

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”

Факультет компьютерных наук

Кафедра программирования и информационных технологий


Аналог Твиттера


Курсовой проект

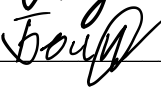
09.03.04 Программная инженерия

Информационные системы и сетевые технологии


Зав. кафедрой _____ С.Д. Махортов, д.ф.- м.н., доцент _____.20__

Обучающийся  _____ А.В. Фуфаева, 3 курс, д/о

Обучающийся  _____ М.С. Ларионов, 3 курс, д/о

Обучающийся  _____ А.Д. Болдырев, 3 курс, д/о

Руководитель _____ В.С. Тарасов, ст. преподаватель

Руководитель  _____ И.В. Клейменов, ассистент

Воронеж 2022

Содержание

Анализ предметной области.	3
Постановка задачи.	4
Функциональные требования	4
Технические требования	5
Требования к интерфейсу	5
Макеты интерфейса	5
Обзор аналогов.	11
Twitter	11
Функциональность	11
Интерфейс	12
Достоинства и недостатки	15
Вывод	16
Входные-выходные данные. Диаграмма IDEF0.	16
Диаграмма вариантов использования UseCase.	16
Диаграмма состояний.	18
Структурная схема приложения.	19
Схема базы данных.	19
Диаграмма потоков данных.	20
Диаграмма классов.	20
Диаграммы классов сервера	20
Диаграмма классов библиотеки controller	20
Диаграмма классов библиотеки dto	21
Диаграмма классов библиотеки entity	23
Диаграмма классов библиотеки mapper	23
Диаграмма классов библиотеки repository	24
Диаграммы классов клиента	24
Диаграмма главного класса Main	24
Диаграмма классов библиотеки ui	24
Сценарии воронок.	26
Воронка «Новый активный пользователь»	26
Воронка «Популярный пост»	27
Воронка «Авторизация пользователя»	27
Используемая платформа.	28
Тестирование.	28
Список планируемых работ.	32
Календарный план.	33
Отзывы пользователей.	34
Заключение.	36

Анализ предметной области.

Социальная сеть (сокр. соцсеть)— онлайн-платформа, которая используется для общения, знакомств, создания социальных отношений между людьми, которые имеют схожие интересы или офлайн-связи, а также для развлечения (музыка, фильмы) и работы.

В настоящее время количество социальных сетей и численность их участников растет с невероятной быстротой. Социальные сети сегодня уже посещает более чем две трети онлайн-аудитории во всем мире, и это четвертая по популярности онлайн-категория после поисковых порталов, информационных порталов и программного обеспечения, которая опережает даже электронную почту (по данным компании Nielsen Online, исследующей онлайн поведение в 9 странах). По данным той же компании, использование онлайн-сообществ сегодня растет вдвое более быстрыми темпами, чем любой из четырех других секторов сети Интернета и в три раза быстрее, чем пользование Интернетом в целом.

Социальные сети привлекают людей, преследующих различные цели: поддержание контакта со старыми знакомыми и поиск новых, в т. ч. обустройство личной жизни; поиск работы, продвижение своего бизнеса, профессиональное общение; обмен информацией и медиаконтентом с другими пользователями. Исходя из такого разнообразия целей, возрастает и количество социальных сетей, ведь каждая из них имеет общие черты с другими, но остается неповторимой в общей массе.

На данный момент востребованными являются тематические социальные сети, реализующие функционал и принципы общения, используемые в заданной тематике.

К тематическим относят и социальные сети для публичного обмена сообщениями, где пользователь может публиковать короткие заметки в формате блога.

В переводе с английского blog (или web log) означает «интернет-журнал», «онлайн-дневник». В нем пользователь делится своими мыслями, рассказывает о своей жизни, публикует новости или просто информационный материал. Каждое опубликованное сообщение называют постом.

Постановка задачи.

Разработать приложение для публичного обмена сообщениями-постами, с возможностью комментирования постов – прикрепления к посту собственной заметки-комментария, а также оценки постов – отметку «Лайк» можно поставить на любой понравившийся пост.

Функциональные требования.

Неавторизованный пользователь должен иметь возможность:

- Зарегистрироваться в приложении при первичном использовании;
- Авторизоваться в приложении при повторном использовании;
- Просмотреть ленту новых постов пользователей;
- Найти другого пользователя;

Авторизованный пользователь должен иметь возможность:

- Просмотреть ленту новых постов пользователей;
- Найти другого пользователя;
- Подписаться на другого пользователя;
- Отписаться от пользователя, на которого он уже подписан;
- Просмотреть свою личную страницу;
- Добавить новый пост на своей личной странице;
- Удалить пост на своей личной странице;

- Оставить комментарий на любой пост;
- Оставить отметку «Лайк» на любой пост.

Технические требования.

Приложение должно удовлетворять следующим техническим требованиям:

- Соответствие техническому заданию;
- При добавлении новых информационных блоков пользователем – нового поста или комментария, приложение должно обновиться и отобразить новый информационных блок в соответствующем добавлению месте.

Требования к интерфейсу.

Требования к интерфейсу соответствуют Техническому заданию.

Общие требования:

- Все элементы приложения выполнены в едином стиле и тематике.

Основное меню должно соответствовать следующим требованиям:

- Элементы меню должны быть выделены на фоне основной части содержимого приложения.

Основное содержимое страниц должно соответствовать следующим требованиям:

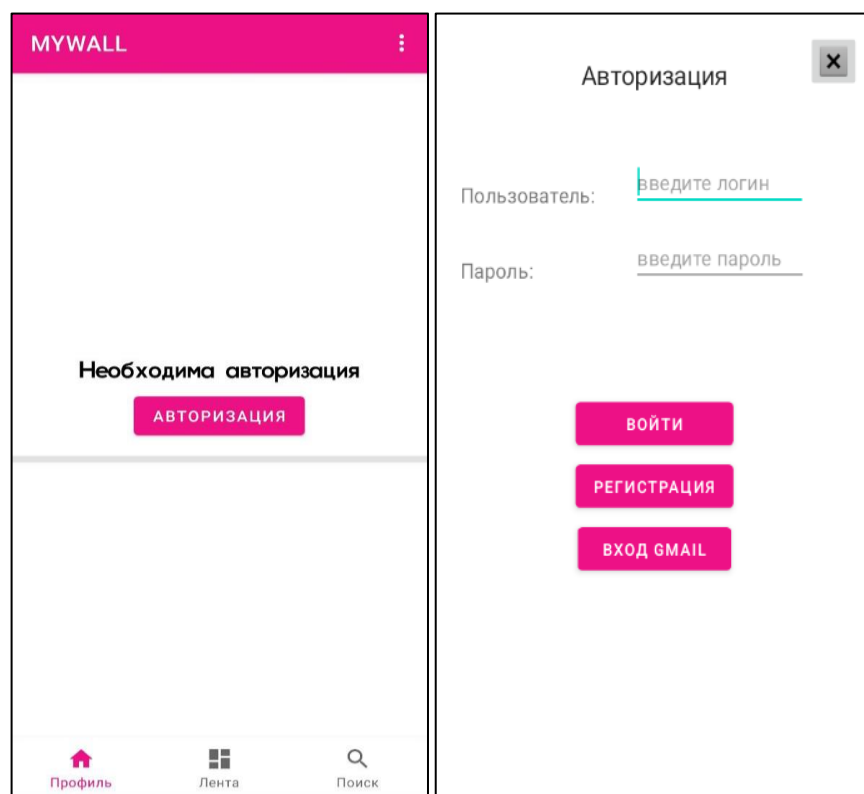
- Шрифт среднего размера, не менее 3 мм;
- Цвет шрифта контрастный на фоне содержимого страницы.

Макеты интерфейса.

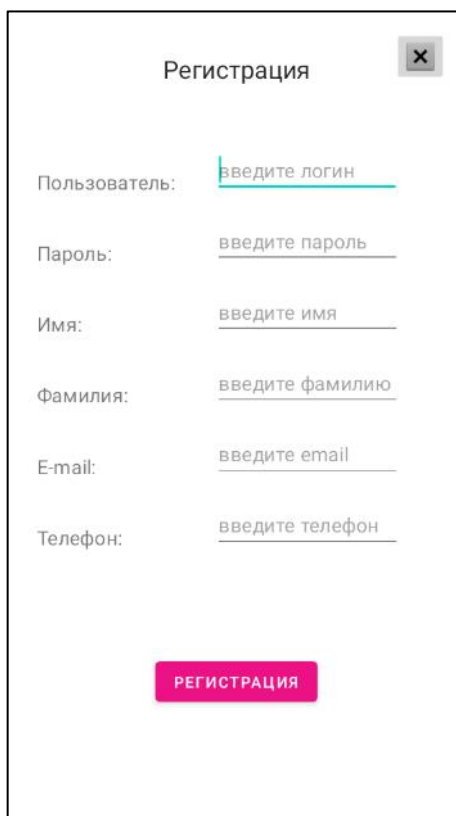
При запуске приложения открывается экран-заставка:



Неавторизованному пользователю доступна авторизация/регистрация:



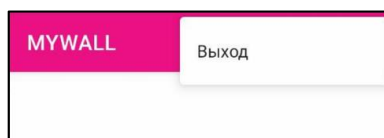
Для перехода на форму регистрации необходимо нажать кнопку «Регистрация»:

A screenshot of a registration form titled "Регистрация" with a close button (X) in the top right corner. The form contains several input fields with placeholder text: "Пользователь:" with "введите логин", "Пароль:" with "введите пароль", "Имя:" with "введите имя", "Фамилия:" with "введите фамилию", "E-mail:" with "введите email", and "Телефон:" with "введите телефон". At the bottom center is a pink button labeled "РЕГИСТРАЦИЯ".

