

МИНОБРНАУКИ РОССИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»  
(ФГБОУ ВО «ВГУ»)

Факультет Компьютерных наук  
Кафедра программирования и информационных технологий

Отчет по курсу  
«Технологии программирования»

«MyWall»  
Аналог Твиттера

Исполнители

\_\_\_\_\_ *Фуфаева А.В.*  
\_\_\_\_\_ *Ларионов М.С.*  
\_\_\_\_\_ *Болдырев А.Д.*

Заказчик

\_\_\_\_\_ *Тарасов В.С.*

Воронеж 2022

# Оглавление

<b>Анализ предметной области.....</b>	<b>3</b>
<b>Постановка задачи.....</b>	<b>4</b>
Функциональные требования.....	4
Технические требования.....	5
Требования к интерфейсу.....	5
Макеты интерфейса. ....	5
<b>Обзор аналогов. ....</b>	<b>10</b>
Twitter. ....	10
Функциональность. ....	10
Интерфейс. ....	11
Достоинства и недостатки. ....	14
Вывод.....	15
<b>Входные-выходные данные. Диаграмма IDEF0.....</b>	<b>15</b>
<b>Диаграмма вариантов использования UseCase.....</b>	<b>15</b>
<b>Диаграмма состояний.....</b>	<b>17</b>
<b>Структурная схема приложения. ....</b>	<b>18</b>
<b>Схема базы данных. ....</b>	<b>18</b>
<b>Диаграмма потоков данных.....</b>	<b>19</b>
<b>Диаграмма классов. ....</b>	<b>19</b>
Диаграммы классов сервера.....	19
Диаграмма классов библиотеки controller. ....	19
Диаграмма классов библиотеки dto.....	20
Диаграмма классов библиотеки entity.....	22
Диаграмма классов библиотеки mapper.....	23
Диаграмма классов библиотеки repository.....	23
Диаграммы классов клиента. ....	24
Диаграмма главного класса Main. ....	24
Диаграмма классов библиотеки ui.....	24
<b>Сценарии воронок.....</b>	<b>26</b>
Воронка «Новый активный пользователь».....	27
Воронка «Популярный пост».....	27
Воронка «Авторизация пользователя».....	28
<b>Используемая платформа.....</b>	<b>28</b>
<b>Список планируемых работ. ....</b>	<b>28</b>
<b>Календарный план.....</b>	<b>29</b>

## **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:

Авторизация

Пользователь:

введите логин

Пароль:

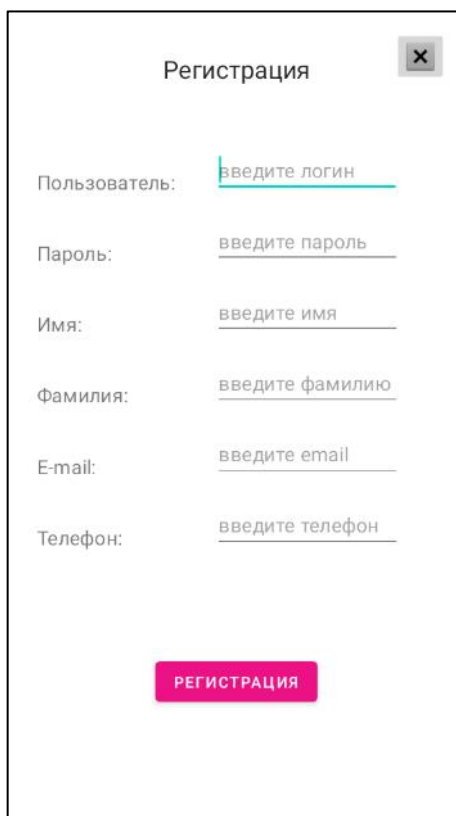
введите пароль

ВОЙТИ

РЕГИСТРАЦИЯ

ВХОД GMAIL

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

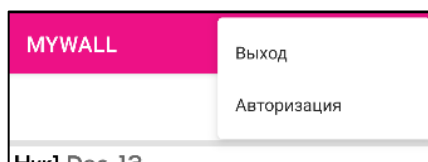


The image shows a registration form titled "Регистрация" (Registration) with a close button (X) in the top right corner. The form contains several input fields with placeholder text: "Пользователь:" (User) with "введите логин" (enter login), "Пароль:" (Password) with "введите пароль" (enter password), "Имя:" (Name) with "введите имя" (enter name), "Фамилия:" (Surname) with "введите фамилию" (enter surname), "E-mail:" with "введите email" (enter email), and "Телефон:" (Phone) with "введите телефон" (enter phone). At the bottom of the form is a pink button labeled "РЕГИСТРАЦИЯ" (Registration).

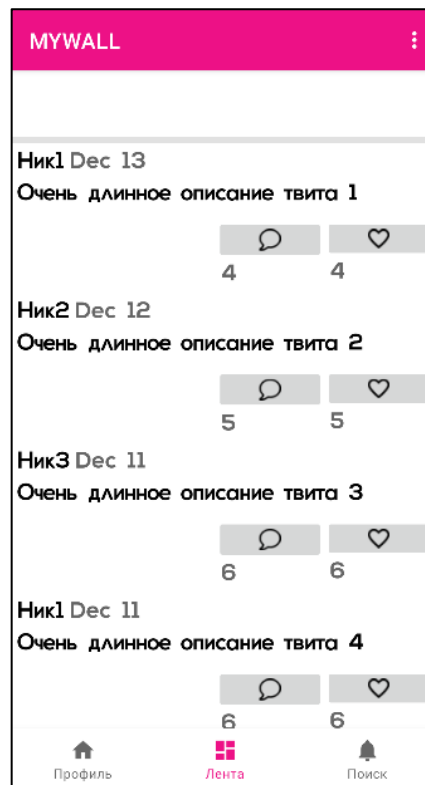
Для перехода между основными страницами приложения используется навигационное меню в нижней части экрана:



