Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

**Высшая школа программной инженерии**

**КУРСОВой проект**

по дисциплине

«Конструирование программного обеспечения»

Выполнили

студенты гр.3530904/80103 Д.А Фирсов

Д.О Королев

К.В Коршунов

Руководитель

доцент, к.т.н. А.С. Иванов

«14» декабря 2020 г.

Санкт-Петербург

2020

# **Проблема**

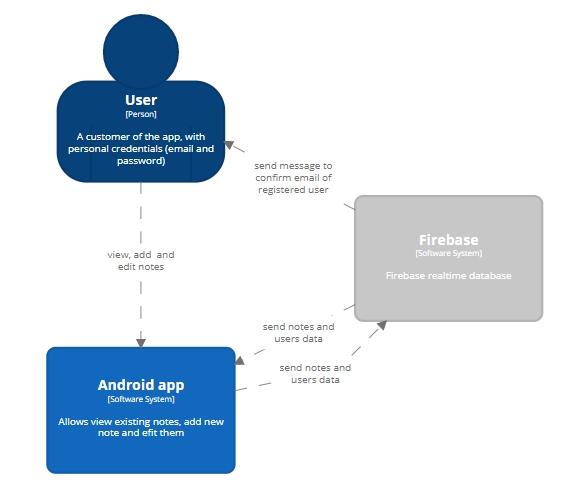
Данное приложение решает проблему структурирования заметок.

**Требования**

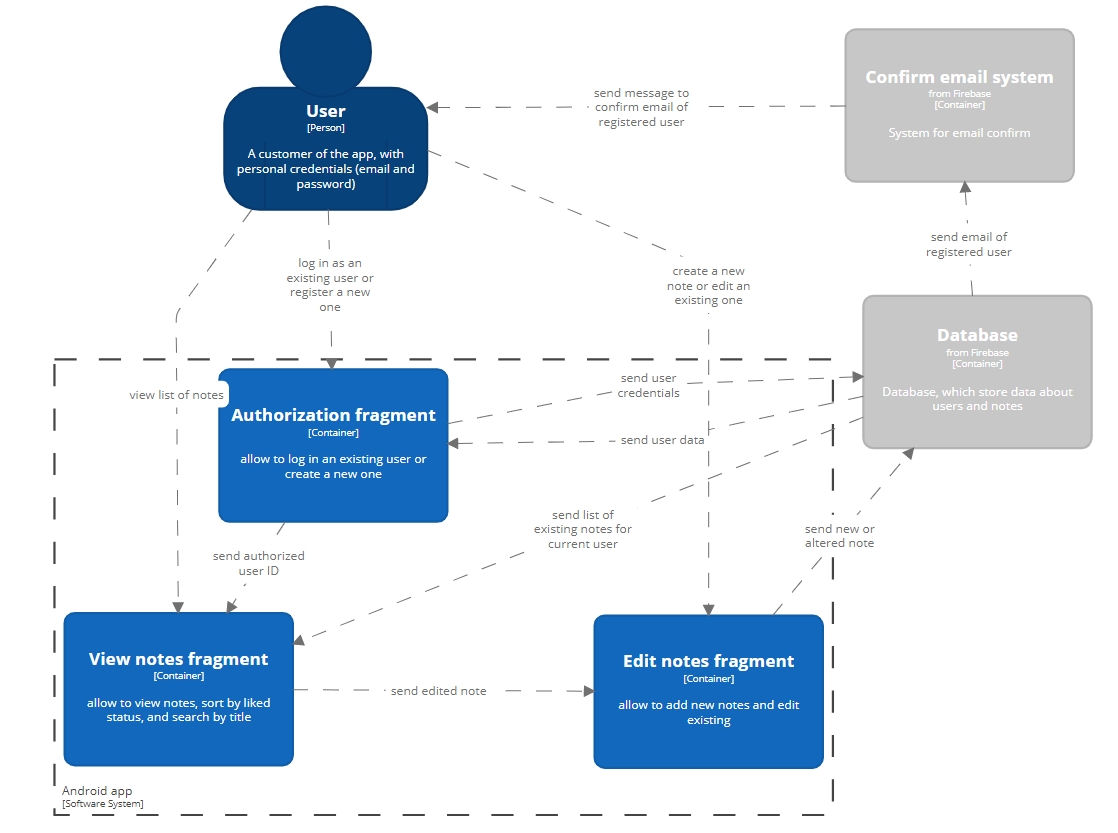
1. Разработать систему позволяющую удаленно хранить заметки, изменять, просматривать и удалять их
2. Реализовать возможность изменять статус «избранное» у заметок и сортировать их по данному статусу
3. Реализовать поиск заметок по заглавию