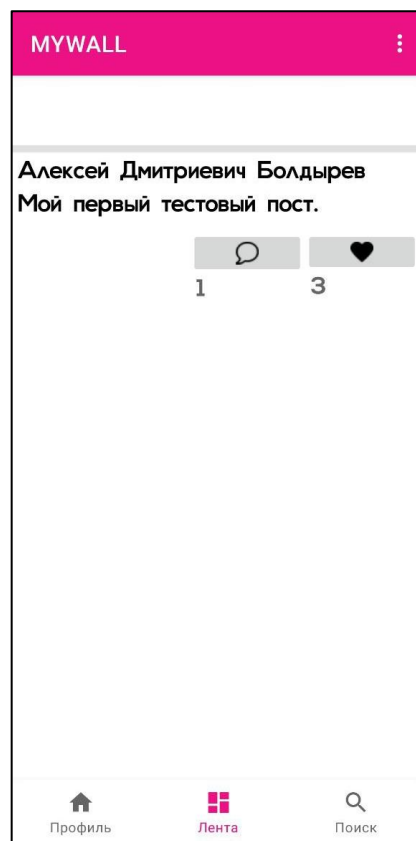
Для перехода между основными страницами приложения используется навигационное меню в нижней части экрана:



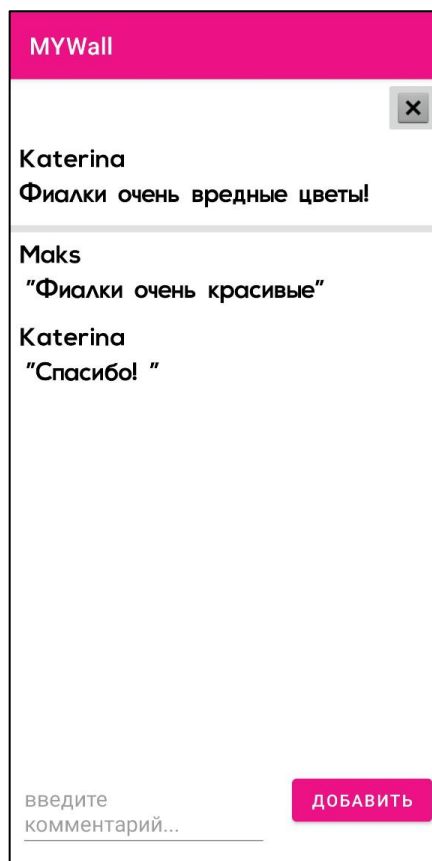
Для выхода из приложения необходимо нажать на кнопку с троеточием в верхнем меню. И выбрать опцию выход:



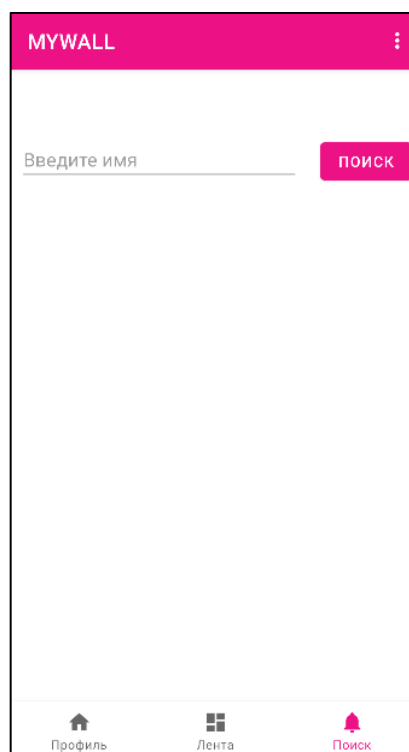
Страница основной ленты доступна для авторизованного и неавторизованного пользователя. Кнопки комментария и отметки «Лайк» активны только для авторизованного пользователя:



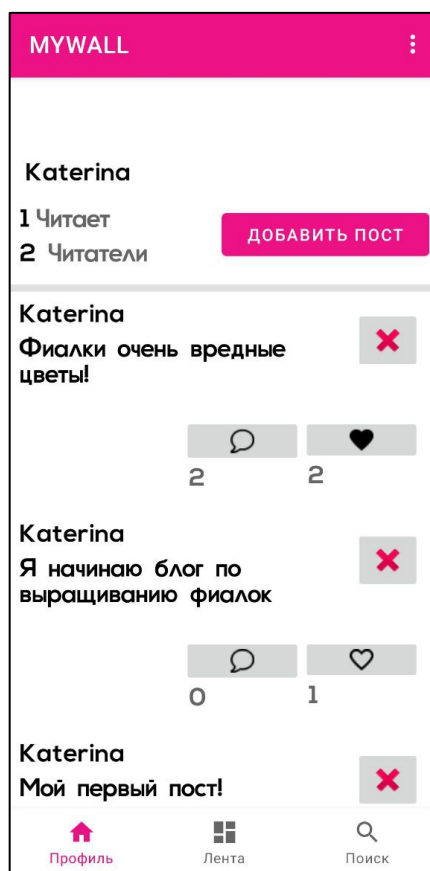
Чтобы оставить комментарий на пост, необходимо нажать кнопку со значком комментария. Откроется форма со всеми комментариями к посту, а также полем ввода комментария и кнопкой добавить:



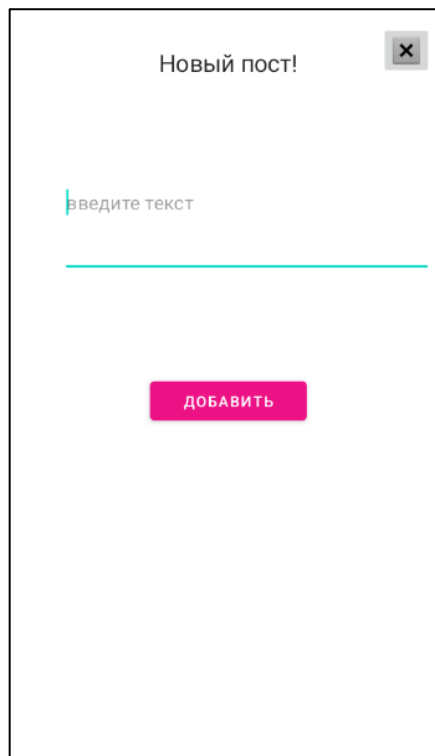
Страница поиска доступна для авторизованного и неавторизованного пользователя:



Личная страница неавторизованного пользователя не содержит информации и включает в себя только кнопку авторизации. Для авторизованного пользователя доступен весь функционал страницы:



Для удаления поста необходимо нажать на крестик рядом с соответствующим постом. Для добавления нового поста необходимо нажать на кнопку «Добавить пост», после чего откроется форма добавления поста:



Обзор аналогов.

В ходе работы над данным проектом был проведен поиск аналогов. Необходимо рассмотреть их достоинства и недостатки, удобство в использовании, содержание необходимого функционала. На основе этого анализа можно сделать выводы о том, каким функционалом должен обладать разрабатываемый продукт.

Twitter.

Функциональность.

Личная страница:

- Изменение данных пользователя;
- Поиск по своим постам;
- Возможность выложить/удалить/изменить новый пост;
- Возможность добавить тему для чтения.

Лента:

- Возможность читать посты по интересным темам;
- Возможность комментировать посты;
- Возможность создавать посты;
- Подписка на получение новых публикаций пользователей;
- Возможность перехода на страничку пользователя.

Поиск:

- Поиск в приложении (по пользователям и по постам;
- Возможность перехода на страничку пользователя;
- Подписка на получение новых публикаций пользователей.

Уведомления:

- Поиск по всем уведомлениям;
- Поиск по упоминаниям.

Обмен сообщениями:

- Возможность писать сообщения пользователям и читать сообщения от пользователей.

Первичный вход:

- Заполнение личных данных;
- Выбор интересных тем (по которым приложение будет показывать посты).

Интерфейс.

Формы заполнения данных для первичного входа:

MTS RUS
Tele2You

14:41

←

✈

запись

Аня

47

annaufufoeva01@gmail.com

22 мая 2001 г.

Эта информация не будет общедоступной. Подтвердите свой возраст, даже если эта учетная запись предназначена для компании, домашнего животного и т. д.

Далее

21

апр.

2000

22

май

2001

23

июнь

2002

◀ ○ ▶

MTS RUS
Tele2You

14:42

✈

Как вас называть?

Ваше имя пользователя является уникальным. Его можно изменить в любое время.

@Имя пользователя

@annaufufoeva01, @Ana011921951

Показать больше

Пропустить

Далее

◀ ○ ▶

MTS RUS
Tele2You

14:42

✈

Выберите изображение профиля

Загрузите свое лучшее селфи.

+

Загрузить

Пропустить

Далее

◀ ○ ▶

MTS RUS
Tele2You

14:42

✈

Опишите себя

Чем вы отличаетесь от других? Особо не раздумывайте, просто напишите что придет в голову.

Расскажите о себе

160

Пропустить

Далее

◀ ○ ▶

MTS RUS
Tele2You

14:43

✈

Что вы хотите видеть в Твиттере?

Список ваших интересов используется для персонализации и виден в вашем профиле.