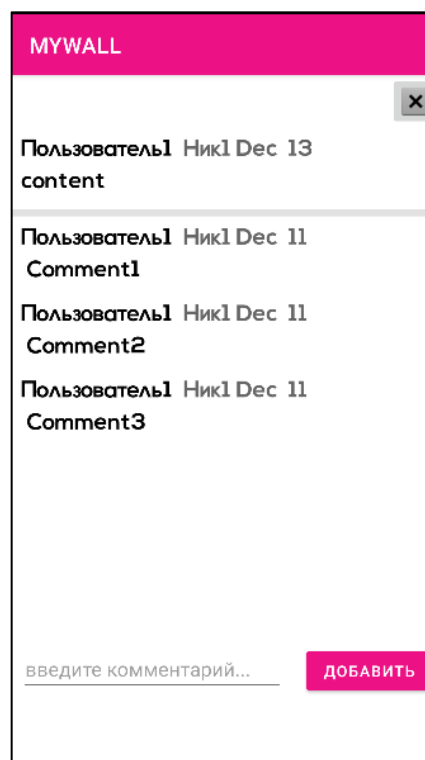
Для выхода из приложения необходимо нажать на кнопку с троеточием в верхнем меню. И выбрать опцию выход:



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:

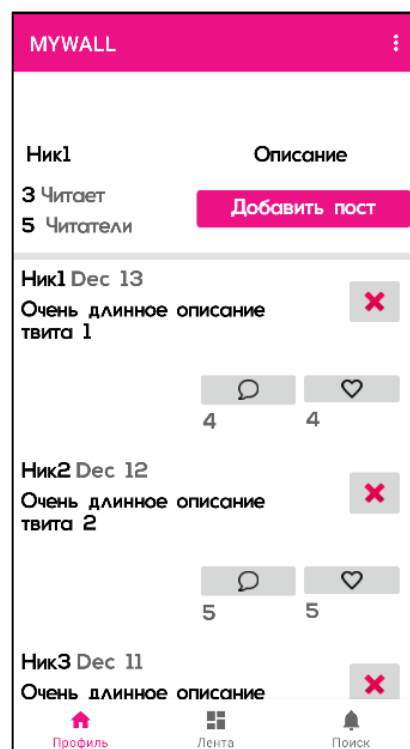




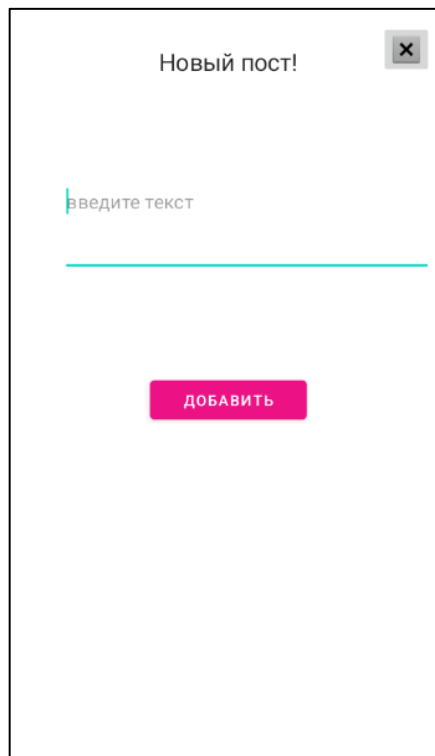
Страница поиска доступна для авторизованного и неавторизованного пользователя:



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 функционал. 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

