateArticle::class.java)
            toArticleIntent.putExtra("userId", user.id)
            toArticleIntent.putExtra("articleId",
article.id)

            startActivity(toArticleIntent)
            finish() }}}

    private fun createBackDialog (_header: String,
_alertText:String, _buttonText: String, _negativeButtonText:
String) {
        Log.i("createBackDialog", "Создаем диалог")
        val dialog: Dialog = Dialog(context)
        dialog setContentView(R.layout.custom_dialog)
        val header: TextView =
dialog.findViewById(R.id.dialogHeader)
        val text: TextView = dialog.findViewById(R.id.alertText)
        val positiveButton: Button =
dialog.findViewById(R.id.positiveButton)
        val negativeButton: Button =
dialog.findViewById(R.id.negativeButton)

```

```

        header.setText(_header)
        text.setText(_alertText)
        positiveButton.setText(_buttonText)
        negativeButton.setText(_negativeButtonText)
        positiveButton.isVisible = true
        negativeButton.isVisible = true
        dialog.show()
        negativeButton.setOnClickListener() {
            dialog.dismiss()
        }
        positiveButton.setOnClickListener() {
            val toArticleIntent: Intent = Intent(this,
CreateArticle::class.java)
            toArticleIntent.putExtra("userId", user.id)
            toArticleIntent.putExtra("articleId", article.id)
            startActivity(toArticleIntent)
            finish()
        }

        private fun createOneButtonDialog (_header: String,
_alertText:String, _buttonText: String) {
            Log.d("createDialog", "Создаем диалоговое окно")
            val dialog: Dialog = Dialog(context)
            dialog setContentView(R.layout.custom_dialog)
            val header: TextView =
dialog.findViewById(R.id.dialogHeader)
            val text: TextView = dialog.findViewById(R.id.alertText)
            val okButton: Button =
dialog.findViewById(R.id.positiveButton)
            header.setText(_header)
            text.setText(_alertText)
            okButton.setText(_buttonText)
            okButton.isVisible = true
            dialog.show()
            okButton.setOnClickListener() {
                dialog.dismiss()
            }

            private fun createImportantBlockEditionForm (_isEdition:
Boolean, _block: SpecialBlock?) {
                Log.i("createImportantBlockEditionForm", "Окно
создания/редактирования особого блока")
                val dialog: Dialog = Dialog(context)
                dialog setContentView(R.layout.edit_special_block)
                val positiveButton: Button =
dialog.findViewById(R.id.positiveButton)
                val neutralButton: Button =
dialog.findViewById(R.id.neutralButton)
                val negativeButton: Button =
dialog.findViewById(R.id.negativeButton)
                val text: EditText =
dialog.findViewById(R.id.definitelyUniqueId)

```

```

        if (_block != null) {
            text.setText(_block.getText())
        }
        dialog.show()
        positiveButton.setOnClickListener() {
            if (text.text.isNullOrEmpty()) {
                createOneButtonDialog("Пустой блок", "Для
сохранения необходимо ввести текст", "ОК")
            } else if (!_isEdition) {
                var synthId = importantEdited.size
                synthId = 0 - synthId
                importantBlocks.add(SpecialBlock(synthId,
text.text.toString()))
                importantAdapter.setData(importantBlocks)

                importantEdited.put(synthId, "new")
                isEdited = true
            } else {
                importantBlocks.remove(_block)
                importantBlocks.add(SpecialBlock(_block!!.getId(),
text.text.toString()))
                importantAdapter.setData(importantBlocks)
                importantEdited.replace(_block.getId(), "new")
                isEdited = true
            }
            dialog.dismiss()

            neutralButton.setOnClickListener() {
                if (!_isEdition) {
                    //Удаление "нового" блока = Кнопка "Отмена" =
просто закрыть диалог
                    dialog.dismiss()
                } else {
                    //Удаление ранее имеющегося блока с формы
                    importantBlocks.remove(_block)
                    importantEdited.replace(_block!!.getId(),
"removed")

                    importantAdapter.notifyDataSetChanged()
                    dialog.dismiss()
                }
            }
            negativeButton.setOnClickListener() {
                dialog.dismiss()
            }
        }

        private fun createClarificationBlockEditionForm (_isEdition:
Boolean, _block: SpecialBlock?) {
            Log.i("createClarificationBlockEditionForm", "Окно
создания/редактирования особого блока")
            val dialog: Dialog = Dialog(context)
            dialog setContentView(R.layout.edit_special_block)
            val positiveButton: Button =
dialog.findViewById(R.id.positiveButton)
            val neutralButton: Button =
dialog.findViewById(R.id.neutralButton)

```

```

        val negativeButton: Button =
dialog.findViewById(R.id.negativeButton)
        val text: EditText =
dialog.findViewById(R.id.definitelyUniqueId)
        if (_block != null) {
            text.setText(_block.getText())
        }
dialog.show()

        positiveButton.setOnClickListener() {
            if (text.text.isNullOrEmpty()) {
                createOneButtonDialog("Пустой блок", "Для
сохранения необходимо ввести текст", "OK")
            } else if (!_isEdition) {
                var synthId = clarificationEdited.size
                synthId = 0 - synthId
                clarifications.add(SpecialBlock(synthId,
text.text.toString()))
                clarificationAdapter.setData(clarifications)
                clarificationEdited.put(synthId, "new")
                isEdited = true
            } else {
                clarifications.remove(_block)
clarifications.add(SpecialBlock(_block!!.getId(),
text.text.toString()))
                clarificationAdapter.setData(clarifications)
                clarificationEdited.replace(_block.getId(),
"new")

                isEdited = true }
dialog.dismiss() }

        neutralButton.setOnClickListener() {
            if (!_isEdition) {
                dialog.dismiss()
            } else {
                clarifications.remove(_block)
                clarificationEdited.replace(_block!!.getId(),
"removed")

                clarificationAdapter.notifyDataSetChanged()
                dialog.dismiss() }}
        negativeButton.setOnClickListener() {
            dialog.dismiss() }}}

```

Приложение В.17

Файл DBContract.kt

```

package com.example.lab7
import android.provider.BaseColumns
class DBContract {

```

```

object TableUsers : BaseColumns {
    const val TABLE_NAME = "user"
    const val COLUMN_NAME_KEY_ID = "id"
    const val COLUMN_NAME_LOGIN = "login"
    const val COLUMN_NAME_HASH = "hash"
    const val COLUMN_NAME_ROLE = "role" }

object TableArticles : BaseColumns {
    const val TABLE_NAME = "article"
    const val COLUMN_NAME_KEY_ID = "id"
    const val COLUMN_NAME_NAME = "name"
    const val COLUMN_NAME_TEXT = "text"
    const val COLUMN_NAME_CREATED_AT = "created_at"
    const val COLUMN_NAME_CREATED_BY = "created_by"
    const val COLUMN_NAME_UPDATED_AT = "updated_at"
    const val COLUMN_NAME_UPDATED_BY = "updated_by"
    const val COLUMN_NAME_IMPORTANT_BLOCK_FLAG =
"has_important_block"
    const val COLUMN_NAME_CLARIFICATION_BLOCK_FLAG =
"has_clarification_block"}

object TableImportant : BaseColumns {
    const val TABLE_NAME = "important_block"
    const val COLUMN_NAME_KEY_ID = "id"
    const val COLUMN_NAME_ARTICLE_ID = "article_id"
    const val COLUMN_NAME_TEXT = "text"}

object TableClarification : BaseColumns {
    const val TABLE_NAME = "clarification_block"
    const val COLUMN_NAME_KEY_ID = "id"
    const val COLUMN_NAME_ARTICLE_ID = "article_id"
    const val COLUMN_NAME_TEXT = "text"}

object TableFavourites : BaseColumns {
    const val TABLE_NAME = "favourite"
    const val COLUMN_NAME_KEY_ID = "id"
    const val COLUMN_NAME_USER_ID = "user_id"
    const val COLUMN_NAME_ARTICLE_ID = "article_id"}}

```

Приложение В.18

Файл DBHandler.kt