# **Диаграммы**

1. System Context diagram



1. Container diagram



# **Кодирование и отладка**

Проект представляет собой Android приложение, реализованное на языке kotlin с использованием nosql базы данных реального времени Firebase.

# **Unit тестирование**

Было реализовано Unit тестирование авторизации и регистрации новых пользователей в облачной базе данных Firebase. Тестирование было проведено с помощью фреймворка автоматического тестирования JUnit.

Данные тесты представлены в файле SoftwareDesign\_Note4\app\src\androidTest\java\com\university\softwaredesign\_note\FirebaseAuthUnitTest.kt

|  |
| --- |
| …  //smoke test which check that add and delete note is correct  @Test fun fir\_register\_smoke\_create\_and\_delete\_user\_test() {  val login = "testunit@ya.ru"  val password = "123456"   assertEquals(*runBlocking* **{**auth.createNewUser(login, password)**}**, true)  assertEquals(*runBlocking* **{**auth.signIn(login, password)**}**, true)  assertEquals(auth.getEmail(), login)  assertEquals(*runBlocking* **{**auth.deleteUser(login, password)**}**, true)  assertEquals(*runBlocking* **{**auth.signIn(login, password)**}**, false)  assertEquals(auth.getEmail(), "") }  //test which check that register is not correct when login already exists  @Test fun firebase\_register\_failed\_login\_already\_exists\_test() {  val login = "testunit@ya.ru"  val password = "123456"   *runBlocking* **{**auth.createNewUser(login, password)**}** assertEquals(*runBlocking* **{**auth.createNewUser(login, password)**}**, false)  assertEquals(*runBlocking* **{**auth.deleteUser(login, password)**}**,true) }  //test which check that register is not correct when password too short @Test fun firebase\_register\_failed\_password\_is\_too\_short\_test() {  val login = "testunit@ya.ru"  val password = "12"   assertEquals(*runBlocking* **{**auth.createNewUser(login, password)**}**, false) }  //test which check that sign in is successful @Test fun firebase\_signIn\_successfully\_test() {  val login = "testunit@ya.ru"  val password = "123456"   assertEquals(*runBlocking* **{**auth.createNewUser(login, password)**}**, true)  assertEquals(*runBlocking* **{**auth.signIn(login, password)**}**, true)  assertEquals(auth.getEmail(), login)  assertEquals(*runBlocking* **{**auth.deleteUser(login, password)**}**,true) }  //test which check that sign in is not successful when password is not correct @Test fun firebase\_signIn\_failed\_invalid\_password\_test() {  val login = "testunit@ya.ru"  val password = "1234564"   assertEquals(*runBlocking* **{**auth.createNewUser(login, password)**}**, true)  assertEquals(*runBlocking* **{**auth.signIn(login, password + "d")**}**, false)   assertEquals(*runBlocking* **{**auth.deleteUser(login, password)**}**,true) }  //test which check that sign in is not successful when login does not exist  @Test fun firebase\_signIn\_failed\_login\_does\_not\_exist\_test() {  val login = "testunit@ya.ru"  val password = "1234564"   assertEquals(*runBlocking* **{**auth.createNewUser(login, password)**}**, true)  assertEquals(*runBlocking* **{**auth.deleteUser(login, password)**}**,true)  assertEquals(*runBlocking* **{**auth.signIn(login, password + "d")**}**, false) }  … |

Также были реализованы Unit тесты, проверяющие корректность добавления и удаления заметок в базу данных Firebase.