Видеоигры

Игры

Minecraft

PlayStation

Call of Duty

Xbox

Epic Games

Animal Crossing

League of Legends

Музыка

К-роп

Поп-музыка

Рок

Хип-хоп/Рэп

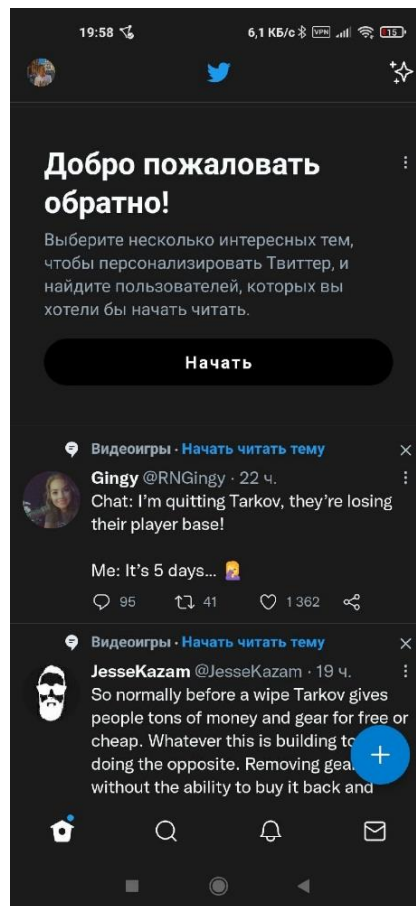
Новости музыки

Пропустить

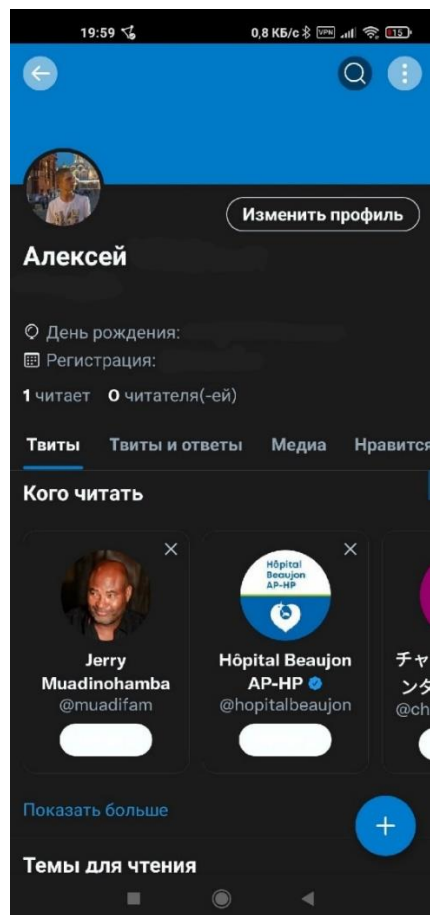
Далее

◀ ○ ▶

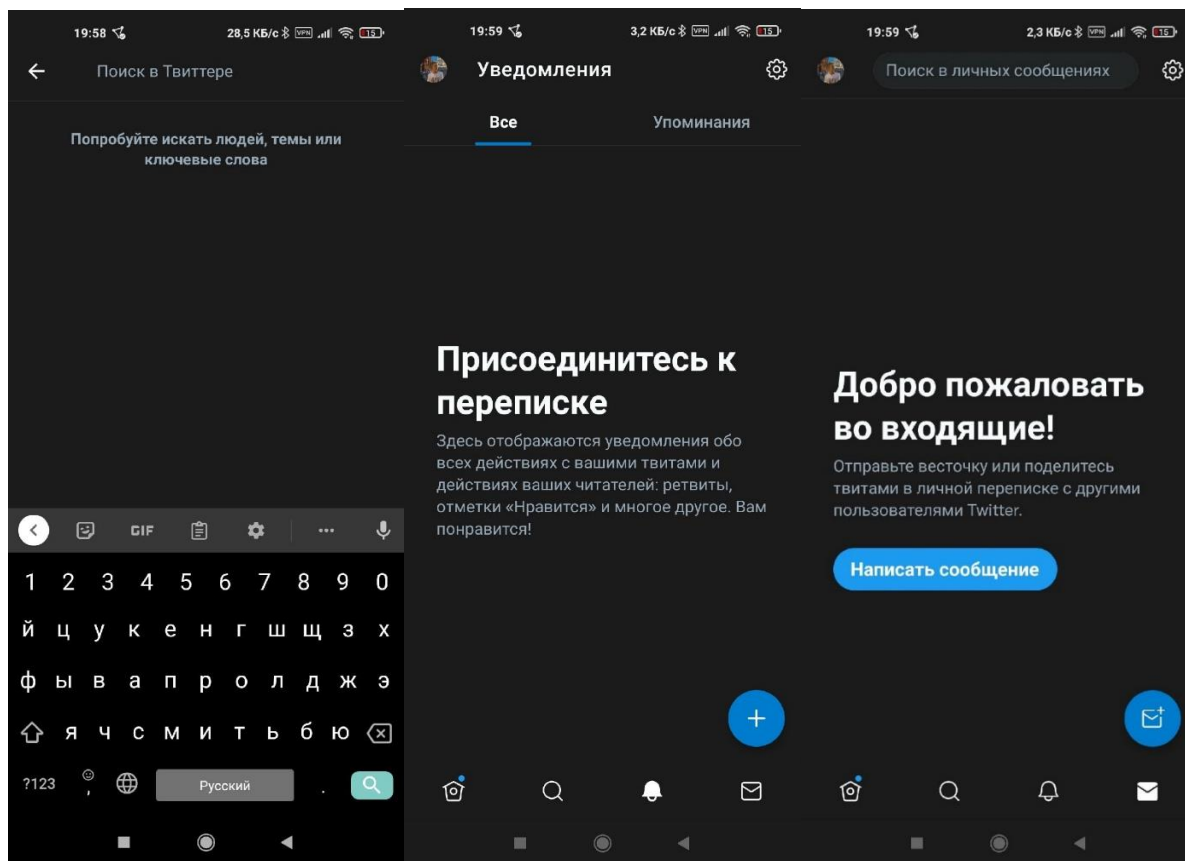
При повторном входе открывается приветственное окно:



Личная страница пользователя:



Вкладки «Поиск», «Уведомления», «Сообщения»:



Достоинства и недостатки.

Достоинства:

- простой и понятный интерфейс;
- удобная навигационная панель;
- на страницах много подсказок;
- приложение существует на разных платформах, а также в web-версии;
- невысокие требования к системным ресурсам;
- большая популярность;
- поддержка и обновления приложения.

Недостатки:

- доступно в России только через VPN (с недавнего времени);

- сильная модерация – постоянная проверка постов на соответствие тематике и правилам ресурса.

Вывод.

ПО имеет большой функционал, красивый, понятный и удобный интерфейс, и другие плюсы, но имеет так же определённые недостатки.

Входные-выходные данные. Диаграмма IDEF0.

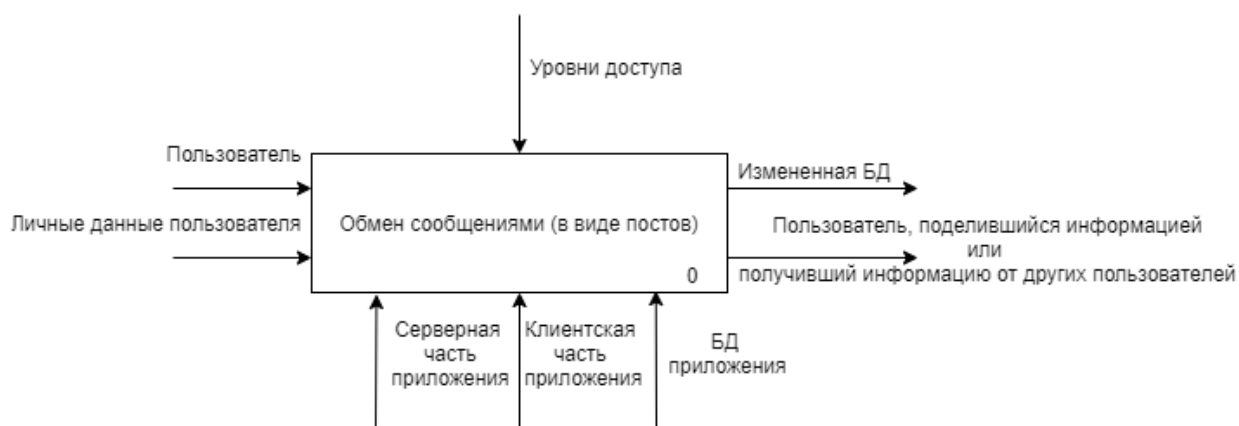


Диаграмма вариантов использования UseCase.

Неавторизованный пользователь:



Авторизованный пользователь:

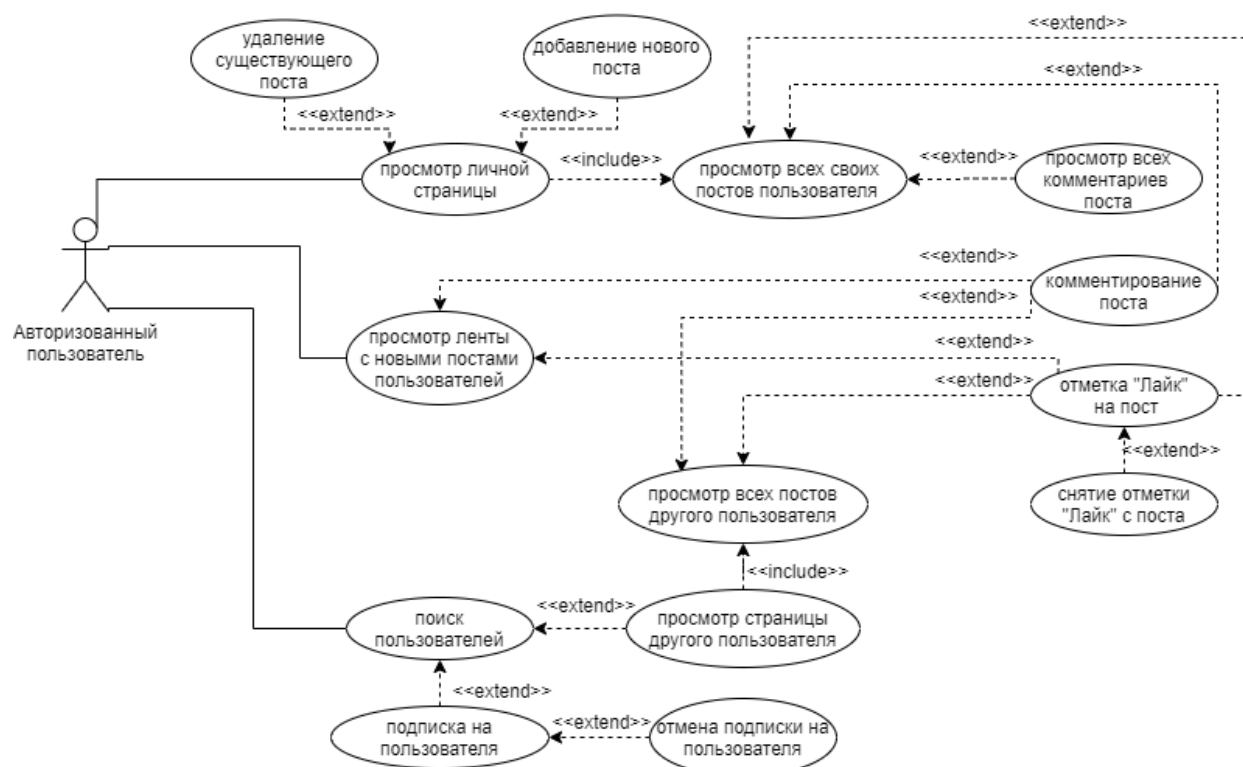
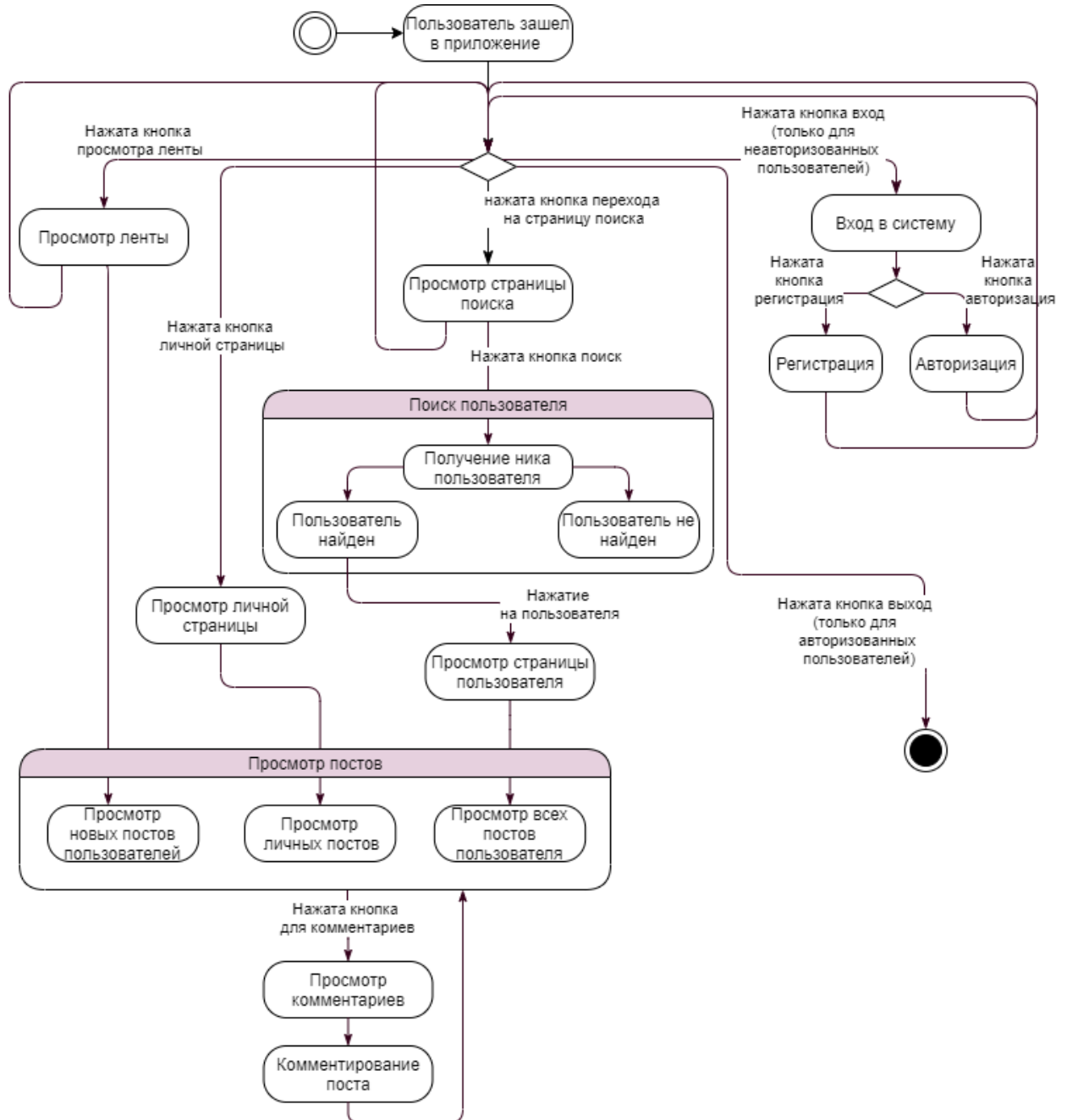