annaufuaeva01@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

@annaufuaeva01, @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-поп

Поп-музыка

Рок

Хип-хоп/Рэп

Новости музыки

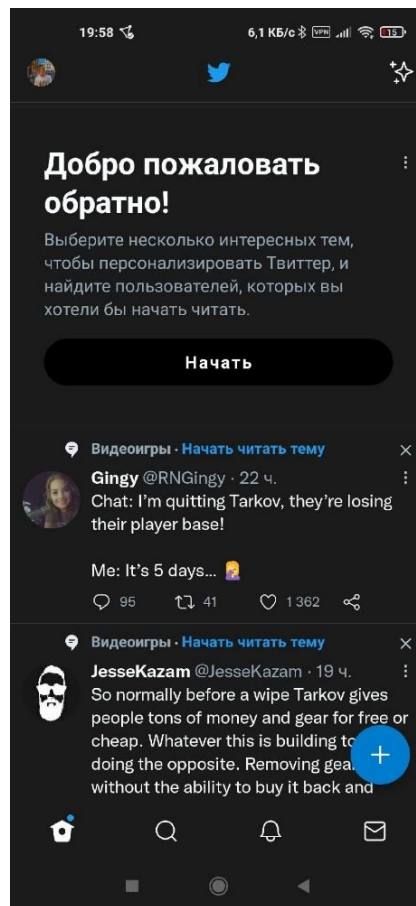
Музыка

Пропустить

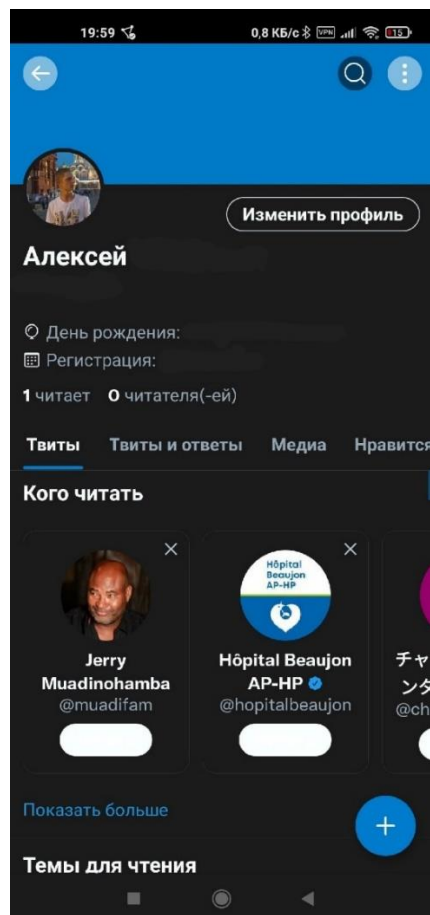
Далее

◀ ○ □

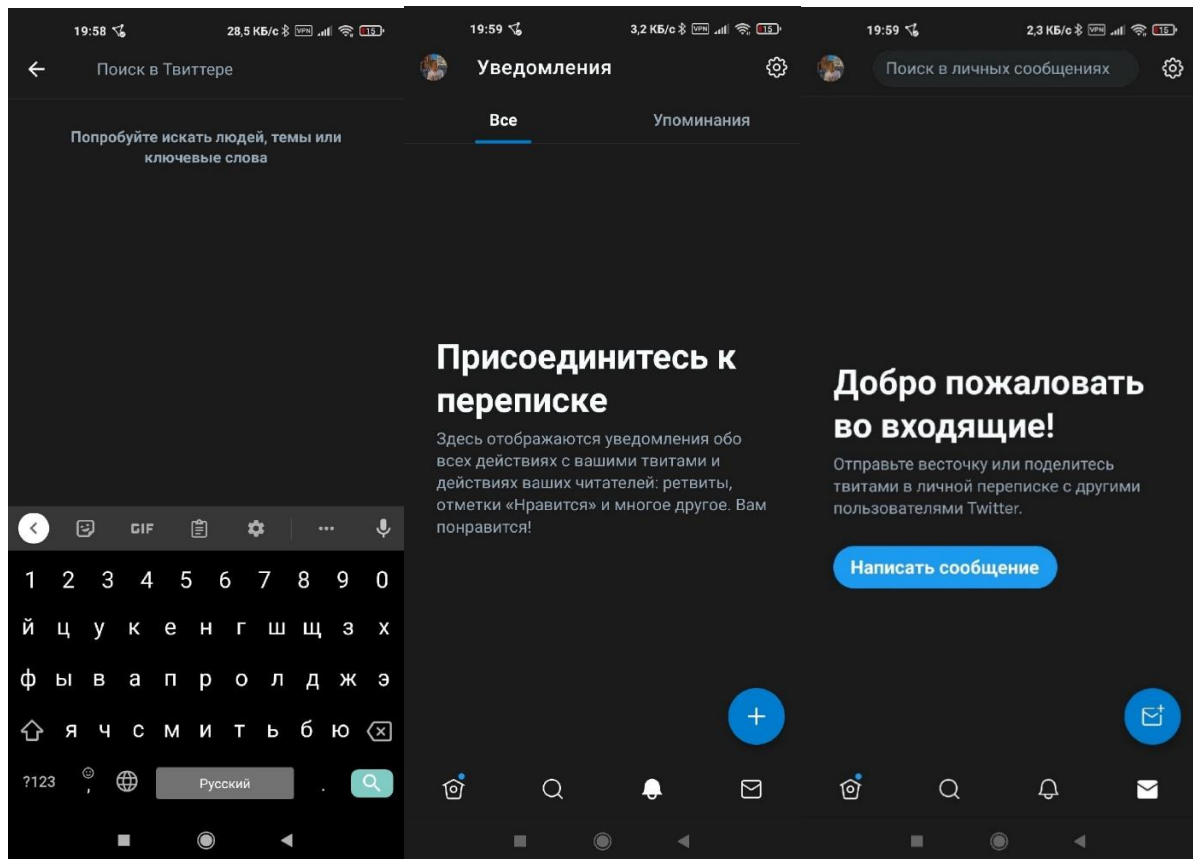
При повторном входе открывается приветственное окно:



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:

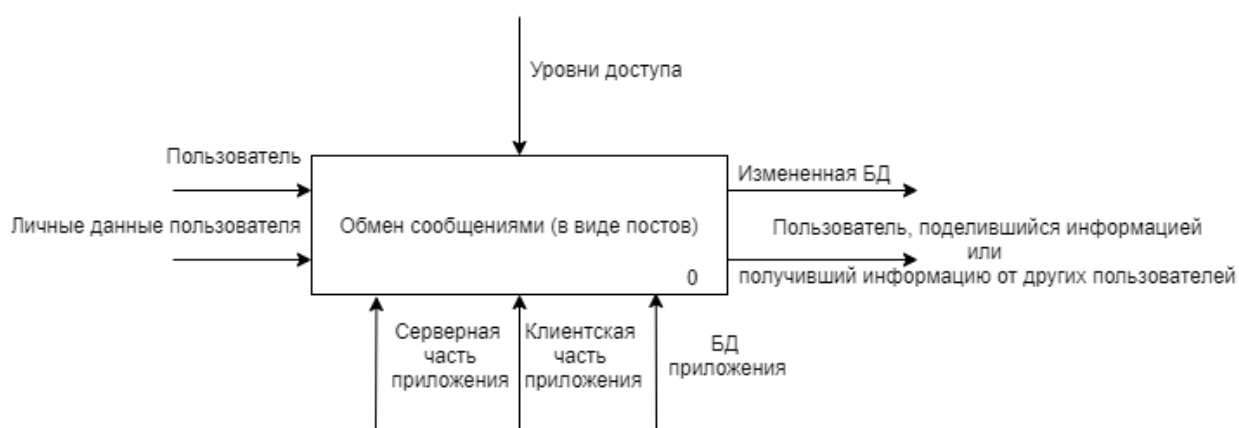
- доступно в России только через VPN (с недавнего времени);

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

Вывод.

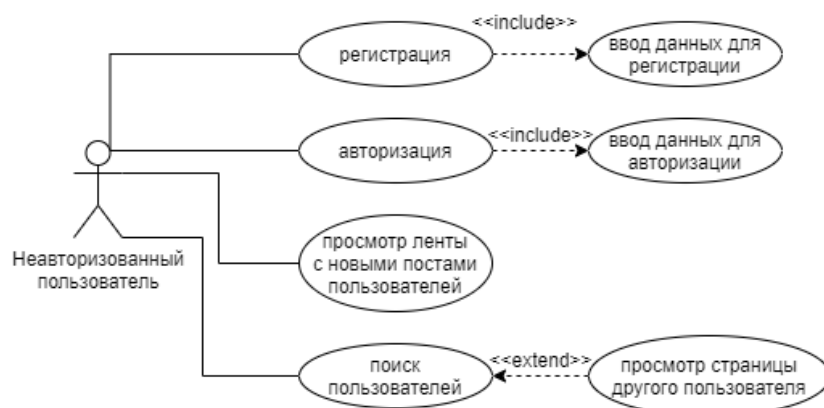
ПО имеет большой функционал, красивый, понятный и удобный интерфейс, и другие плюсы, но имеет так же определённые недостатки.