Данные тесты представлены в файле SoftwareDesign\_Note4\app\src\androidTest\java\com\university\softwaredesign\_note\ FirebaseUnitTest.kt

|  |
| --- |
| …  @Test fun firebase\_add\_note\_is\_correct\_test() {  val note = Note.createNote()   assertEquals(*runBlocking* **{**addNote(note)**}**, true)  assertEquals(*runBlocking* **{**updateNotes()**}**, true)   assertEquals(notes.*value*?.size, 1)  val tmp = notes.*value*?.*find* **{** it **->** it.id == note.id **}** val index = notes.*value*?.*indexOf*(tmp)   val lastNote = notes.*value*?.get(index ?: 0)   assertEquals(*runBlocking* **{**deleteNote(note)**}**, true)  assertEquals(note, lastNote) }  @Test fun firebase\_delete\_note\_is\_correct\_test() {  val count = 3  val noteArray = ArrayList<Note>()  for(i in 0..count-1) {  val note = Note.createNote()  noteArray.add(Note.createNote())  assertEquals(*runBlocking* **{**addNote(note)**}**, true)  }   assertEquals(*runBlocking* **{**deleteNote(noteArray[1])**}**, true)  assertEquals(*runBlocking* **{**updateNotes()**}**, true)   assertEquals(notes.*value*?.size, 2)  assertEquals(notes.*value*!![0], noteArray[0])  assertEquals(notes.*value*!![1], noteArray[2])   assertEquals(*runBlocking* **{**deleteNote(noteArray[0])**}**, true)  assertEquals(*runBlocking* **{**deleteNote(noteArray[2])**}**, true) }  … |

# **Интеграционное** **тестирование**

Реализованы интеграционные тесты, которые покрывают следующие пользовательские сценарии:

1. Добавление и удаление заметки (fragment\_view\_model\_add\_Note\_is\_current\_test)
2. Изменение статуса “в избранных” заметки (fragment\_view\_model\_change\_like\_status\_is\_current\_test)
3. Поиск заметок по заглавию (fragment\_view\_model\_search\_is\_current\_test)
4. Сортировка заявок по статусу “в избранных” (fragment\_view\_model\_filter\_by\_like\_is\_current\_test)

Данные тесты представлены в файле SoftwareDesign\_Note4\app\src\androidTest\java\com\university\softwaredesign\_note\NoteViewIntegrationTest.kt