```

package com.example.lab7
import android.content.ContentValues
import android.content.Context
import android.database.Cursor
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

```

```

import com.example.hgg.Article
import com.example.hgg.SpecialBlock
import com.example.hgg.User

class DBHandler (context: Context) : SQLiteOpenHelper(context,
DATABASE_NAME, null, DATABASE_VERSION) {
    companion object {
        private const val DATABASE_VERSION = 1
        private const val DATABASE_NAME = "hitchhikers_guide.db"
    }
    override fun onCreate(db: SQLiteDatabase) {
        val createUsersTable = "CREATE TABLE IF NOT EXISTS
${DBContract.TableUsers.TABLE_NAME} (" +
            "${DBContract.TableUsers.COLUMN_NAME_KEY_ID}
INTEGER PRIMARY KEY AUTOINCREMENT," + //первичный ключ
            "${DBContract.TableUsers.COLUMN_NAME_LOGIN} TEXT
UNIQUE," +
            "${DBContract.TableUsers.COLUMN_NAME_HASH}
TEXT," +
            "${DBContract.TableUsers.COLUMN_NAME_ROLE}
TEXT)"
        val createArticlesTable =
            "CREATE TABLE IF NOT EXISTS
${DBContract.TableArticles.TABLE_NAME} (" +
            "${DBContract.TableArticles.COLUMN_NAME_KEY_ID} INTEGER PRIMARY
KEY AUTOINCREMENT," + //первичный ключ
            "${DBContract.TableArticles.COLUMN_NAME_NAME} TEXT," +
            "${DBContract.TableArticles.COLUMN_NAME_TEXT} TEXT," +
            "${DBContract.TableArticles.COLUMN_NAME_CREATED_AT} TEXT," +
            "${DBContract.TableArticles.COLUMN_NAME_CREATED_BY} INTEGER," +
            "${DBContract.TableArticles.COLUMN_NAME_UPDATED_AT} TEXT," +
            "${DBContract.TableArticles.COLUMN_NAME_UPDATED_BY} INTEGER," +
            "${DBContract.TableArticles.COLUMN_NAME_IMPORTANT_BLOCK_FLAG}
BOOLEAN DEFAULT 0," +
            "${DBContract.TableArticles.COLUMN_NAME_CLARIFICATION_BLOCK_FLAG}
} BOOLEAN DEFAULT 0)"
        val createImportantBlockTable =
            "CREATE TABLE IF NOT EXISTS
${DBContract.TableImportant.TABLE_NAME} (" +

```

```

"${DBContract.TableImportant.COLUMN_NAME_KEY_ID} INTEGER PRIMARY
KEY AUTOINCREMENT," + //первичный ключ

"${DBContract.TableImportant.COLUMN_NAME_ARTICLE_ID} INTEGER," +

"${DBContract.TableImportant.COLUMN_NAME_TEXT} TEXT, " +
        "CONSTRAINT important_block_fkey FOREIGN KEY
(${DBContract.TableImportant.COLUMN_NAME_ARTICLE_ID}) " +
        "
        REFERENCES
${DBContract.TableArticles.TABLE_NAME}
(${DBContract.TableArticles.COLUMN_NAME_KEY_ID}) " +
        "
        ON UPDATE CASCADE " +
        "
        ON DELETE CASCADE)"

val createClarificationBlockTable =
    "CREATE TABLE IF NOT EXISTS
${DBContract.TableClarification.TABLE_NAME} (" +

"${DBContract.TableClarification.COLUMN_NAME_KEY_ID} INTEGER
PRIMARY KEY AUTOINCREMENT," + //первичный ключ

"${DBContract.TableClarification.COLUMN_NAME_ARTICLE_ID}
INTEGER," +

"${DBContract.TableClarification.COLUMN_NAME_TEXT} TEXT, " +
        "CONSTRAINT clarification_block_fkey FOREIGN
KEY (${DBContract.TableClarification.COLUMN_NAME_ARTICLE_ID}) "
+
        "
        REFERENCES
${DBContract.TableArticles.TABLE_NAME}
(${DBContract.TableArticles.COLUMN_NAME_KEY_ID}) " +
        "
        ON UPDATE CASCADE " +
        "
        ON DELETE CASCADE)"

val createFavouritesTable =
    "CREATE TABLE IF NOT EXISTS
${DBContract.TableFavourites.TABLE_NAME} (" +

"${DBContract.TableFavourites.COLUMN_NAME_KEY_ID} INTEGER
PRIMARY KEY AUTOINCREMENT," + //первичный ключ

"${DBContract.TableFavourites.COLUMN_NAME_USER_ID} INTEGER," +

"${DBContract.TableFavourites.COLUMN_NAME_ARTICLE_ID} INTEGER, "
+
        "CONSTRAINT favourite_user_fkey FOREIGN KEY
(${DBContract.TableFavourites.COLUMN_NAME_USER_ID}) " +
        "
        REFERENCES
${DBContract.TableUsers.TABLE_NAME}
(${DBContract.TableUsers.COLUMN_NAME_KEY_ID}) " +

```

```

        "            ON UPDATE CASCADE " +
        "            ON DELETE CASCADE, " +
        "CONSTRAINT favourite_article_fkey FOREIGN
KEY (${DBContract.TableFavourites.COLUMN_NAME_ARTICLE_ID}) " +
        "            REFERENCES
${DBContract.TableArticles.TABLE_NAME}
(${DBContract.TableArticles.COLUMN_NAME_KEY_ID}) " +
        "            ON UPDATE CASCADE " +
        "            ON DELETE CASCADE) "
    db.execSQL(createUsersTable)
    db.execSQL(createArticlesTable)
    db.execSQL(createImportantBlockTable)
    db.execSQL(createClarificationBlockTable)
    db.execSQL(createFavouritesTable)

    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int,
newVersion: Int) {
        db.execSQL("DROP TABLE IF EXISTS
${DBContract.TableUsers.TABLE_NAME}")
        db.execSQL("DROP TABLE IF EXISTS
${DBContract.TableArticles.TABLE_NAME}")
        db.execSQL("DROP TABLE IF EXISTS
${DBContract.TableImportant.TABLE_NAME}")
        db.execSQL("DROP TABLE IF EXISTS
${DBContract.TableClarification.TABLE_NAME}")
        onCreate(db)

        fun addUser (_login: String, _hash: String, _role: String) {
            val insertQuery = "INSERT INTO
${DBContract.TableUsers.TABLE_NAME} " +
                "(${DBContract.TableUsers.COLUMN_NAME_LOGIN},
${DBContract.TableUsers.COLUMN_NAME_HASH},
${DBContract.TableUsers.COLUMN_NAME_ROLE}) " +
                "VALUES ('" + _login + "', '" + _hash + "', '" +
                _role + "')"

            val db = this.writableDatabase
            db.execSQL(insertQuery)

            fun updateUser (_id: Int, _role: String) {
                val updateQuery = "UPDATE
${DBContract.TableUsers.TABLE_NAME} " +
                    "SET ${DBContract.TableUsers.COLUMN_NAME_ROLE} =
'" + _role + "'" " +
                    "WHERE
${DBContract.TableUsers.COLUMN_NAME_KEY_ID} = " + _id

                val db = this.writableDatabase
                db.execSQL(updateQuery)
            }
        }
    }

```

```

        fun getUser (_login: String) : User? {
            val selectQuery = "SELECT * FROM
${DBContract.TableUsers.TABLE_NAME} WHERE
${DBContract.TableUsers.COLUMN_NAME_LOGIN} = '" + _login + "'"
            val db = this.writableDatabase

            val cursor: Cursor = db.rawQuery(selectQuery, null)
            if (cursor.moveToFirst()) {
                var id: Int
                var login: String
                var role: String

                do {
                    id = cursor.getString(

cursor.getColumnIndexOrThrow(DBContract.TableUsers.COLUMN_NAME_KEY_ID)

                    ).toInt()
                    login = cursor.getString(

cursor.getColumnIndexOrThrow(DBContract.TableUsers.COLUMN_NAME_LOGIN)