## Входные-выходные данные. Диаграмма IDEF0.

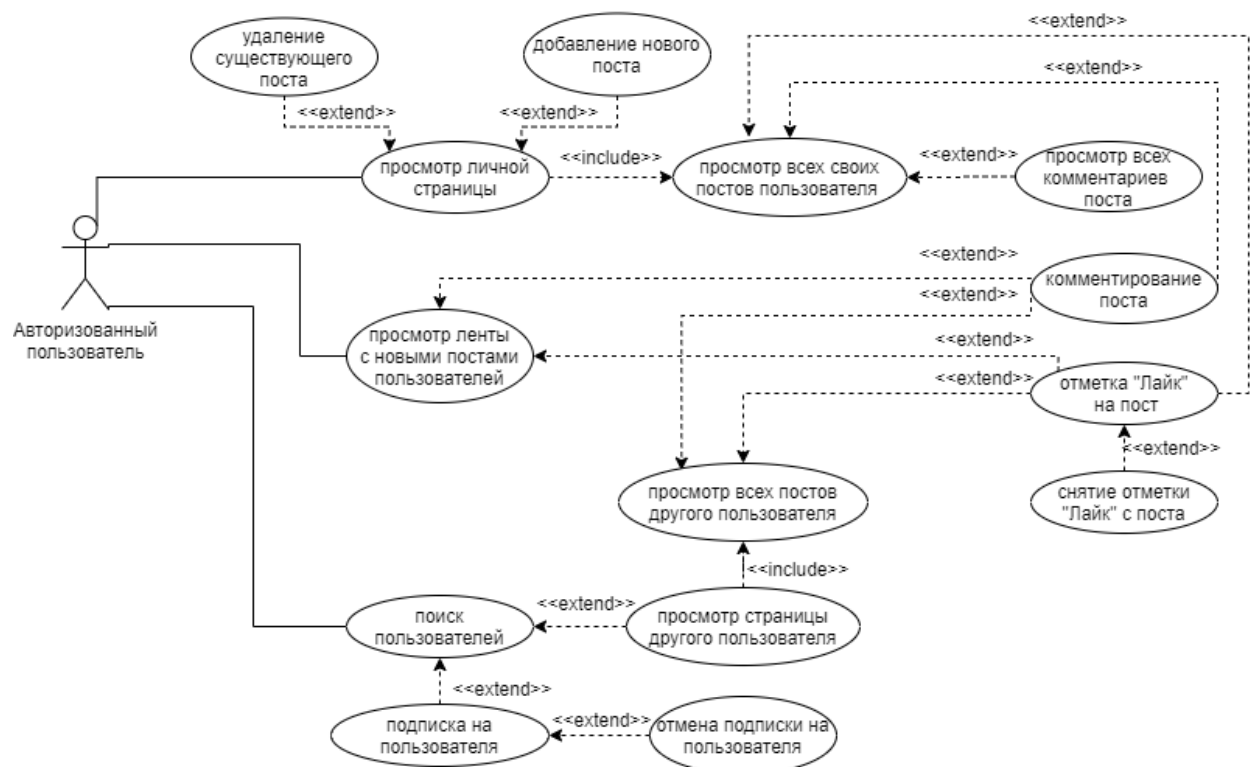


## Диаграмма вариантов использования UseCase.

Неавторизованный пользователь:

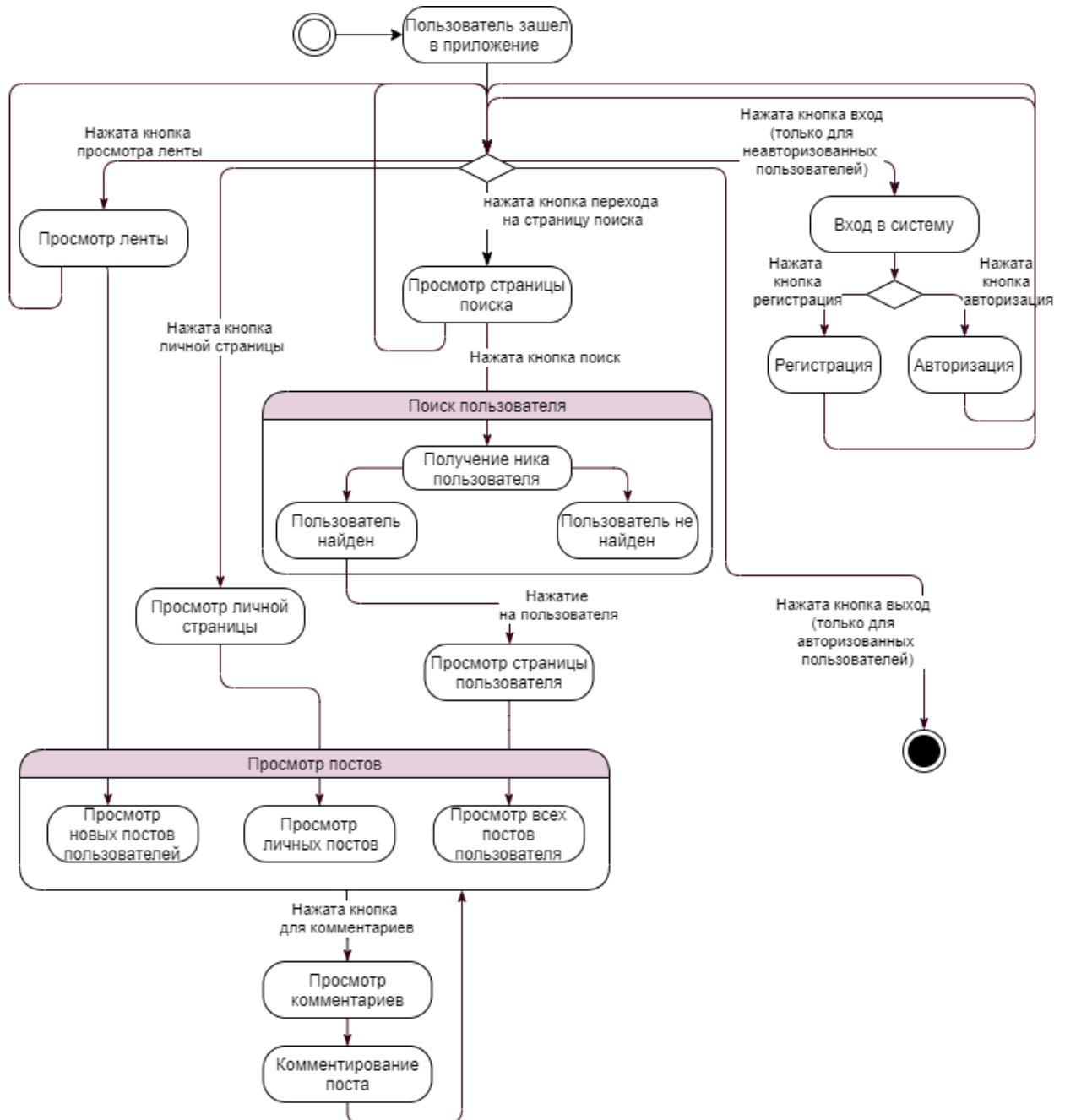


Авторизованный пользователь:

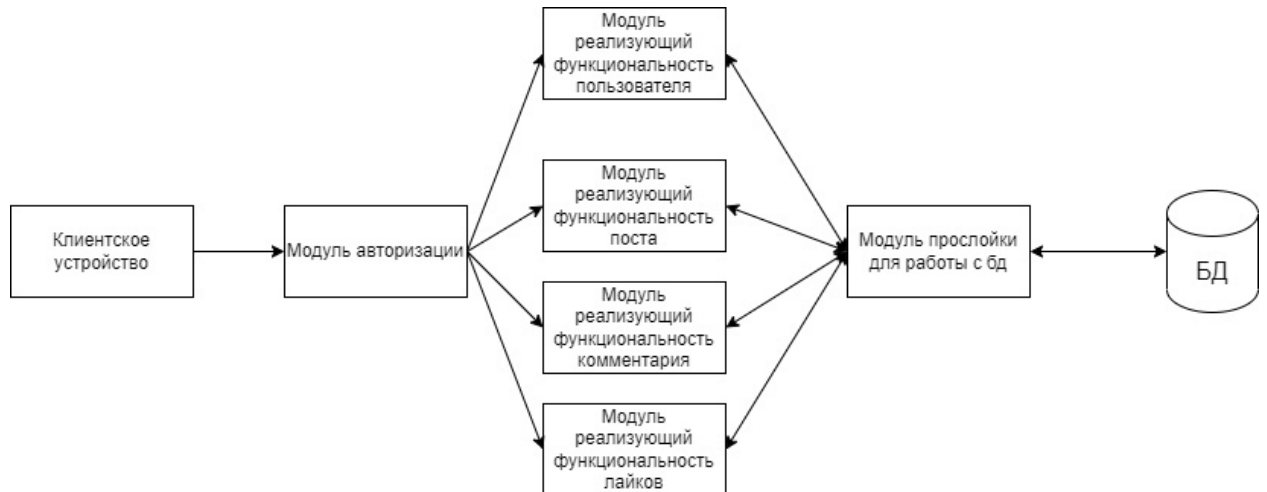




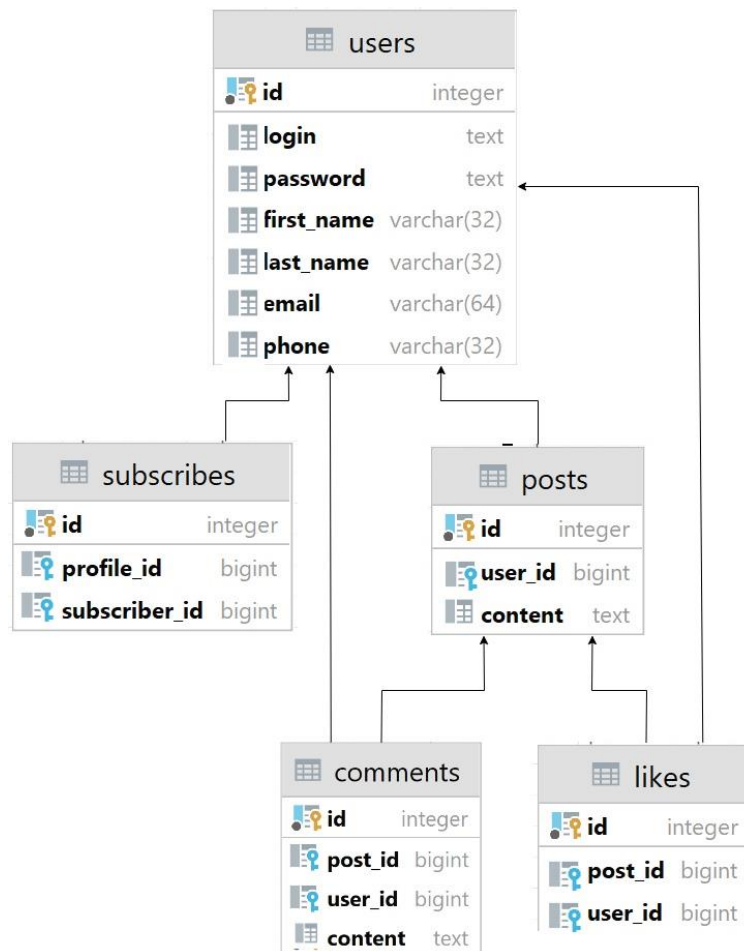
## Диаграмма состояний.