Диаграмма состояний.



Структурная схема приложения.

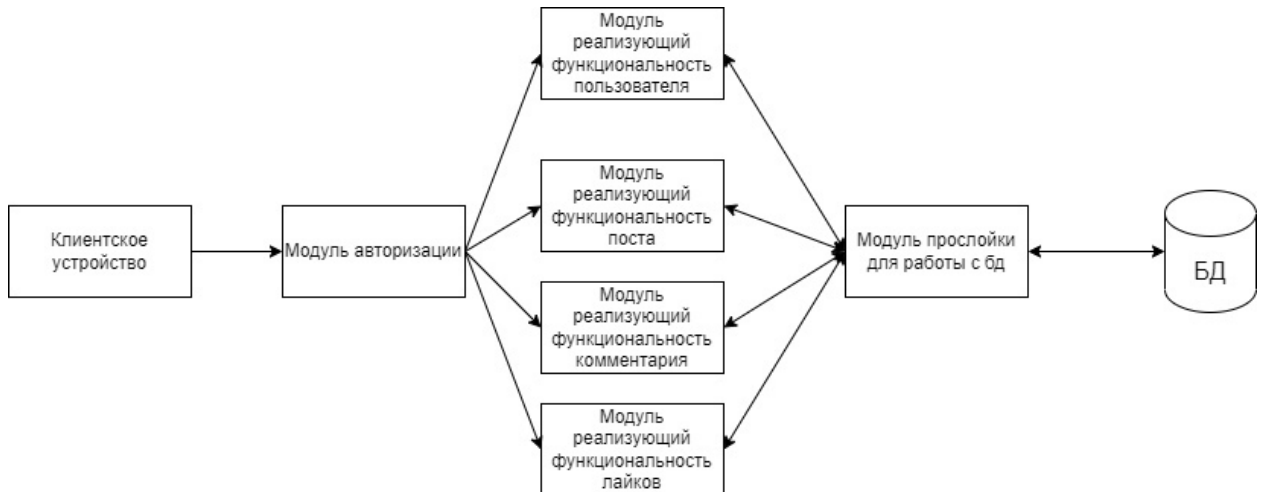
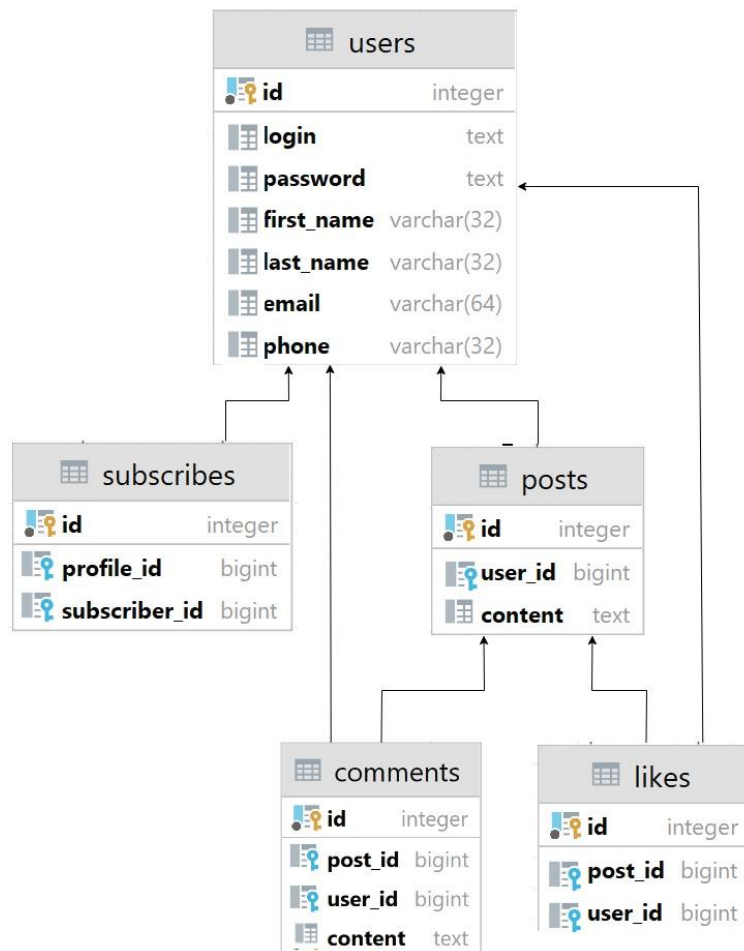


Схема базы данных.



Users - таблица, содержащая информацию о зарегистрированных пользователях приложения.

Subscribes - таблица, содержащая информацию о подписках пользователей.

Post - таблица, содержащая информацию о всех постах пользователей.

Comments - таблица, содержащая информацию о всех комментариях пользователей.

Likes - таблица, содержащая информацию о всех отметках «Лайк» пользователей.

Диаграмма потоков данных.

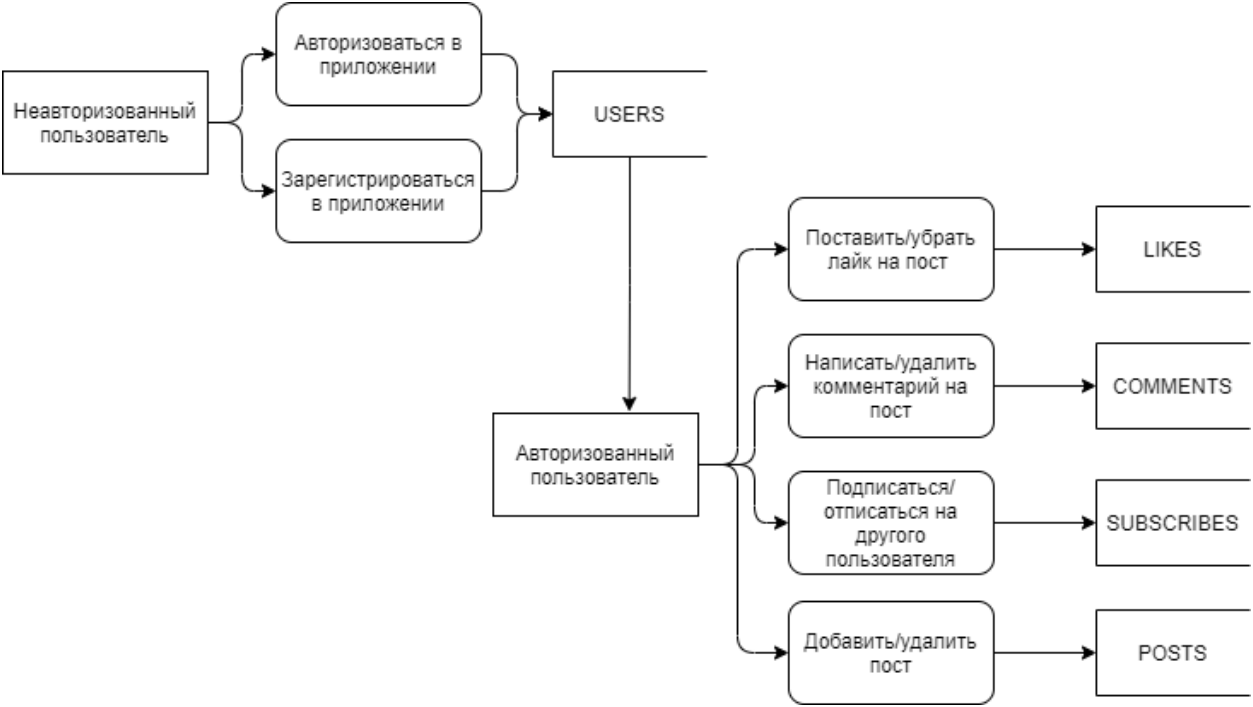


Диаграмма классов.