                    )
                    role = cursor.getString(

cursor.getColumnIndexOrThrow(DBContract.TableUsers.COLUMN_NAME_ROLE)

                    )
                } while (cursor.moveToNext())
                cursor.close()
                return User(id, login, role)
            } else {
                cursor.close()
                return null}}

        fun getUser (_id: Int) : User? {
            val selectQuery = "SELECT * FROM
${DBContract.TableUsers.TABLE_NAME} WHERE
${DBContract.TableUsers.COLUMN_NAME_KEY_ID} = " + _id
            val db = this.writableDatabase
            val cursor: Cursor = db.rawQuery(selectQuery, null)
            if (cursor.moveToFirst()) {
                var id: Int
                var login: String
                var role: String
                do {
                    id = cursor.getString(

cursor.getColumnIndexOrThrow(DBContract.TableUsers.COLUMN_NAME_KEY_ID)

```

```

        ).toInt()
        login = cursor.getString(

cursor.getColumnIndexOrThrow(DBContract.TableUsers.COLUMN_NAME_LOGIN)

        )
        role = cursor.getString(

cursor.getColumnIndexOrThrow(DBContract.TableUsers.COLUMN_NAME_ROLE)

        )
    } while (cursor.moveToNext())
    cursor.close()
    return User(id, login, role)
} else {
    cursor.close()
    return null}}

fun getUsers (_queryLogin: String): ArrayList<User>? {
    val selectQuery = "SELECT * FROM
${DBContract.TableUsers.TABLE_NAME} WHERE
${DBContract.TableUsers.COLUMN_NAME_LOGIN} LIKE '%" +
_queryLogin + "%'"
    val db = this.writableDatabase
    val usersList: ArrayList<User> = ArrayList<User>()
    val cursor: Cursor = db.rawQuery(selectQuery, null)
    if (cursor.moveToFirst()) {
        var id: Int
        var login: String
        var role: String
        do {
            id = cursor.getString(

cursor.getColumnIndexOrThrow(DBContract.TableUsers.COLUMN_NAME_KEY_ID)

        ).toInt()
            login = cursor.getString(

cursor.getColumnIndexOrThrow(DBContract.TableUsers.COLUMN_NAME_LOGIN)

        )
            role = cursor.getString(

cursor.getColumnIndexOrThrow(DBContract.TableUsers.COLUMN_NAME_ROLE)

        )
            usersList.add(User(id, login, role))
        } while (cursor.moveToNext())
    cursor.close()
    return usersList
}

```

```

    } else {
        cursor.close()
        return null}}

fun getUserHash (_login: String) : String? {
    val selectQuery = "SELECT * FROM
${DBContract.TableUsers.TABLE_NAME} WHERE
${DBContract.TableUsers.COLUMN_NAME_LOGIN} = '" + _login + "'"
    val db = this.writableDatabase

    lateinit var resultHash: String
    val cursor: Cursor = db.rawQuery(selectQuery, null)
    if (cursor.moveToFirst()) {
        do {
            resultHash =
cursor.getString(cursor.getColumnIndexOrThrow(DBContract.TableUs
ers.COLUMN_NAME_HASH)
        ).toString()
        } while (cursor.moveToNext())
        cursor.close()
        return resultHash
    } else {
        cursor.close()
        return null}}

fun deleteUser (_userId: Int) {
    val deleteQuery = "DELETE FROM
${DBContract.TableUsers.TABLE_NAME} " +
        "WHERE
${DBContract.TableUsers.COLUMN_NAME_KEY_ID} = " + _userId
    val db = this.writableDatabase
    db.execSQL(deleteQuery)
}

fun addArticle(_articleName: String, _text: String,
_creationTime: String, _authorId: Int) {
    val insertQuery = "INSERT INTO
${DBContract.TableArticles.TABLE_NAME} " +
        "(${DBContract.TableArticles.COLUMN_NAME_NAME},
${DBContract.TableArticles.COLUMN_NAME_TEXT}, " +

"${DBContract.TableArticles.COLUMN_NAME_CREATED_AT},
${DBContract.TableArticles.COLUMN_NAME_CREATED_BY}, " +

"${DBContract.TableArticles.COLUMN_NAME_UPDATED_AT},
${DBContract.TableArticles.COLUMN_NAME_UPDATED_BY}) " +
        "VALUES ('" + _articleName + "', '" + _text +
        "', '" +
        _creationTime + "', " + _authorId + ", '" +
        _creationTime + "', " + _authorId + ")"

```

```

        val db = this.writableDatabase
        db.execSQL(insertQuery)

        fun updateArticle(_articleId: Int, _newArticleName: String?,
            _newText: String?, _updateTime: String, _authorId: Int) {
            var updateQuery = "UPDATE
        ${DBContract.TableArticles.TABLE_NAME} " +
            "SET
        ${DBContract.TableArticles.COLUMN_NAME_UPDATED_AT} = '" +
            _updateTime + "', " +

            "${DBContract.TableArticles.COLUMN_NAME_UPDATED_BY} = " +
            _authorId

            if (!_newArticleName.isNullOrEmpty())
                updateQuery += ",
        ${DBContract.TableArticles.COLUMN_NAME_NAME} = '" +
            _newArticleName + "'"
            if (!_newText.isNullOrEmpty())
                updateQuery += ",
        ${DBContract.TableArticles.COLUMN_NAME_TEXT} = '" + _newText +
            "'"

            updateQuery += " WHERE
        ${DBContract.TableArticles.COLUMN_NAME_KEY_ID} = " + _articleId
            val db = this.writableDatabase
            db.execSQL(updateQuery) }

        fun getArticle(_id: Int): Article? {
            val selectQuery = "SELECT * FROM
        ${DBContract.TableArticles.TABLE_NAME} " +
            "WHERE
        ${DBContract.TableArticles.COLUMN_NAME_KEY_ID} = " + _id
            val db = this.writableDatabase
            val cursor: Cursor = db.rawQuery(selectQuery, null)
            var resArticle: Article? = null
            if (cursor.moveToFirst()) {
                do {
                    val id =

cursor.getString(cursor.getColumnIndexOrThrow(DBContract.TableAr
ticles.COLUMN_NAME_KEY_ID))
                        .toInt()
                    val name =

cursor.getString(cursor.getColumnIndexOrThrow(DBContract.TableAr
ticles.COLUMN_NAME_NAME))
                        .toString()
                    val text =

```

```

cursor.getString(cursor.getColumnIndexOrThrow(DBContract.TableAr
ticles.COLUMN_NAME_TEXT))
        .toString()
        val createdAt =

cursor.getString(cursor.getColumnIndexOrThrow(DBContract.TableAr
ticles.COLUMN_NAME_CREATED_AT))
        .toString()
        val createdBy =

cursor.getString(cursor.getColumnIndexOrThrow(DBContract.TableAr
ticles.COLUMN_NAME_CREATED_BY))
        .toInt()
        val updatedAt =

cursor.getString(cursor.getColumnIndexOrThrow(DBContract.TableAr
ticles.COLUMN_NAME_UPDATED_AT))
        .toString()
        val updatedBy =

cursor.getString(cursor.getColumnIndexOrThrow(DBContract.TableAr
ticles.COLUMN_NAME_UPDATED_BY))
        .toInt()
        val importantFlag =

cursor.getString(cursor.getColumnIndexOrThrow(DBContract.TableAr
ticles.COLUMN_NAME_IMPORTANT_BLOCK_FLAG))
        .toBoolean()
        val clarificationFlag =

cursor.getString(cursor.getColumnIndexOrThrow(DBContract.TableAr
ticles.COLUMN_NAME_CLARIFICATION_BLOCK_FLAG))
        .toBoolean()
        val author: User? = this.getUser(createdAt)
        var authorLogin: String = "Пользователь удалён"
        if (author != null)
            authorLogin = author.login
        val editor: User? = this.getUser(updatedBy)
        var editorLogin: String = "Пользователь удалён"
        if (editor != null)
            editorLogin = editor.login
        resArticle = Article(id, name, text, createdAt,
authorLogin, updatedAt, editorLogin, importantFlag,
clarificationFlag)
        val ib = this.getImportantBlock(resArticle.id)
        if (ib.isEmpty())
            resArticle.hasImportantBlock = false
        else
            for (b in ib)

```