## 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.

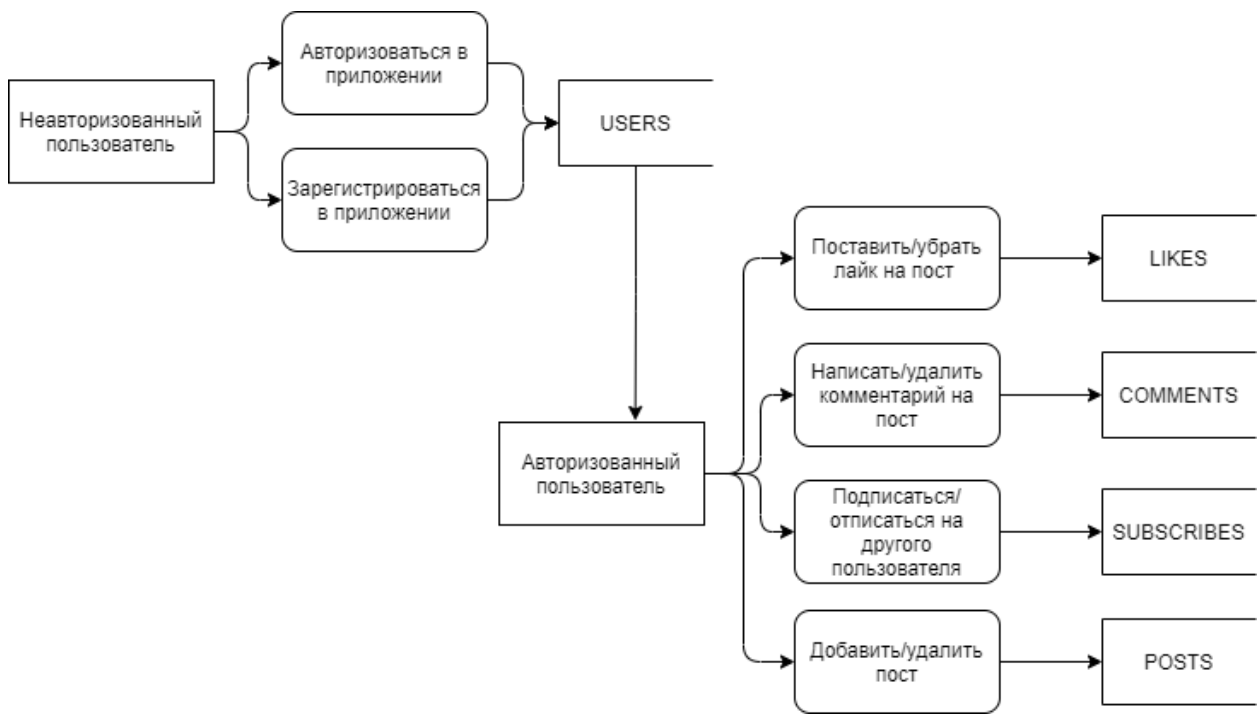
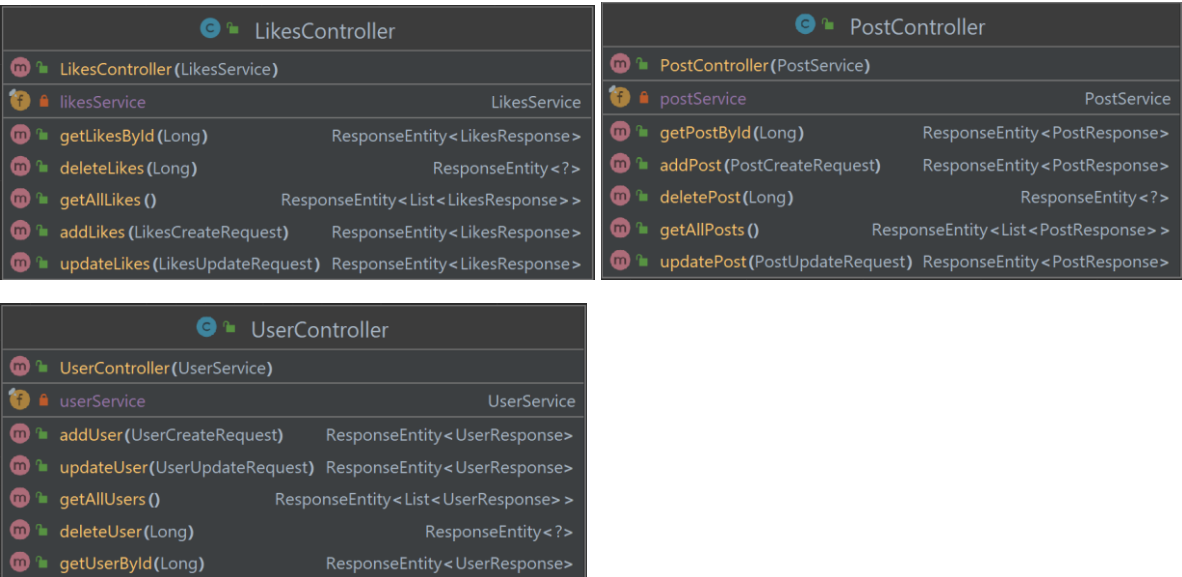


Диаграмма классов.

Диаграммы классов сервера.

Диаграмма классов библиотеки controller.



CommentsController	
CommentsController (CommentsService)	
commentsService	CommentsService
getCommentsById (Long)	ResponseEntity<CommentsResponse>
getAllComments ()	ResponseEntity<List<CommentsResponse>>
updateComments (CommentsUpdateRequest)	ResponseEntity<CommentsResponse>
deleteComments (Long)	ResponseEntity<?>
addComments (CommentsCreateRequest)	ResponseEntity<CommentsResponse>

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

## Библиотека Post:

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

## Библиотека Subscribe:

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

## Библиотека Users:



User	Comments
<b>User</b> (Long, String, String, String, String, String, String)	<b>Comments</b> (Long, String, String, String)
User()	Comments()
id	id
phone	post_id
login	content
password	user_id
lastName	
firstName	
email	
getId()	setPost_id(String)
getLogin()	getId()
setLogin(String)	getPost_id()
getPassword()	getUser_id()
getFirstName()	getContent()
setPhone(String)	setId(Long)
getLastName()	setUser_id(String)
getEmail()	setContent(String)
getPhone()	
setLastName(String)	
builder()	builder()
setEmail(String)	
setPassword(String)	
setFirstName(String)	
setId(Long)	

Диаграмма классов библиотеки mapper.

CommentsMapper
commentsToCommentsResponse (Comments)      CommentsResponse
commentsCreateRequestToComments (CommentsCreateRequest )      Comments
updateRequestToComments (CommentsUpdateRequest )      Comments

SubscribeMapper
subscribeToSubscribeResponse (Subscribe)      SubscribeResponse
updateRequestToSubscribe (SubscribeUpdateRequest )      Subscribe
SubscribeCreateRequestToSubscribe (SubscribeCreateRequest )      Subscribe

LikesMapper
updateRequestToLikes (LikesUpdateRequest )      Likes
likesToLikesResponse (Likes)      LikesResponse
likesCreateRequestToLikes (LikesCreateRequest )      Likes

PostMapper
PostCreateRequestToPost (PostCreateRequest)      Post
updateRequestToPost (PostUpdateRequest )      Post
postToPostResponse (Post)      PostResponse

UserMapper
UserCreateRequestToUser (UserCreateRequest)      User
userToUserResponse (User)      UserResponse
updateRequestToUser (UserUpdateRequest )      User

Диаграмма классов библиотеки repository.