Диаграммы классов сервера.

Диаграмма классов библиотеки controller.

<div>LikesController</div> <div>LikesController(LikesService)</div> <div>likesService LikesService</div> <div><div>getLikesByld(Long) ResponseEntity< LikesResponse ></div><div>deleteLikes(Long) ResponseEntity<?></div><div>getAllLikes() ResponseEntity< List< LikesResponse > ></div><div>addLikes(LikesCreateRequest) ResponseEntity< LikesResponse ></div><div>updateLikes(LikesUpdateRequest) ResponseEntity< LikesResponse ></div></div>	<div>PostController</div> <div>PostController(PostService)</div> <div>postService PostService</div> <div><div>getPostByld(Long) ResponseEntity< PostResponse ></div><div>addPost(PostCreateRequest) ResponseEntity< PostResponse ></div><div>deletePost(Long) ResponseEntity<?></div><div>getAllPosts() ResponseEntity< List< PostResponse > ></div><div>updatePost(PostUpdateRequest) ResponseEntity< PostResponse ></div></div>
<div>UserController</div> <div>UserController(UserService)</div> <div>userService UserService</div> <div><div>addUser(UserCreateRequest) ResponseEntity< UserResponse ></div><div>updateUser(UserUpdateRequest) ResponseEntity< UserResponse ></div><div>getAllUsers() ResponseEntity< List< UserResponse > ></div><div>deleteUser(Long) ResponseEntity<?></div><div>getUserByld(Long) ResponseEntity< UserResponse ></div></div>	<div>CommentsController</div> <div>CommentsController(CommentsService)</div> <div>commentsService CommentsService</div> <div><div>getCommentsByld(Long) ResponseEntity< CommentsResponse ></div><div>getAllComments() ResponseEntity< List< CommentsResponse > ></div><div>updateComments(CommentsUpdateRequest) ResponseEntity< CommentsResponse ></div><div>deleteComments(Long) ResponseEntity<?></div><div>addComments(CommentsCreateRequest) ResponseEntity< CommentsResponse ></div></div>

SubscribeController	
SubscribeController (SubscribeService)	
subscribeService SubscribeService	
getAllSubscribes ()	ResponseEntity <List <SubscribeResponse > >
updateSubscribe (SubscribeUpdateRequest)	ResponseEntity <SubscribeResponse >
addSubscribe (SubscribeCreateRequest)	ResponseEntity <SubscribeResponse >
getSubscribeById (Long)	ResponseEntity <SubscribeResponse >
deleteSubscribe (Long)	ResponseEntity <? >

Диаграмма классов библиотеки dto.

Библиотека Comments:

CommentsCreateRequest	CommentsUpdateRequest	CommentsResponse
CommentsCreateRequest (String, String, String)	CommentsUpdateRequest (Long, String, String, String)	CommentsResponse ()
CommentsCreateRequest ()	CommentsUpdateRequest ()	CommentsResponse (Long, String, String, String)
content String	id Long	id Long
user_id String	content String	content String
post_id String	user_id String	post_id String
setUser_id (String) void	post_id String	user_id String
getPost_id () String	getId () Long	toString () String
getUser_id () String	getPost_id () String	getId () Long
getContent () String	getUser_id () String	getPost_id () String
setPost_id (String) void	getContent () String	getUser_id () String
toString () String	canEqual (Object) boolean	getContent () String
setContent (String) void	hashCode () int	setId (Long) void
equals (Object) boolean	toString () String	setPost_id (String) void
canEqual (Object) boolean	setid (Long) void	hashCode () int
hashCode () int	setPost_id (String) void	setUser_id (String) void
	setUser_id (String) void	setContent (String) void
	setContent (String) void	equals (Object) boolean
	equals (Object) boolean	canEqual (Object) boolean

Библиотека Likes:

LikesUpdateRequest	LikesResponse	LikesCreateRequest
LikesUpdateRequest (Long, String, String)	LikesResponse (Long, String, String)	LikesCreateRequest (String, String)
LikesUpdateRequest ()	LikesResponse ()	LikesCreateRequest ()
user_id String	post_id String	post_id String
id Long	user_id String	user_id String
post_id String	id Long	
getPost_id () String	setPost_id (String) void	getPost_id () String
getId () Long	getId () Long	getUser_id () String
getUser_id () String	getPost_id () String	setPost_id (String) void
setId (Long) void	getUser_id () String	setUser_id (String) void
setPost_id (String) void	toString () String	equals (Object) boolean
setUser_id (String) void	setid (Long) void	canEqual (Object) boolean
equals (Object) boolean	setUser_id (String) void	hashCode () int
canEqual (Object) boolean	equals (Object) boolean	toString () String
hashCode () int	canEqual (Object) boolean	
toString () String	hashCode () int	

Библиотека Post:

PostUpdateRequest	PostResponse	PostCreateRequest
PostUpdateRequest(Long, Long, String) PostUpdateRequest()	PostResponse() PostResponse(Long, Long, String)	PostCreateRequest(Long, String) PostCreateRequest()
userId Long content String id Long	userId Long content String id Long	userId Long content String
getUserId() Long getId() Long getContent() String setId(Long) void toString() String setUserId(Long) void setContent(String) void equals(Object) boolean canEqual(Object) boolean hashCode() int	getId() Long getUserId() Long toString() String getContent() String setId(Long) void setUserId(Long) void setContent(String) void equals(Object) boolean canEqual(Object) boolean hashCode() int	getUserId() Long getContent() String setUserId(Long) void setContent(String) void equals(Object) boolean canEqual(Object) boolean hashCode() int toString() String

Библиотека Subscribe:

SubscribeResponse	SubscribeUpdateRequest	SubscribeCreateRequest
SubscribeResponse() SubscribeResponse(Long, Long, String)	SubscribeUpdateRequest(Long, Long, String) SubscribeUpdateRequest()	SubscribeCreateRequest(Long, String) SubscribeCreateRequest()
id Long profileId Long subscriberId String	profileId Long subscriberId String id Long	profileId Long subscriberId String
getId() Long getProfileId() Long getSubscriberId() String setId(Long) void toString() String setProfileId(Long) void setSubscriberId(String) void equals(Object) boolean canEqual(Object) boolean hashCode() int	getProfileId() Long getId() Long getSubscriberId() String setId(Long) void setProfileId(Long) void setSubscriberId(String) void equals(Object) boolean toString() String canEqual(Object) boolean hashCode() int	getProfileId() Long getSubscriberId() String setProfileId(Long) void setSubscriberId(String) void equals(Object) boolean canEqual(Object) boolean hashCode() int toString() String

Библиотека Users:

UserUpdateRequest	UserCreateRequest	UserResponse
UserUpdateRequest(Long, String, String, String, String, String, String) UserUpdateRequest()	UserCreateRequest() UserCreateRequest(String, String, String, String, String, String, String)	UserResponse() UserResponse(Long, String, String, String, String, String, String)
firstName String email String phone String login String password String id Long lastName String	phone String lastName String email String password String login String firstName String	login String email String id Long lastName String phone String firstName String
getId() Long getLogin() String equals(Object) boolean getPassword() String setPhone(String) void setLogin(String) void setLastName(String) void getFirstName() String setPassword(String) void toString() String getLastName() String canEqual(Object) boolean hashCode() int setEmail(String) void setFirstName(String) void getEmail() String getPhone() String setId(Long) void	getLogin() String getPassword() String getFirstName() String canEqual(Object) boolean setEmail(String) void getLastName() String setPhone(String) void hashCode() int getEmail() String getPhone() String toString() String setLogin(String) void equals(Object) boolean setPassword(String) void setFirstName(String) void setLastName(String) void	getId() Long getLogin() String getFirstName() String setPhone(String) void getLastName() String hashCode() int getEmail() String setEmail(String) void getPhone() String setId(Long) void setLastName(String) void canEqual(Object) boolean setLogin(String) void toString() String equals(Object) boolean setFirstName(String) void

Диаграмма классов библиотеки entity.

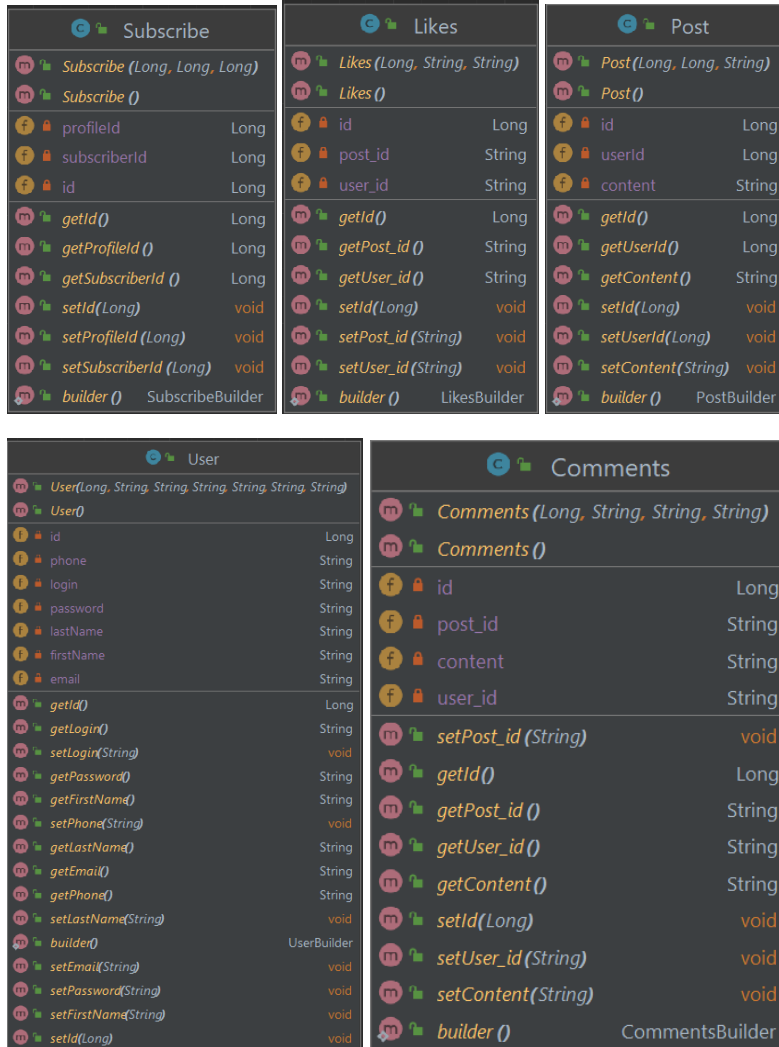


Диаграмма классов библиотеки mapper.