```

        resArticle.setImportantBlock(b)
        val cb =
this.getClarificationBlock(resArticle.id)
        if (cb.isEmpty())
            resArticle.hasClarificationBlock = false
        else
            for (b in cb)
                resArticle.setClarifications(b)
        } while (cursor.moveToNext())
    }
    cursor.close()
    return resArticle }

fun getArticles(_query: String): ArrayList<Article> {
    val selectQuery = "SELECT * FROM
${DBContract.TableArticles.TABLE_NAME} " +
        "WHERE
${DBContract.TableArticles.COLUMN_NAME_NAME} LIKE '%" + _query +
        "%'"
    val db = this.writableDatabase
    val cursor: Cursor = db.rawQuery(selectQuery, null)
    val articleList: ArrayList<Article> =
ArrayList<Article>()
    if (cursor.moveToFirst()) {
        do {
            val id =

cursor.getString(cursor.getColumnIndexOrThrow(DBContract.TableAr
ticles.COLUMN_NAME_KEY_ID))
                .toInt()
            val name =

cursor.getString(cursor.getColumnIndexOrThrow(DBContract.TableAr
ticles.COLUMN_NAME_NAME))
                .toString()
            val text =

cursor.getString(cursor.getColumnIndexOrThrow(DBContract.TableAr
ticles.COLUMN_NAME_TEXT))
                .toString()
            val createdAt =

cursor.getString(cursor.getColumnIndexOrThrow(DBContract.TableAr
ticles.COLUMN_NAME_CREATED_AT))
                .toString()
            val createdBy =

cursor.getString(cursor.getColumnIndexOrThrow(DBContract.TableAr
ticles.COLUMN_NAME_CREATED_BY))
                .toInt()

```

```

        val updatedAt =

cursor.getString(cursor.getColumnIndexOrThrow(DBContract.TableAr
ticles.COLUMN_NAME_UPDATED_AT))
        .toString()
        val updatedBy =

cursor.getString(cursor.getColumnIndexOrThrow(DBContract.TableAr
ticles.COLUMN_NAME_UPDATED_BY))
        .toInt()
        val importantFlag =

cursor.getString(cursor.getColumnIndexOrThrow(DBContract.TableAr
ticles.COLUMN_NAME_IMPORTANT_BLOCK_FLAG))
        .toBoolean()
        val clarificationFlag =

cursor.getString(cursor.getColumnIndexOrThrow(DBContract.TableAr
ticles.COLUMN_NAME_CLARIFICATION_BLOCK_FLAG))
        .toBoolean()
        val author: User? = this.getUser(createdBy)
        var authorLogin: String = "Пользователь удалён"
        if (author != null)
            authorLogin = author.login
        val editor: User? = this.getUser(updatedBy)
        var editorLogin: String = "Пользователь удалён"
        if (editor != null)
            editorLogin = editor.login
        articleList.add(
            Article(id, name, text, createdAt,
authorLogin, updatedAt, editorLogin, importantFlag,
clarificationFlag))
        } while (cursor.moveToNext())
        cursor.close()
        return articleList}

fun deleteArticle(_id: Int) {
    val deleteQuery = "DELETE FROM
${DBContract.TableArticles.TABLE_NAME} " +
        "WHERE
${DBContract.TableArticles.COLUMN_NAME_KEY_ID} = " + _id
    val db = this.writableDatabase
    db.execSQL(deleteQuery)
}

fun addImportantBlock(_articleId: Int, _importantText:
String, _creationTime: String, _authorId: Int) {
    val insertQuery = "INSERT INTO
${DBContract.TableImportant.TABLE_NAME} " +

```

```

"(${DBContract.TableImportant.COLUMN_NAME_ARTICLE_ID},
${DBContract.TableImportant.COLUMN_NAME_TEXT}) " +
        "VALUES (" + _articleId + ", '" + _importantText
+ "'" )"

        val updateArticleQuery = "UPDATE
${DBContract.TableArticles.TABLE_NAME} " +
        "SET
${DBContract.TableArticles.COLUMN_NAME_UPDATED_AT} = '" +
_creationTime + "', " +

"${DBContract.TableArticles.COLUMN_NAME_UPDATED_BY} = " +
_authorId + ", " +

"${DBContract.TableArticles.COLUMN_NAME_IMPORTANT_BLOCK_FLAG} =
1 " +

        "WHERE
${DBContract.TableArticles.COLUMN_NAME_KEY_ID} = " + _articleId

        val db = this.writableDatabase
        db.execSQL(insertQuery)
        db.execSQL(updateArticleQuery)
    }

    fun updateImportantBlock(_blockId: Int, _importantNewText:
String) {
        val updateQuery = "UPDATE
${DBContract.TableImportant.TABLE_NAME} " +
        "SET
${DBContract.TableImportant.COLUMN_NAME_TEXT} = '" +
_importantNewText +
        "WHERE
${DBContract.TableImportant.COLUMN_NAME_KEY_ID} = " + _blockId
        val db = this.writableDatabase
        db.execSQL(updateQuery)
    }

    fun getImportantBlock(_articleId: Int):
ArrayList<SpecialBlock> {
        val selectQuery = "SELECT * FROM
${DBContract.TableImportant.TABLE_NAME} " +
        "WHERE
${DBContract.TableImportant.COLUMN_NAME_ARTICLE_ID} = " +
_articleId
        val db = this.writableDatabase

        val blockList: ArrayList<SpecialBlock> =
ArrayList<SpecialBlock>()
        val cursor: Cursor = db.rawQuery(selectQuery, null)

```

```

        if (cursor.moveToFirst()) {
            do {
                val blockId = cursor.getString(

cursor.getColumnIndexOrThrow(DBContract.TableClarification.COLUMN
N_NAME_KEY_ID))

                .toInt()
                val blockText = cursor.getString(

cursor.getColumnIndexOrThrow(DBContract.TableImportant.COLUMN_NAME_TEXT)

                )
                blockList.add(SpecialBlock(blockId, blockText))
            } while (cursor.moveToNext())
            cursor.close()
        }
        return blockList
    }

    fun deleteImportantBlock(_id: Int) {
        val db = this.writableDatabase
        val deleteImportantQuery = "DELETE FROM
${DBContract.TableImportant.TABLE_NAME} " +
            "WHERE
${DBContract.TableImportant.COLUMN_NAME_KEY_ID} = " + _id
        db.execSQL(deleteImportantQuery)
    }

    fun addClarificationBlock(_articleId: Int,
        _clarificationText: String, _creationTime: String, _authorId:
        Int) {
        val insertQuery = "INSERT INTO
${DBContract.TableClarification.TABLE_NAME} " +

        "(${DBContract.TableClarification.COLUMN_NAME_ARTICLE_ID},
        ${DBContract.TableClarification.COLUMN_NAME_TEXT}) " +
            "VALUES (" + _articleId + ", '" +
            _clarificationText + "')"

        val updateArticleQuery = "UPDATE
${DBContract.TableArticles.TABLE_NAME} " +
            "SET
${DBContract.TableArticles.COLUMN_NAME_UPDATED_AT} = '" +
            _creationTime + "', " +

            "${DBContract.TableArticles.COLUMN_NAME_UPDATED_BY} = " +
            _authorId + ", " +

```