CommentsRepository
SubscribeRepository
PostRepository
LikesRepository
UserRepository

Диаграммы классов клиента.

Диаграмма главного класса Main.

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

Диаграмма классов библиотеки ui.

Login:

NewUserLoginFragment		
m	NewUserLoginFragment ()	
f	phone	EditText
f	surname	EditText
f	name	EditText
f	exitButtton	ImageButton
f	password	EditText
f	email	EditText
f	login	EditText
f	registerButton	Button
m	onCreate(Bundle)	void

LoginActivity		
m	LoginActivity ()	
f	loginButton	Button
f	login	EditText
f	password	EditText
f	registerButton	Button
f	exitButton	ImageButton
f	loginByEmailButton	Button
m	onCreate(Bundle)	void

Post:

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

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



CommentListFragment	
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	
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	
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	
AddPostActivity ()	
exitButton	ImageButton
USER_ID	String
textPost	EditText
addPostButton	Button
onCreate (Bundle)	void

Search:

UserSearchViewModel	
UserSearchViewModel ()	
mText	MutableLiveData <String>
getText ()	LiveData <String>

UserSearchFragment	
UserSearchFragment ()	
binding	FragmentUserSearchBinding
onCreateView (LayoutInflater, ViewGroup, Bundle)	View
onDestroyView ()	void

User\_info:

UserInfoFragment	
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	
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:

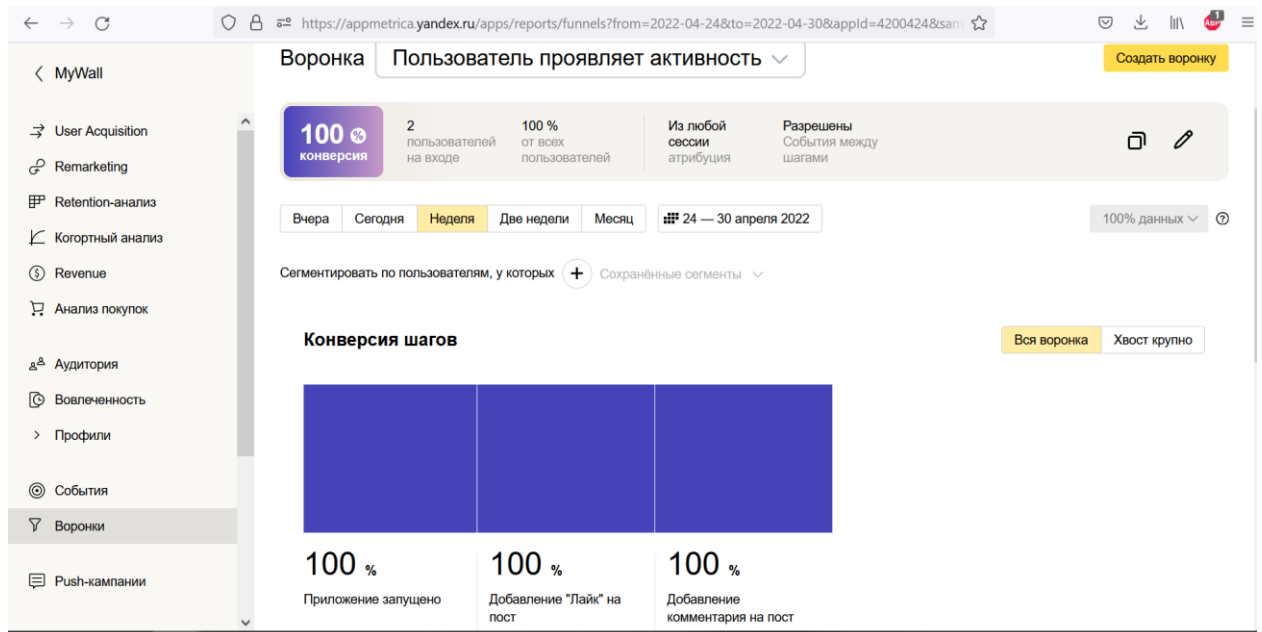
WallViewModel	WallFragment
<code>WallViewModel ()</code>	<code>WallFragment ()</code>
<code>mText MutableLiveData &lt;String &gt;</code>	<code>binding</code> FragmentWallBinding
<code>getText() LiveData &lt;String &gt;</code>	<code>postAdapter</code> PostAdapter
	<code>postsRecyclerView</code> RecyclerView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

## Сценарии воронок.

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

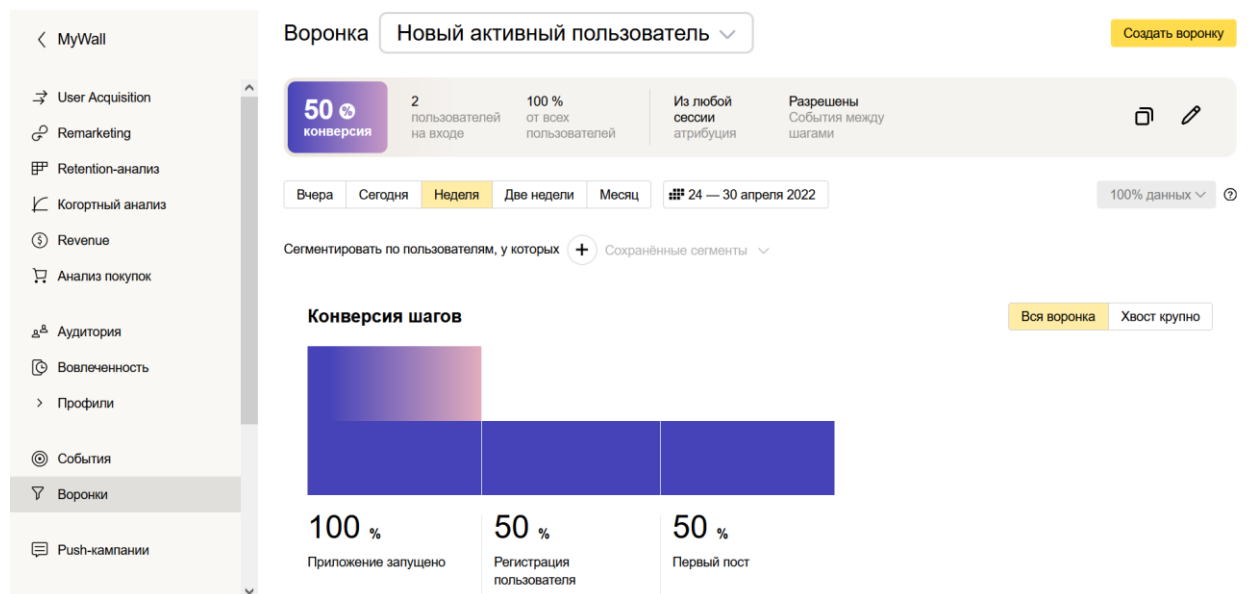
Воронка «Пользователь проявляет активность».