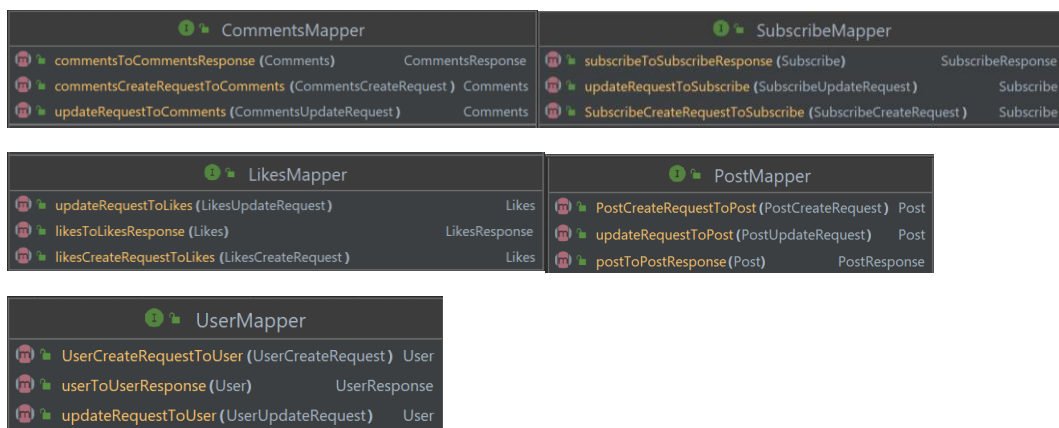


Диаграмма классов библиотеки repository.

CommentsRepository
SubscribeRepository
PostRepository
LikesRepository
UserRepository

Диаграммы классов клиента.

Диаграмма главного класса Main.







	MainActivity	
	MainActivity ()	
	binding	ActivityMainBinding
	onCreate(Bundle)	void
	onCreateOptionsMenu (Menu)	boolean
	onOptionsItemSelected (MenuItem)	boolean

Диаграмма классов библиотеки ui.

Login:

NewUserLoginFragment	LoginActivity
NewUserLoginFragment ()	LoginActivity ()
phone	EditText
surname	EditText
name	EditText
exitButton	ImageButton
password	EditText
email	EditText
login	EditText
registerButton	Button
onCreate(Bundle)	void
loginButton	Button
login	EditText
password	EditText
registerButton	Button
exitButton	ImageButton
loginByEmailButton	Button
onCreate(Bundle)	void

Post:

PostPrivateAdapter	PostAdapter
PostPrivateAdapter(OnCommentClickListener, OnLikeClickListener)	PostAdapter(OnCommentClickListener, OnLikeClickListener)
tweetList	List<Post>
TWITTER_RESPONSE_FORMAT	String
onLikeClickListener	OnLikeClickListener
MONTH_DAY_FORMAT	String
onCommentClickListener	OnCommentClickListener
onDeleteClickListener	onDeleteClickListener
onCreateViewHolder(ViewGroup, int)	PostViewHolder
getItemCount()	int
setItems(Collection<Post>)	void
onBindViewHolder(PostViewHolder, int)	void
clearItems()	void
clearItems()	void
onBindViewHolder(PostViewHolder, int)	void
getItemCount()	int
setItems(Collection<Post>)	void
onCreateViewHolder(ViewGroup, int)	PostViewHolder

CommentListFragment	CommentListViewModel
CommentListFragment () commentAdapter CommentAdapter commentText EditText commentsRecyclerView RecyclerView nameTextView TextView MONTH_DAY_FORMAT String TWITTER_RESPONSE_FORMAT String commentButton Button exitButton ImageButton creationDateTextView TextView contentTextView TextView nickTextView TextView comments Collection <Comment> POST_ID String loadComments (Long) void initRecyclerView () void getFormattedDate (String) String onCreate (Bundle) void loadPostInfo (Long) void	CommentListViewModel () TWITTER_RESPONSE_FORMAT String nameText MutableLiveData <String> MONTH_DAY_FORMAT String commentAdapterData MutableLiveData <CommentAdapter> comments Collection <Comment> nickText MutableLiveData <String> creationDateTextView MutableLiveData <String> contentText MutableLiveData <String> commentAdapter CommentAdapter setData (Long) void getNickText () MutableLiveData <String> _getCommentAdapterData () MutableLiveData <CommentAdapter> loadPostInfo (Long) void loadComments (Long) void getContentText () MutableLiveData <String> getCommentAdapter () CommentAdapter getFormattedDate (String) String getCreationDateTextView () MutableLiveData <String> getNameText () MutableLiveData <String>

CommentAdapter	AddPostActivity
CommentAdapter () TWITTER_RESPONSE_FORMAT String MONTH_DAY_FORMAT String commentList List<Comment> onBindViewHolder (CommentViewHolder, int) void setItems (Collection<Comment>) void onCreateViewHolder (ViewGroup, int) CommentViewHolder getItemCount () int clearItems () void	AddPostActivity () exitButton ImageButton USER_ID String textPost EditText addPostButton Button onCreate (Bundle) void

Search:

UserSearchViewModel	UserSearchFragment
UserSearchViewModel () mText MutableLiveData <String> getText () LiveData <String>	UserSearchFragment () binding FragmentUserSearchBinding onCreateView (LayoutInflater, ViewGroup, Bundle) View onDestroyView () void

User_info:

UserInfoFragment	UserInfoViewModel
UserInfoFragment () followersCountTextView TextView postsRecyclerView RecyclerView postPrivateAdapter PostPrivateAdapter addPostButton Button nickTextView TextView descriptionTextView TextView user User binding FragmentUserInfoBinding followingCountTextView TextView onCreateView (LayoutInflater, ViewGroup, Bundle) View getPostAdapter () PostPrivateAdapter	UserInfoViewModel () user User userDescriptionText MutableLiveData <String> userNickText MutableLiveData <String> userFollowingText MutableLiveData <String> userFollowersText MutableLiveData <String> getUserFollowingText () MutableLiveData <String> loadUserInfo () void getUserNickText () MutableLiveData <String> getUserDescriptionText () MutableLiveData <String> getUserFollowersText () MutableLiveData <String>

Wall:

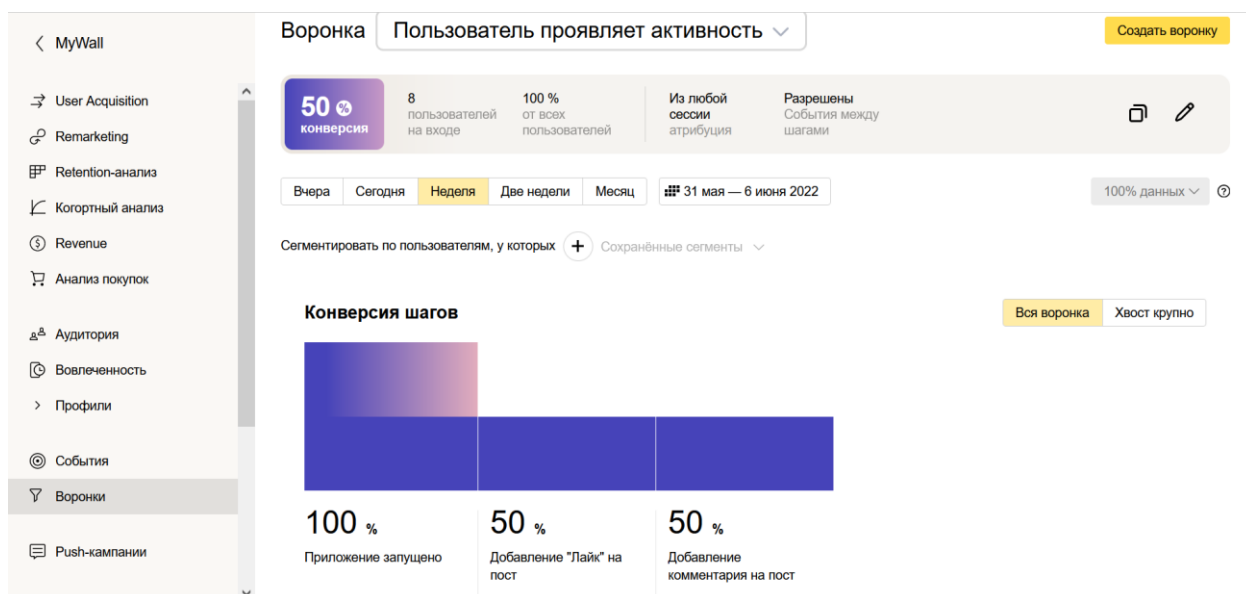
WallViewModel	WallFragment
WallViewModel () mText MutableLiveData <String> getText () LiveData <String>	WallFragment () binding FragmentWallBinding postAdapter PostAdapter postsRecyclerView RecyclerView getPostAdapter () PostAdapter onCreateView (LayoutInflater, ViewGroup, Bundle) View

Сценарии воронок.

В аналитической системе appmetrica.yandex.ru созданы сценарии воронок.