```

"${DBContract.TableArticles.COLUMN_NAME_CLARIFICATION_BLOCK_FLAG
} = 1 " +

        "WHERE
${DBContract.TableArticles.COLUMN_NAME_KEY_ID} = " + _articleId

        val db = this.writableDatabase
        db.execSQL(insertQuery)
        db.execSQL(updateArticleQuery)
    }

    fun updateClarificationBlock(_blockId: Int,
        _clarificationNewText: String) {
        val updateQuery = "UPDATE
${DBContract.TableClarification.TABLE_NAME} " +
            "SET
${DBContract.TableClarification.COLUMN_NAME_TEXT} = '" +
            _clarificationNewText +
            "WHERE
${DBContract.TableClarification.COLUMN_NAME_KEY_ID} = " +
            _blockId
        val db = this.writableDatabase
        db.execSQL(updateQuery)
    }

    fun getClarificationBlock(_articleId: Int):
        ArrayList<SpecialBlock> {
        val selectQuery = "SELECT * FROM
${DBContract.TableClarification.TABLE_NAME} " +
            "WHERE
${DBContract.TableImportant.COLUMN_NAME_ARTICLE_ID} = " +
            _articleId
        val db = this.writableDatabase

        val blockList: ArrayList<SpecialBlock> =
        ArrayList<SpecialBlock>()
        val cursor: Cursor = db.rawQuery(selectQuery, null)

        if (cursor.moveToFirst()) {
            do {
                val blockId = cursor.getString(

                cursor.getColumnIndexOrThrow(DBContract.TableClarification.COLUMN
                N_NAME_KEY_ID))
                    .toInt()
                val blockText = cursor.getString(

                cursor.getColumnIndexOrThrow(DBContract.TableClarification.COLUMN
                N_NAME_TEXT)
            )

```

```

        blockList.add(SpecialBlock(blockId, blockText))
    } while (cursor.moveToNext())
    cursor.close()
}
return blockList
}

fun deleteClarificationBlock(_id: Int) {
    val db = this.writableDatabase
    val deleteImportantQuery = "DELETE FROM
${DBContract.TableClarification.TABLE_NAME} " +
        "WHERE
${DBContract.TableClarification.COLUMN_NAME_KEY_ID} = " + _id
    db.execSQL(deleteImportantQuery)
}

fun addFavourite(_userId: Int, _articleId: Int) {
    val insertQuery = "INSERT INTO
${DBContract.TableFavourites.TABLE_NAME} " +
        "(${DBContract.TableFavourites.COLUMN_NAME_USER_ID},
${DBContract.TableFavourites.COLUMN_NAME_ARTICLE_ID}) " +
        "VALUES (" + _userId + ", " + _articleId + ")"

    val db = this.writableDatabase
    db.execSQL(insertQuery)
}

fun getFavourites(_userId: Int): ArrayList<Int> {
    val selectQuery = "SELECT * FROM
${DBContract.TableFavourites.TABLE_NAME} " +
        "WHERE
${DBContract.TableFavourites.COLUMN_NAME_USER_ID} = " + _userId
    val db = this.writableDatabase

    val favList: ArrayList<Int> = ArrayList<Int>()
    val cursor: Cursor = db.rawQuery(selectQuery, null)

    if (cursor.moveToFirst()) {
        do {
            val favId = cursor.getString(

cursor.getColumnIndexOrThrow(DBContract.TableFavourites.COLUMN_N
AME_ARTICLE_ID))
                .toInt()
            favList.add(favId)
        } while (cursor.moveToNext())
        cursor.close()
    }
    return favList
}

```

```

    }

    fun deleteFavourite(_userId: Int, _articleId: Int) {
        val db = this.writableDatabase
        val deleteImportantQuery = "DELETE FROM
${DBContract.TableFavourites.TABLE_NAME} " +
            "WHERE
${DBContract.TableFavourites.COLUMN_NAME_ARTICLE_ID} = " +
            _articleId +
            " AND
${DBContract.TableFavourites.COLUMN_NAME_USER_ID} = " + _userId
        db.execSQL(deleteImportantQuery)
    }
}

```

Приложение В.19

Файл Favourites.kt

```

package com.example.hgg
import android.app.Activity
import android.content.Context
import android.content.Intent
import android.content.SharedPreferences
import android.os.Bundle

import android.widget.ImageButton
import android.widget.TextView
import androidx.core.view.isVisible
import androidx.recyclerview.widget.DividerItemDecoration
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.example.lab7.DBHandler
class Favourites : Activity() {
    var user: User = User()
    val context: Context = this
    private val dbh: DBHandler = DBHandler(context)
    private lateinit var favouritesList: RecyclerView
    private val favouritesAdapter = SearchAdapter()
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.favourites_screen)
        val sharedPref: SharedPreferences =
            getSharedPreferences("settings", MODE_PRIVATE)
        var userId = sharedPref.getInt("userId", 0)
        if (userId > 0)
            user = dbh.getUser(userId)!!
        val favouritesList: RecyclerView =
            findViewById(R.id.favouritesList)
    }
}

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.16.001 ПЗ

Лист

143

```

        val emptyText: TextView =
findViewById(R.id.emptyListText)
        val backButton: ImageButton =
findViewById(R.id.backToMainFromFavourites)
        favouritesList.layoutManager = LinearLayoutManager(this)

favouritesList.addItemDecoration(DividerItemDecoration(this,
DividerItemDecoration.VERTICAL))
        if (user.id == -1) {
            emptyText.setText("Необходимо авторизоваться, чтобы
сохранять избранные статьи")
        } else {
            val favIdsList: ArrayList<Int> =
dbh.getFavourites(user.id)
            val favArticlesList: ArrayList<Article> =
ArrayList<Article>()

            if (favIdsList.isEmpty()) {
                emptyText.setText("Нет избранных статей")
            } else {
                emptyText.isVisible = false
                for (id in favIdsList) {
                    val article: Article? = dbh.getArticle(id)
                    if (article != null)
                        favArticlesList.add(article) }}
                favouritesAdapter.setData(favArticlesList)
                favouritesList.adapter = favouritesAdapter }

favouritesAdapter.setListener(object:
ArticleClickListener {
    override fun onItemClick(article: Article) {
        val toArticleIntent: Intent =
Intent(this@Favourites, ArticleScreen::class.java)
        toArticleIntent.putExtra("articleId",
article.id)

        toArticleIntent.putExtra("userId", user.id)
        startActivity(toArticleIntent) })})

backButton.setOnClickListener() {
    finish()}}

override fun onResume() {
    super.onResume()
    val emptyText: TextView =
findViewById(R.id.emptyListText)
    val favouritesList: RecyclerView =
findViewById(R.id.favouritesList)
    if (user.id != -1) {
        val favIdsList: ArrayList<Int> =
dbh.getFavourites(user.id)

```

```

        val favArticlesList: ArrayList<Article> =
ArrayList<Article>()
        if (favIdsList.isEmpty()) {
            emptyText.setText("Нет избранных статей")
            emptyText.isVisible = true
        } else {
            emptyText.isVisible = false
            for (id in favIdsList)
                favArticlesList.add(dbh.getArticle(id!!))
            favouritesAdapter.setData(favArticlesList)
            favouritesList.adapter = favouritesAdapter }}}

```

Приложение В.20

Файл MainActivity.kt

```

package com.example.hgg
import android.app.Activity
import android.app.Dialog
import android.content.Context
import android.content.Intent
import android.content.SharedPreferences
import android.os.Bundle

import android.widget.Button
import android.widget.EditText
import android.widget.ImageButton
import android.widget.LinearLayout
import android.widget.TextView
import androidx.core.content.ContextCompat
import androidx.core.view.isVisible
import androidx.recyclerview.widget.DividerItemDecoration
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import at.favre.lib.crypto.bcrypt.BCrypt
import com.example.lab7.DBHandler
import okhttp3.OkHttpClient
import okhttp3.Request
import org.json.JSONObject
import java.io.IOException
class MainActivity : Activity() {
    var user: User = User()
    val context: Context = this
    private lateinit var rvResults: RecyclerView
    private val searchArticlesAdapter = SearchAdapter()
    private val searchUsersAdapter = SearchUserAdapter()
    private val dbh: DBHandler = DBHandler(context)
    var isUsersSearching: Boolean = false
    val httpClient = OkHttpClient()
    override fun onCreate(savedInstanceState: Bundle?) {