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



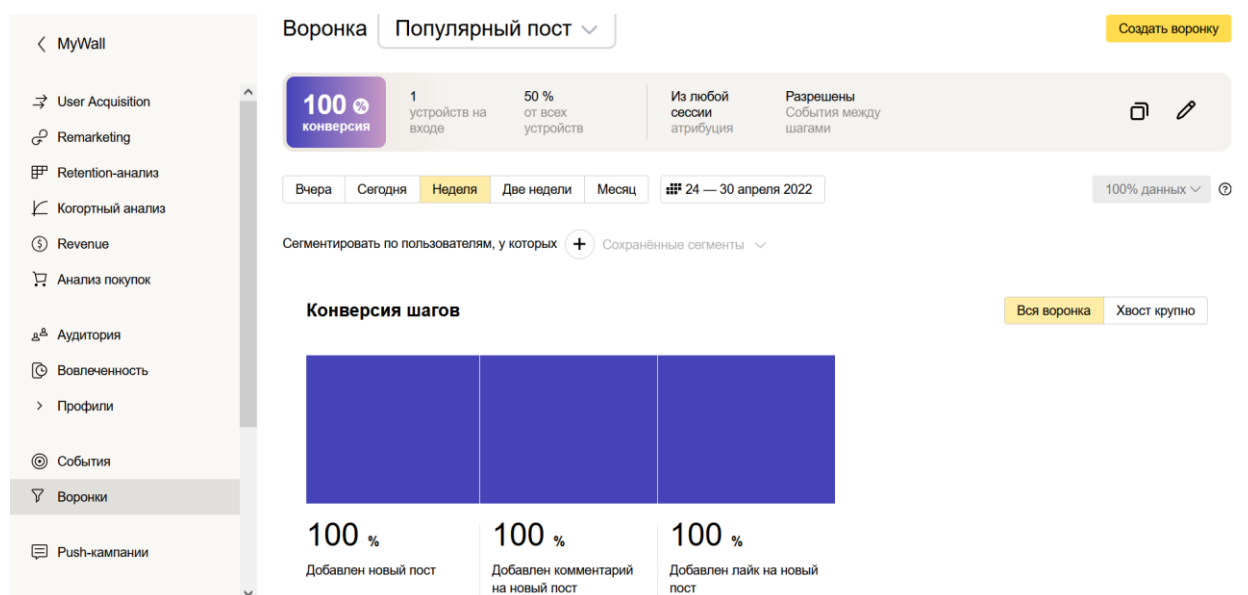
## Воронка «Новый активный пользователь».

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



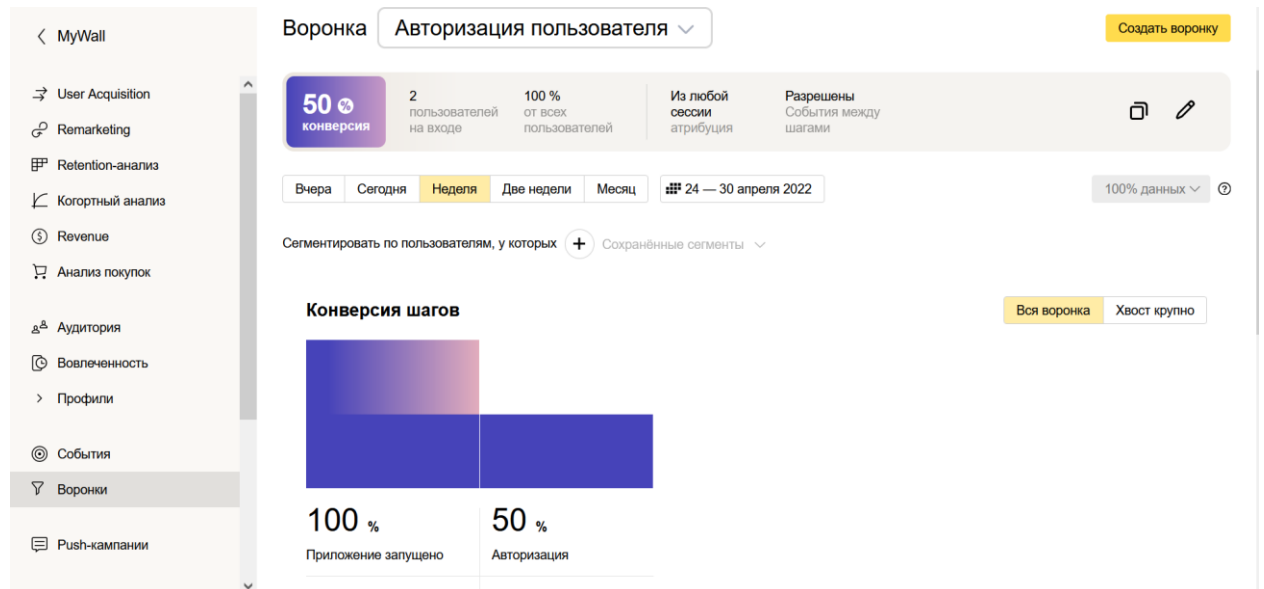
## Воронка «Популярный пост».

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.

## 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 subscribe 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.

## Календарный план.

задача	Дата начала	Дата завершения	Длительность
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 проекту	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