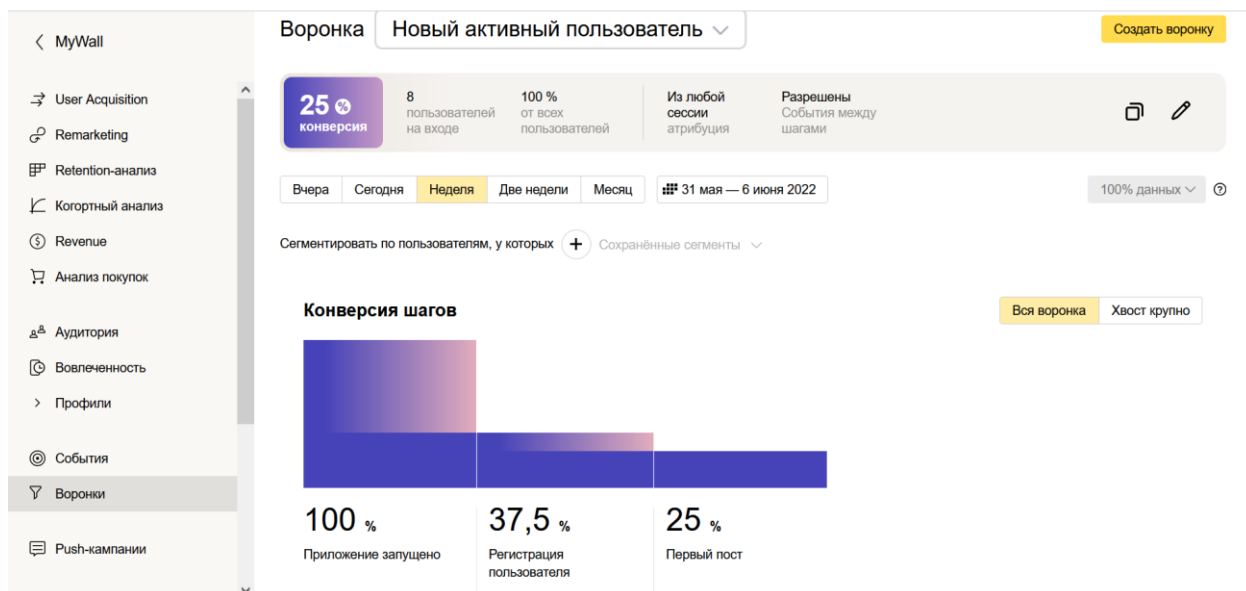
Воронка «Пользователь проявляет активность».

Расчет метрики ведется по пользователям. Необходимо проанализировать количество пользователей, которые заинтересованы в информации приложения, представленной в виде постов, и после запуска приложения проявляют активность.



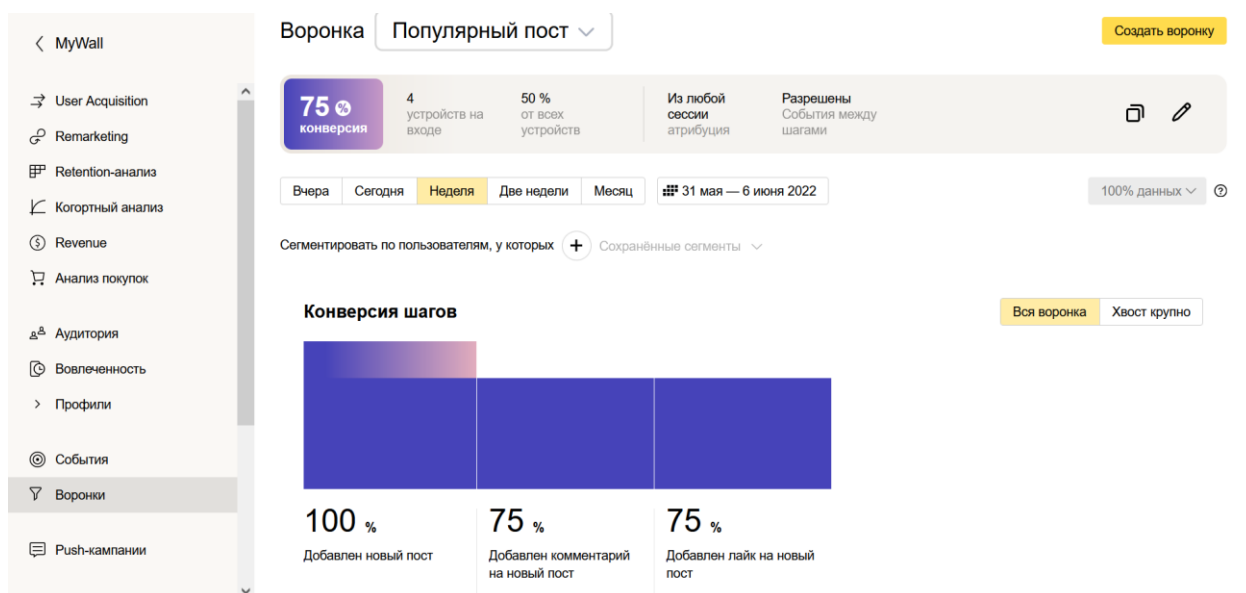
Воронка «Новый активный пользователь».

Расчет метрики ведется по пользователям. Необходимо проанализировать процент новых «живых» пользователей, которые после регистрации аккаунта выкладывают новую информацию в виде постов.



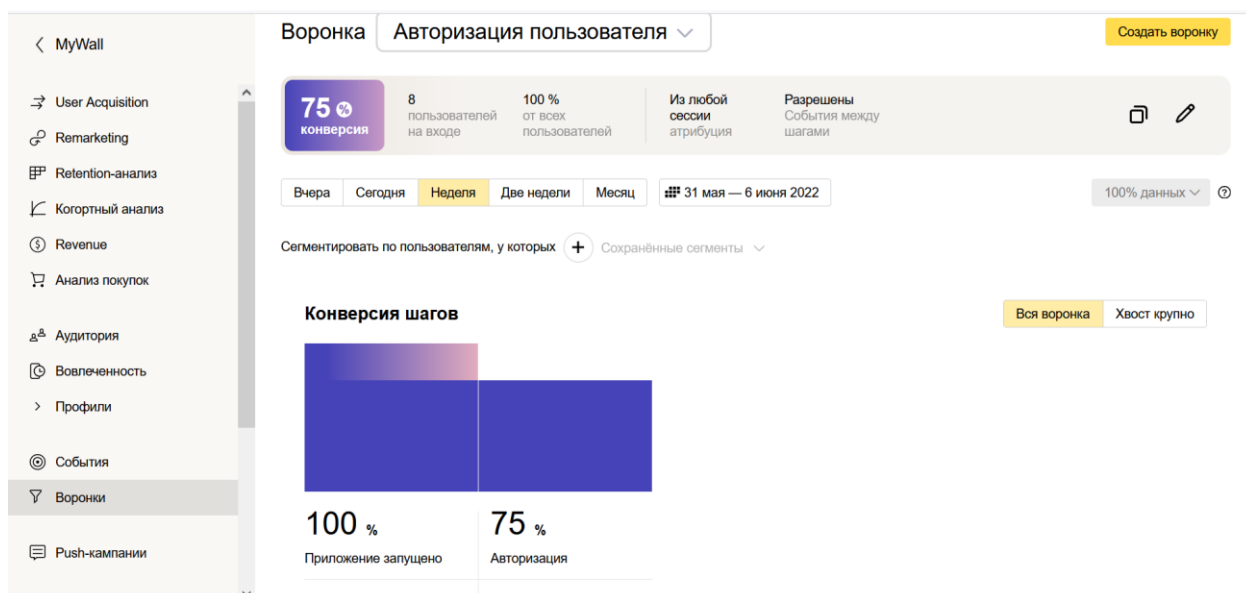
Воронка «Популярный пост».

Расчет метрики ведется по устройствам. Необходимо проанализировать заинтересованность пользователей в новой поступающей информации в виде постов.



Воронка «Авторизация пользователя».

Расчет метрики ведется по пользователям. Необходимо проанализировать сколько пользователей проходят авторизацию с новых устройств.



Используемая платформа.

- ОС Windows 10;
- Spring Framework
- База данных – PostgreSQL;
- Язык разработки – Java;
- Используемые IDE - IntelliJ IDEA 2019.2.2, Android Studio;
- Система контроля версий – GitHub;
- Хостинг firstvds.ru;
- Диаграммы и макеты интерфейсов draw.io.

Тестирование.

Тестирование приложения выполнялось по методу черного ящика. Для тестирования были составлены следующие тест-кейсы.

Номер	1
Заголовок	Авторизация в приложении уже зарегистрированного пользователя
Предусловие	Приложение запущено
Шаг	Ожидаемый результат
Нажать кнопку «авторизоваться» на личной странице пользователя	Открылась форма ввода логина и пароля

Ввести существующие логин и пароль в соответствующие области ввода	Введенный логин отображается в области ввода, введенный пароль отображается в области ввода скрытым
Нажать кнопку «войти»	Форма авторизации закрылась, открылась личная страница пользователя с загруженными данными

Номер	2
Заголовок	Авторизация в приложении с неправильным логином или паролем
Предусловие	Приложение запущено
Шаг	Ожидаемый результат
Нажать кнопку «авторизоваться» на личной странице пользователя	Открылась форма ввода логина и пароля
Ввести несуществующий логин или пароль в соответствующую область ввода	Введенный логин отображается в области ввода, введенный пароль отображается в области ввода скрытым
Нажать кнопку «войти»	Всплывающее сообщение с текстом «Ошибка авторизации. Проверьте входные данные»

Номер	3
Заголовок	Регистрация в приложении
Предусловие	Приложение запущено
Шаг	Ожидаемый результат
Нажать кнопку «авторизоваться» на личной странице пользователя	Открылась форма ввода логина и пароля
Нажать кнопку «регистрация»	Открылась форма регистрации
Заполнить не все текстовые поля	Всплывающее сообщение «Пожалуйста, заполните все поля»
Заполнить все текстовые поля	Все вводимы отображаются в соответствующих полях
Нажать кнопку «регистрация»	Форма регистрации закрылась. Открылась личная страница пользователя. В ленте постов надпись «добавьте первый пост»

Номер	4
Заголовок	Добавление поста
Предусловие	Приложение запущено, пользователь авторизован. Открыта личная страница пользователя
Шаг	Ожидаемый результат
Нажать кнопку «добавить пост» на личной странице пользователя	Открылась форма добавления поста
Заполнить текстовое поле поста	Вводимая информация отображается в текстовом поле
Нажать кнопку «добавить»	Форма добавления поста закрылась, на личной странице отображается новый пост

Номер	5
Заголовок	Добавление/удаление отметки «лайк»
Предусловие	Приложение запущено, пользователь авторизован, открыта лента постов
Шаг	Ожидаемый результат
Нажать кнопку «лайк» на пост, где не установлен лайк	Кнопка «лайк» закрашена, количество лайков увеличилось на единицу
Повторить нажатие на эту же кнопку «лайк»	Кнопка вернулась в первоначальное состояние, количество лайков уменьшилось на единицу

Номер	6
Заголовок	Добавление комментария на пост
Предусловие	Приложение запущено, пользователь авторизован, открыта лента постов
Шаг	Ожидаемый результат
Нажать кнопку «комментарий» на пост	Открылось окно со списком комментариев к посту
Ввести комментарий в поле ввода комментария	Вводимый текст отображается в поле ввода
Нажать кнопку «добавить»	Комментарий отображается в списке комментариев
Нажать кнопку закрытия окна	Открылась страница, на которой находился пост. Количество комментариев под постом увеличилось на единицу