```

```

        super.onCreate(savedInstanceState)
        setContentView(R.layout.main_screen)
        val backButton: ImageButton =
findViewById(R.id.backButton)
        val registerButton: ImageButton =
findViewById(R.id.registerButton)
        val updateDBButton: ImageButton =
findViewById(R.id.updateDBButton)
        val favoritesButton: ImageButton =
findViewById(R.id.favoritesListButton)
        val addArticleButton: ImageButton =
findViewById(R.id.addArticleButton)
        val searchField: EditText =
findViewById(R.id.searchField)
        val searchButton:ImageButton =
findViewById(R.id.searchButton)
        val searchResults: RecyclerView =
findViewById(R.id.searchResults)
        val lowAppBar: LinearLayout =
findViewById(R.id.lowAppBarMainScreen)
        val articleFilter:Button =
findViewById(R.id.articleFilter)
        val userFilter:Button = findViewById(R.id.usersFilter)
        val dontPanicText: TextView =
findViewById(R.id.dontPanicText)
        val arguments: Bundle? = intent.extras
        val sharedPref: SharedPreferences =
getSharedPreferences("settings", MODE_PRIVATE)
        var userId = sharedPref.getInt("userId", 0)
        if (userId > 0)
            user = dbh.getUser(userId)!!
        lowAppBar.isVisible = false
        registerButton.isVisible = false
        addArticleButton.isVisible = false
        backButton.isVisible = true
        if (user.id == -1) {
            registerButton.isVisible = true
            backButton.isVisible = false
        } else {
            backButton.isVisible = true}

        if (user.role == "editor" || user.role == "moderator") {
            addArticleButton.isVisible = true}
        registerButton.setOnClickListener() {
            val toAuthorizationIntent: Intent = Intent(this,
Authorization::class.java)
            startActivity(toAuthorizationIntent)
            finish()}
        backButton.setOnClickListener() {

```

```

        val toAuthorizationIntent: Intent = Intent(this,
Authorization::class.java)
        startActivity(toAuthorizationIntent)
        finish() }
        if (user.role == "moderator") {
            lowAppBar.isVisible = true} }
        searchButton.setOnClickListener() {
            rvResults = findViewById(R.id.searchResults)
            rvResults.layoutManager = LinearLayoutManager(this)
            rvResults.addItemDecoration(DividerItemDecoration(this,
DividerItemDecoration.VERTICAL))
            val searchQuery: String =
searchField.text.toString()
            lowAppBar.isVisible = false
            if (!isUsersSearching &&
!searchQuery.isNullOrEmpty()) {
                val results = dbh.getArticles(searchQuery)
                if (results.isEmpty()) {
                    dontPanicText.isVisible = true
                    createOneButtonDialog(
                        "Результатов нет",
                        "По данному запросу не найдена ни одна
статья", "OK" )
                    searchResults.isVisible = false
                    dontPanicText.isVisible = true
                    searchField.text.clear()
                } else {
                    dontPanicText.isVisible = false
                    searchResults.isVisible = true
                    searchArticlesAdapter.setData(results)
                    rvResults.adapter = searchArticlesAdapter}
            } else if (isUsersSearching &&
!searchQuery.isNullOrEmpty()){
                val results = dbh.getUsers(searchQuery)
                if (results == null) {
                    dontPanicText.isVisible = true
                    createOneButtonDialog(
                        "Результатов нет",
                        "По данному запросу не найдено ни одного
пользователя", "OK")
                    searchResults.isVisible = false
                    dontPanicText.isVisible = true
                } else {
                    dontPanicText.isVisible = false
                    searchResults.isVisible = true
                    searchUsersAdapter.setData(results)
                    rvResults.adapter = searchUsersAdapter}
            } else if (searchQuery.isNullOrEmpty()) {
                createOneButtonDialog(
                    "Пустой запрос",

```

```

        "Для выполнения поиска необходимо ввести
запрос", "OK")
        searchResults.isVisible = false
        dontPanicText.isVisible = true}}

        updateDBButton.setOnClickListener() {
            createTwoButtonDialog("Обновление базы данных",
"Выполнять обновление следует только при наличии устойчивого
Интернет-соединения!\nПродолжить?", "Да", "Нет"))
            addArticleButton.setOnClickListener() {
                val toNewArticleIntent: Intent = Intent(this,
CreateArticle::class.java)
                toNewArticleIntent.putExtra("isEdition", false)
                toNewArticleIntent.putExtra("userId", user.id)
                startActivity(toNewArticleIntent) }

            favoritesButton.setOnClickListener() {
                val toFavouritesIntent: Intent = Intent(this,
Favourites::class.java)
                startActivity(toFavouritesIntent) }
            articleFilter.setOnClickListener() {
                isUsersSearching = false
                articleFilter.background =
ContextCompat.getDrawable(context,
R.drawable.filter_button_active)
                userFilter.background =
ContextCompat.getDrawable(context, R.drawable.filter_button) }
            userFilter.setOnClickListener() {
                isUsersSearching = true
                userFilter.background =
ContextCompat.getDrawable(context,
R.drawable.filter_button_active)
                articleFilter.background =
ContextCompat.getDrawable(context, R.drawable.filter_button) }

            searchArticlesAdapter.setListener(object:
ArticleClickListener {
                override fun onItemClick(article: Article) {
                    val toArticleIntent: Intent =
Intent(this@MainActivity, ArticleScreen::class.java)
                    toArticleIntent.putExtra("articleId",
article.id)
                    toArticleIntent.putExtra("userId", user.id)
                    startActivity(toArticleIntent) } })
            searchUsersAdapter.setListener(object: UserClickListener
{
                override fun onDelete(user: User) {
                    dbh.deleteUser(user.id)
                    createOneButtonDialog("Удаление пользователя",
"Пользователь " + user.login + " удалён", "OK")

```

```

searchUsersAdapter.setData(dbh.getUsers(searchField.text.toString()))!!))}

        override fun onReaderClick(user: User) {
            dbh.updateUser(user.id, "reader")
            createOneButtonDialog("Изменение роли", "Роль
пользователя " + user.login + " изменена на \"Читатель\"", "OK")
searchUsersAdapter.setData(dbh.getUsers(searchField.text.toString()))!!))}

        override fun onEditorClick(user: User) {
            dbh.updateUser(user.id, "editor")
            createOneButtonDialog("Изменение роли", "Роль
пользователя " + user.login + " изменена на \"Редактор\"", "OK")
searchUsersAdapter.setData(dbh.getUsers(searchField.text.toString()))!!))}

        override fun onModeratorClick(user: User) {
            dbh.updateUser(user.id, "moderator")
            createOneButtonDialog("Изменение роли", "Роль
пользователя " + user.login + " изменена на \"Модератор\"",
"OK")

searchUsersAdapter.setData(dbh.getUsers(searchField.text.toString()))!!))}}})

        override fun onRestart() {
            super.onRestart()
            val searchField: EditText =
findViewById(R.id.searchField)
            val searchResults: RecyclerView =
findViewById(R.id.searchResults)
            val dontPanicText: TextView =
findViewById(R.id.dontPanicText)
            val searchQuery: String = searchField.text.toString()
            if (!isUsersSearching && !searchQuery.isNullOrEmpty()) {
                val results = dbh.getArticles(searchQuery)
                if (results.isEmpty()) {
                    dontPanicText.isVisible = true
                    searchResults.isVisible = false
                    dontPanicText.isVisible = true
                    searchField.text.clear()
                } else {
                    dontPanicText.isVisible = false
                    searchResults.isVisible = true
                    searchArticlesAdapter.setData(results)
                    rvResults.adapter = searchArticlesAdapter
                }
            } else if (isUsersSearching &&
!searchQuery.isNullOrEmpty()){
                val results = dbh.getUsers(searchQuery)
                if (results == null) {
                    dontPanicText.isVisible = true

```

```

        searchResults.isVisible = false
        dontPanicText.isVisible = true
    } else {
        dontPanicText.isVisible = false
        searchResults.isVisible = true
        searchUsersAdapter.setData(results)
        rvResults.adapter = searchUsersAdapter
    }
}

private fun createOneButtonDialog (_header: String,
    _alertText:String, _buttonText: String) {
    Log.d("createDialog", "Создаем диалоговое окно")
    val dialog: Dialog = Dialog(context)
    dialog setContentView(R.layout.custom_dialog)
    val header: TextView =
dialog.findViewById(R.id.dialogHeader)
    val text: TextView = dialog.findViewById(R.id.alertText)
    val okButton: Button =
dialog.findViewById(R.id.positiveButton)
    header.setText(_header)
    text.setText(_alertText)
    okButton.setText(_buttonText)
    okButton.isVisible = true
    dialog.show()
    okButton.setOnClickListener() {
        dialog.dismiss()}}

private fun createTwoButtonDialog (_header: String,
    _alertText:String, _buttonText: String, _negativeButtonText:
String) {
    Log.i("createTwoButtonDialog", "Создаем диалог")
    val dialog: Dialog = Dialog(context)
    dialog setContentView(R.layout.custom_dialog)
    val header: TextView =
dialog.findViewById(R.id.dialogHeader)
    val text: TextView = dialog.findViewById(R.id.alertText)
    val positiveButton: Button =
dialog.findViewById(R.id.positiveButton)
    val negativeButton: Button =
dialog.findViewById(R.id.negativeButton)
    header.setText(_header)
    text.setText(_alertText)
    positiveButton.setText(_buttonText)
    negativeButton.setText(_negativeButtonText)
    positiveButton.isVisible = true
    negativeButton.isVisible = true
    dialog.show()
    negativeButton.setOnClickListener() {
        dialog.dismiss()
    }
}

```

```

    }
    positiveButton.setOnClickListener() {
        var newResponse = ""

        Thread {

            val request = Request.Builder()
                .url("https://localhost:8081/api/v1/articles")
                .build()
            try {
                httpClient.newCall(request).execute().use {
                    response ->
                    if (!response.isSuccessful) {
                        createOneButtonDialog("Что-то пошло
не так", "Не удалось обновить базу данных.\nПовторить попытку
позже", "OK")
                    } else {
                        val jsonResponse =
response.body!!.toString()}}
                    } catch (e: IOException) {
                        createOneButtonDialog("Что-то пошло не так",
"Невозможно выполнить запрос при нестабильном интернет-
соединении.\nПовторите попытку позже", "OK")
                    }
                }.start()}}}}

```

Приложение В.21

Файл Registration.kt

```

package com.example.hgg
import android.app.Activity
import android.app.Dialog
import android.content.Context
import android.content.Intent
import android.content.SharedPreferences
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.ImageButton
import android.widget.TextView
import androidx.core.view.isVisible
import at.favre.lib.crypto.bcrypt.BCrypt
import com.example.lab7.DBHandler
import okhttp3.FormBody
import okhttp3.OkHttpClient
import okhttp3.Request
import okhttp3.RequestBody
import org.json.JSONObject
import java.io.IOException

```

```

class Registration : Activity() {
    val context: Context = this
    private val dbh: DBHandler = DBHandler(context)
    private var user: User = User()
    val httpClient = OkHttpClient()
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.registration_screen)
        val backButton: ImageButton =
findViewById(R.id.backToMainFromRegistrationButton)
        val loginField: TextView =
findViewById(R.id.newLoginField)
        val passwordField: EditText =
findViewById(R.id.newPasswordField)
        val passwordRepeatField: EditText =
findViewById(R.id.repeatPasswordField)
        val registrationButton: Button =
findViewById(R.id.registrationButton)
        val toAuthorizationButton: Button =
findViewById(R.id.toAuthorizationButton)
        backButton.setOnClickListener() {
            val toMainScreenIntent: Intent = Intent(this,
MainActivity::class.java)
            startActivity(toMainScreenIntent)
            finish()
        }
        registrationButton.setOnClickListener() {
            val login: String = loginField.text.toString()
            val password: String = passwordField.text.toString()
            val passwordRepeat: String =
passwordRepeatField.text.toString()
            if (login.isNullOrEmpty() ||
password.isNullOrEmpty() || passwordRepeat.isNullOrEmpty()) {
                createOneButtonDialog(
                    "Чего-то не хватает",
                    "Необходимо указать логин, создать пароль и
подтвердить его", "OK")
            } else if (password != passwordRepeat) {
                loginField.text = ""
                passwordField.text.clear()
                passwordRepeatField.text.clear()
                createOneButtonDialog(
                    "Пароли не совпадают",
                    "Проверьте правильность заполнения паролей и
повторите попытку", "OK")
            } else {
                val bcryptHash: String =
BCrypt.withDefaults().hashToString(12, password.toCharArray())
                Thread {
                    val reqJson = FormBody.Builder()

```

```

        .add("login", login)
        .add("hash", bcryptHash)
        .add("role", "reader")
        .build()
        val request = Request.Builder()
.url("https://localhost:8081/api/v1/users")
        .post(reqJson)
        .build()
        try
{httpClient.newCall(request).execute().use { response ->
            if (!response.isSuccessful) {
                createOneButtonDialog(
                    "Что-то пошло не так",
                    "Не удалось выполнить
регистрацию.\nПовторите попытку позже", "OK")
            } else {
                val response = response.body!!.toString()
                val sharedPref:
SharedPreferences = getSharedPreferences("settings",
MODE_PRIVATE)
                val sharedPrefEditor:
SharedPreferences.Editor = sharedPref.edit()

                sharedPrefEditor.putInt("userId", user.id)
                sharedPrefEditor.apply()
                val toMainScreenIntent: Intent =
Intent(this, MainActivity::class.java)

                startActivity(toMainScreenIntent)
                finish()}} catch (e:
IOException) { createOneButtonDialog(
                    "Что-то пошло не так",
                    "Невозможно выполнить запрос при
нестабильном интернет-соединении.\nПовторите попытку позже",
                    "OK") } }.run()}}

        toAuthorizationButton.setOnClickListener() {
            val toAuthorizationIntent: Intent = Intent(this,
Authorization::class.java)
            startActivity(toAuthorizationIntent)
            finish()}}

        private fun createOneButtonDialog (_header: String,
        _alertText:String, _buttonText: String) {
            val dialog: Dialog = Dialog(context)
            dialog.setContentView(R.layout.custom_dialog)
            val header: TextView =
dialog.findViewById(R.id.dialogHeader)
            val text: TextView = dialog.findViewById(R.id.alertText)

```

```

        val okButton: Button =
dialog.findViewById(R.id.positiveButton)
        header.setText(_header)
        text.setText(_alertText)
        okButton.setText(_buttonText)
        okButton.isVisible = true
        dialog.show()
        okButton.setOnClickListener() {
            dialog.dismiss()}}}

```

Приложение В.22

Файл SearchUserAdapter.kt

```

package com.example.hgg
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageButton
import android.widget.TextView
import androidx.core.view.isVisible
import androidx.recyclerview.widget.RecyclerView
class SearchUserAdapter :
RecyclerView.Adapter<SearchUserAdapter.SearchUserViewHolder>() {
    private var results = listOf<User>()
    override fun onCreateViewHolder(parent: ViewGroup, viewType:
Int): SearchUserViewHolder {
        val view =
LayoutInflater.from(parent.context).inflate(R.layout.users_list_
item, parent, false)
        return SearchUserViewHolder(view) }
    override fun onBindViewHolder(holder: SearchUserViewHolder,
position: Int) = holder.bind(results[position])
    override fun getItemCount(): Int = results.size
    fun setData(resultsList: List<User>) {
        results = resultsList
        notifyDataSetChanged() }
    private var listener: UserClickListener? = null
    fun setListener(_listener: UserClickListener) {
        this.listener = _listener }
    inner class SearchUserViewHolder(itemView: View) :
RecyclerView.ViewHolder(itemView) {
        fun bind(user: User) {
            itemView.findViewById<TextView>(R.id.resultHeader).text
= user.login
            val resultType =
itemView.findViewById<TextView>(R.id.resultType)
            val deleteButton =
itemView.findViewById<ImageButton>(R.id.deleteUser)