Номер	7
Заголовок	Удаление поста
Предусловие	Приложение запущено, пользователь авторизован, открыта личная страница, на которой есть пост
Шаг	Ожидаемый результат
Нажать кнопку удаления поста	Выбранный пост больше не отображается на личной странице

Номер	8
Заголовок	Поиск пользователей
Предусловие	Приложение запущено, открыта страница поиска
Шаг	Ожидаемый результат
Ввести логин в поле ввода	Вводимые данные отображаются в поле ввода
Нажать кнопку «поиск»	Если пользователи не найдены, появляется надпись «пользователи не найдены». Если пользователи найдены, отображается список пользователей

Номер	9
Заголовок	Подписка /отписка на пользователя
Предусловие	Приложение запущено, пользователь авторизован, открыта страница поиска, отображается список найденных пользователей
Шаг	Ожидаемый результат
Нажать кнопку «подписаться» у пользователя, ранее не добавленного в подписки	Кнопка «подписаться» изменилась на «отписаться». Количество читателей пользователя увеличилось на единицу
Перейти на страницу ленты постов	В ленте отображаются посты подписки в случае их наличия
Перейти на личную страницу	Количество подписок (читает) увеличилось на единицу
Перейти на страницу поиска и повторить нажатие на кнопку	Кнопка изменилась на «подписаться», количество читателей пользователя изменилось на единицу

Номер	10
Заголовок	Google-авторизация
Предусловие	Приложение запущено, открыта форма авторизации
Шаг	Ожидаемый результат
Нажать кнопку «Войти через Google»	Открывается форма Google-входа
Ввести необходимые данные для входа в систему Google и войти в систему	Открывается личная страница пользователя с заполненными данными на основе Google-профиля

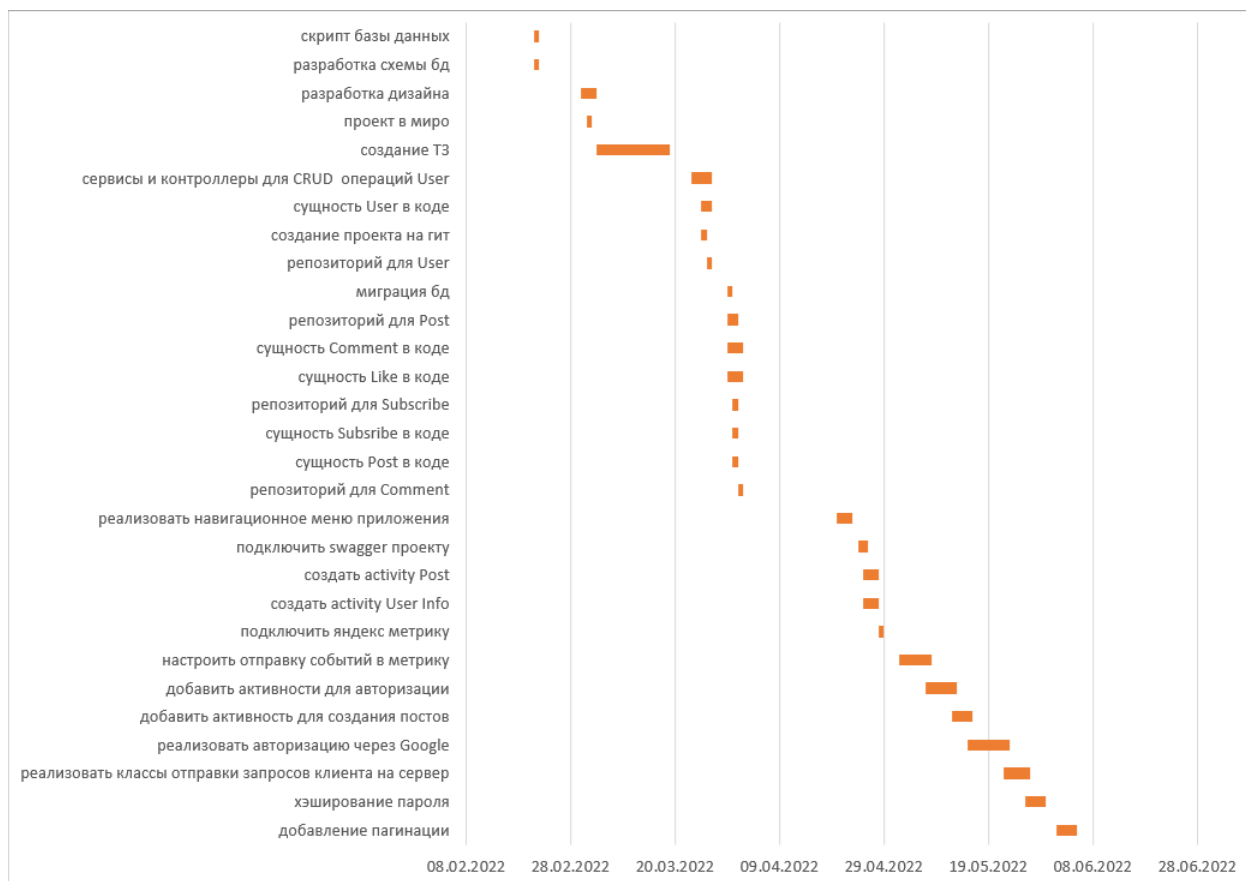
Список планируемых работ.

1. Скрипт базы данных.
2. Разработка схемы Базы данных.
3. Разработка дизайна.
4. Проект в Miro.
5. Создание Технического задания.
6. Сервисы и контроллеры для crud операций user.
7. Сущность user в коде.
8. Создание проекта на GitHub.
9. Репозиторий для user.
10. Миграция бд.
11. Репозиторий для post.
12. Сущность like в коде.
13. Сущность comment в коде.
14. Репозиторий для subscribe.
15. Сущность subsribe в коде.
16. Сущность post в коде.
17. Репозиторий для comment.
18. Реализовать навигационное меню приложения.
19. Подключить swagger проекту.
20. Создать activity post.

21. Создать activity user info.
22. Подключить яндекс метрику.
23. Настроить отправку событий в метрику.
24. Добавить активности для авторизации.
25. Добавить активность для создания постов.
26. Реализовать авторизацию через Google.
27. Реализовать классы отправки запросов клиента на сервер.
28. Хэширование пароля.
29. Добавление пагинации.

Календарный план.

задача	Дата начала	Дата завершения	Длительность
добавление пагинации	01.06.2022	05.06.2022	4
хэширование пароля	26.05.2022	29.05.2022	4
реализовать классы отправки запросов клиента на сервер	22.05.2022	26.05.2022	5
реализовать авторизацию через Google	15.05.2022	22.05.2022	8
добавить активность для создания постов	12.05.2022	15.05.2022	4
добавить активности для авторизации	07.05.2022	12.05.2022	6
настроить отправку событий в метрику	02.05.2022	07.05.2022	6
подключить яндекс метрику	28.04.2022	28.04.2022	1
создать activity User Info	25.04.2022	27.04.2022	3
создать activity Post	25.04.2022	27.04.2022	3
подключить swagger проект	24.04.2022	25.04.2022	2
реализовать навигационное меню приложения	20.04.2022	22.04.2022	3
репозиторий для Comment	01.04.2022	01.04.2022	1
сущность Post в коде	31.03.2022	31.03.2022	1
сущность Subscribe в коде	31.03.2022	31.03.2022	1
репозиторий для Subscribe	31.03.2022	31.03.2022	1
сущность Like в коде	30.03.2022	01.04.2022	3
сущность Comment в коде	30.03.2022	01.04.2022	3
репозиторий для Post	30.03.2022	31.03.2022	2
миграция бд	30.03.2022	30.03.2022	1
репозиторий для User	26.03.2022	26.03.2022	1
создание проекта на гит	25.03.2022	25.03.2022	1
сущность User в коде	25.03.2022	26.03.2022	2
сервисы и контроллеры для CRUD операций User	23.03.2022	26.03.2022	4
создание ТЗ	05.03.2022	18.03.2022	14
проект в мире	03.03.2022	03.03.2022	1
разработка дизайна	02.03.2022	04.03.2022	3
разработка схемы бд	21.02.2022	21.02.2022	1
скрипт базы данных	21.02.2022	21.02.2022	1



Отзывы пользователей.

Очень понравилось приложение, простой и понятный интерфейс, зашел с первого раза, сразу начал выкладывать посты!

04.06.22 Иванков
команда 6.1-2

Очередное пополнение в моей копилке приложений, и оно приятно удивило. Никакой надоедливой рекламы и ботов. Рекомендую к использованию.

04.06.22 Иванов А.Р.
команда 7.3.

Очень яркий дизайн привлек сразу внимание. С регистрацией проблем не возникло. Тестировали Google авторизацию – упростило вход и сделало его моментальным. Обязательно будем пользоваться приложением.

01.06.22 Богословский
команда 6.2-3

Хорошее приложение. Удобный интерфейс. Зарегистрировался и пробовал добавлять посты. Поиск пользователей помогает быстро найти интересных пользователей. Интересно пользоваться приложением.

04.06.22 команда №2
Фамилия / И.С.Смирнов

Зашел в приложение и сразу же захотелось зарегистрироваться. Удобно наличие ночной темы. Понравился простой интерфейс. Жду, когда у меня появятся первые подписчики!

Симонков Сергей
4.06.2022

Заключение

Для подготовки к разработке проекта был проведен детальный анализ предметной области, составлены UML диаграммы, подготовлены планы работ. Для контроля версий использовалась платформа GitHub. Контроль за ходом выполнения работ осуществлялся на платформе Jira.

В приложении реализованы сценарии для двух типов пользователя – авторизованного и неавторизованного. Также были разработаны дизайн и макеты интерфейса.

В ходе работы над проектом были разработаны клиентская и серверная части в соответствии с техническим заданием. Серверная часть приложения и база данных размещены на хостинге.

Для бизнес-аналитики приложения реализованы сценарии воронок и получены статистические данные.

В качестве дальнейшего развития проекта можно рассматривать:

- Возможность публикации фотографий;
- Просмотр личных страниц других пользователей;
- Личную переписку между пользователями.

В заключение следует отметить, что все поставленные задачи были успешно реализованы, для приложения были получены положительные отзывы пользователей.