```

```

        val makeReaderButton =
itemView.findViewById<ImageButton>(R.id.makeReaderButton)
        val makeEditorButton =
itemView.findViewById<ImageButton>(R.id.makeEditorButton)
        val makeModeratorButton =
itemView.findViewById<ImageButton>(R.id.makeModeratorButton)
        when (user.role) {
            "reader" -> {
                resultType.text = "читатель"
                makeReaderButton.isVisible = false
                makeEditorButton.isVisible = true
                makeModeratorButton.isVisible = true
            }
            "editor" -> {
                resultType.text = "редактор"
                makeReaderButton.isVisible = true
                makeEditorButton.isVisible = false
                makeModeratorButton.isVisible = true}
            "moderator" -> {
                resultType.text = "модератор"
                makeReaderButton.isVisible = true
                makeEditorButton.isVisible = true
                makeModeratorButton.isVisible = false}}
        deleteButton.setOnClickListener {
            listener?.onDelete(user) }
        makeReaderButton.setOnClickListener {
            listener?.onReaderClick(user) }
        makeEditorButton.setOnClickListener {
            listener?.onEditorClick(user) }
        makeModeratorButton.setOnClickListener {
            listener?.onModeratorClick(user) }}}}

```

Приложение В.23

Файл UserClickListener.kt

```

package com.example.hgg
interface UserClickListener {
    fun onDelete(user: User)
    fun onReaderClick(user: User)
    fun onEditorClick(user: User)
    fun onModeratorClick(user: User)}

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.16.001 ПЗ

Лист

155

Приложение Г

(Обязательное)

Информационно-справочная система статей энциклопедического характера

Презентация

					ПГУ 1.090301.16.001 ПЗ	Лист
						156
Изм.	Лист	№ докум.	Подпись	Дата		

Информационно-справочная система статей энциклопедического характера

Подготовил: Макарова А.Ю., группа 20ВВП1
Руководитель: к.т.н., доцент каф. ВТ Синев М.П.

Тема работы

«Автостопом по Галактике: Путеводитель вольного странника»

The Hitchhiker's Guide to the Galaxy

- ★ книга, написанная не на Земле, никогда на Земле не публиковавшаяся
- ★ как можно заключить по ее заглавию, это вообще-то классический путеводитель
- ★ на вид это точь-в-точь маленький, плоский, гибкий компьютер класса «лэптоп»
- ★ статьи для «Путеводителя» сочиняет бесчисленное множество исследователей и путешественников
- ★ для решения вопроса об обновлении глав ключевым моментом является одно — увлекательно ли их читать

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.16.001 ПЗ

Лист

157

Постановка задачи

Требуется разработать информационно-справочную систему, основными объектами в которой являются текстовые статьи.

Доступ пользователей к базе статей должен осуществляться посредством мобильного клиента. Должны быть предоставлены возможности по просмотру и изменению базы статей.

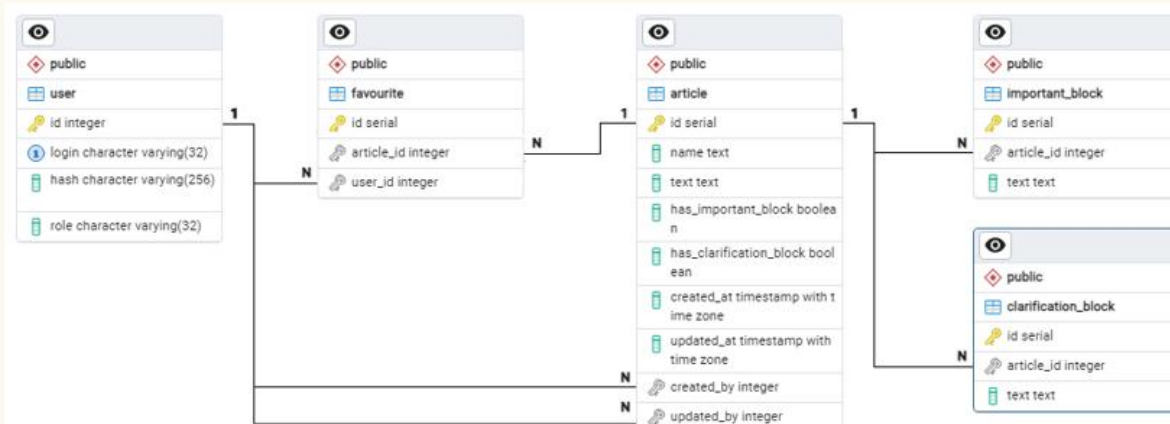


Стек технологий



Изм.	Лист	№ докум.	Подпись	Дата

Физическая схема базы данных



Результаты выполнения некоторых запросов

Добавление статьи

```

POST localhost8081/api/v1/articles
{
  "title": "Земля",
  "text": "Земля",
  "created_at": "2024-05-13 15:00:00+03",
  "created_by": 2,
  "updated_at": "2024-05-13 15:00:00+03",
  "updated_by": 2,
  "has_important_block": false,
  "has_clarification_block": false
}
200 OK 12.9 ms 501 B
  
```

```

GET localhost8081/api/v1/users/login/editor2
{
  "id": 5,
  "login": "editor2",
  "hash": "52a51253b85f571/V8W9pGAKQq_YDTeCn.V8JwLS85j6vlySF2vndp26",
  "role": "editor"
}
200 OK 410 ms 112 B
  
```

Поиск пользователя по логину

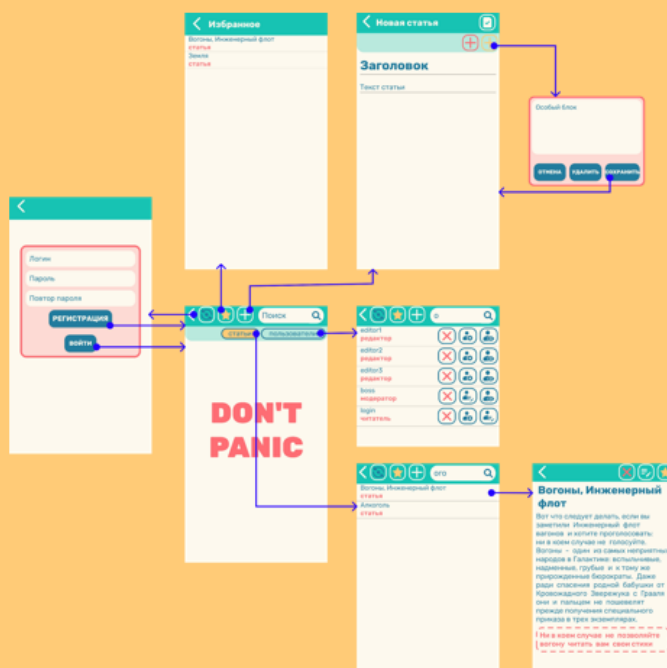
Добавление блока «Важное»

```

POST localhost8081/api/v1/important
{
  "text": "Не в коем случае не позволяйте волгу читать вам свои стихи",
  "article_id": 6
}
200 OK 8.73 ms 142 B
  
```

Карта экранов

- ★ Основной экран
- ★ Экран авторизации/регистрации
- ★ Экран просмотра статьи
- ★ Экран создания / редактирования статьи
- ★ Избранное



Заключение

В результате выполнения работы:

- ★ выполнен анализ предметной области, сравнительный анализ аналогов;
- ★ выявлены требования к системе;
- ★ спроектирована архитектура базы данных и REST API;
- ★ разработано серверное приложение для взаимодействия со справочной системой статей;
- ★ разработано мобильное приложение для устройств на платформе Android.

Закреплены навыки работы с фреймворком Spring, СУБД PostgreSQL и SQLite. Использованы интегрированные среды разработки IntelliJ IDEA и Android Studio. В результате разработана полноценная справочная система, состоящая из мобильного клиента, и серверной части, содержащей основную базу данных системы.

Изм.	Лист	№ докум.	Подпись	Дата