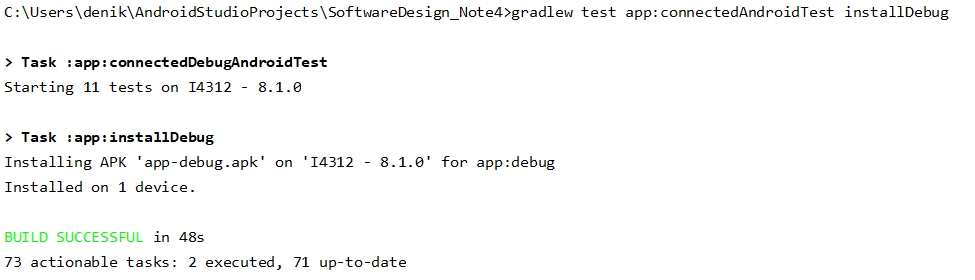
|  |
| --- |
| …  @Test fun fragment\_view\_model\_add\_Note\_is\_current\_test() {  val note = Note.createNote()  assertEquals(auth.getEmail(), "testunit@ya.ru")   viewModel.setData(*arrayListOf*(note))  viewModel.add(note)  viewModel.list()   assertEquals(1, notes.*value*?.size)  assertEquals(note, viewModel.getLast())  viewModel.delete(note)   assertEquals(0, notes.*value*?.size) }  @Test fun fragment\_view\_model\_change\_like\_status\_is\_current\_test() {  val initCount = notes.*value*?.size  val note = Note.createNote()   viewModel.setData(*arrayListOf*(note))  viewModel.add(note)  viewModel.list()   viewModel.changeLikeState(initCount ?: 0)  assertEquals(true, viewModel.getLast()?.liked)   viewModel.changeLikeState(initCount ?: 0)  assertEquals(false, viewModel.getLast()?.liked)   viewModel.delete(note) }  @Test fun fragment\_view\_model\_search\_is\_current\_test() {  val initialNoteArray = notes.*value* for(i in 0..((initialNoteArray?.size) ?:0 -1)) {  viewModel.delete(initialNoteArray!![i])  }   val count = 3  var noteArray = ArrayList<Note>()  val tittleArray = *arrayListOf*("tittle1", "middleTittle", "mineTittle")  for(i in 0..count-1) {  var note = Note.createNote()  note.title = tittleArray[i]  noteArray.add(note)  }   viewModel.setData(noteArray)   for(i in 0..count-1) {  viewModel.add(noteArray[i])  }  viewModel.list()   viewModel.search("mi")  viewModel.list()   assertEquals(noteArray[1], notes.*value*?.get(0))  assertEquals(noteArray[2], notes.*value*?.get(1))   for(i in 0..count-1) {  viewModel.delete(noteArray.get(i))  } }  @Test fun fragment\_view\_model\_filter\_by\_like\_is\_current\_test() {  val initialNoteArray = notes.*value* for(i in 0..((initialNoteArray?.size) ?:0 -1)) {  viewModel.delete(initialNoteArray!![i])  }   val count = 3  var noteArray = *arrayListOf*(Note.createNote(), Note.createNote(), Note.createNote())  viewModel.setData(noteArray)   for(i in 0..count-1) {  viewModel.add(noteArray[i])  }  viewModel.list()   viewModel.changeLikeState(2)  viewModel.list()  val likedTrueNote = notes.*value*?.get(2)  viewModel.filterByLike()  viewModel.list()   assertEquals(true, notes.*value*?.get(0)?.liked)  assertEquals(likedTrueNote, notes.*value*?.get(0))   for(i in 0..count-1) {  viewModel.delete(noteArray.get(i))  } }  … |

# **Сборка**

Для сборки, выполнения тестов, интеграционных тестов и запуска приложения необходимо выполнить следующею команду в терминале в директории с проектом.

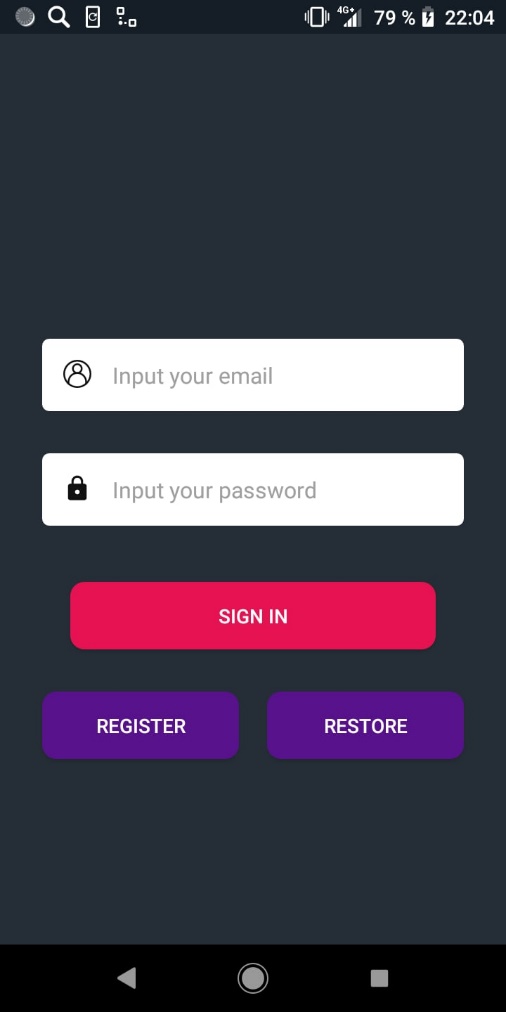
|  |
| --- |
| *gradlew test app:connectedAndroidTest installDebug* |

Результат исполнения команды:

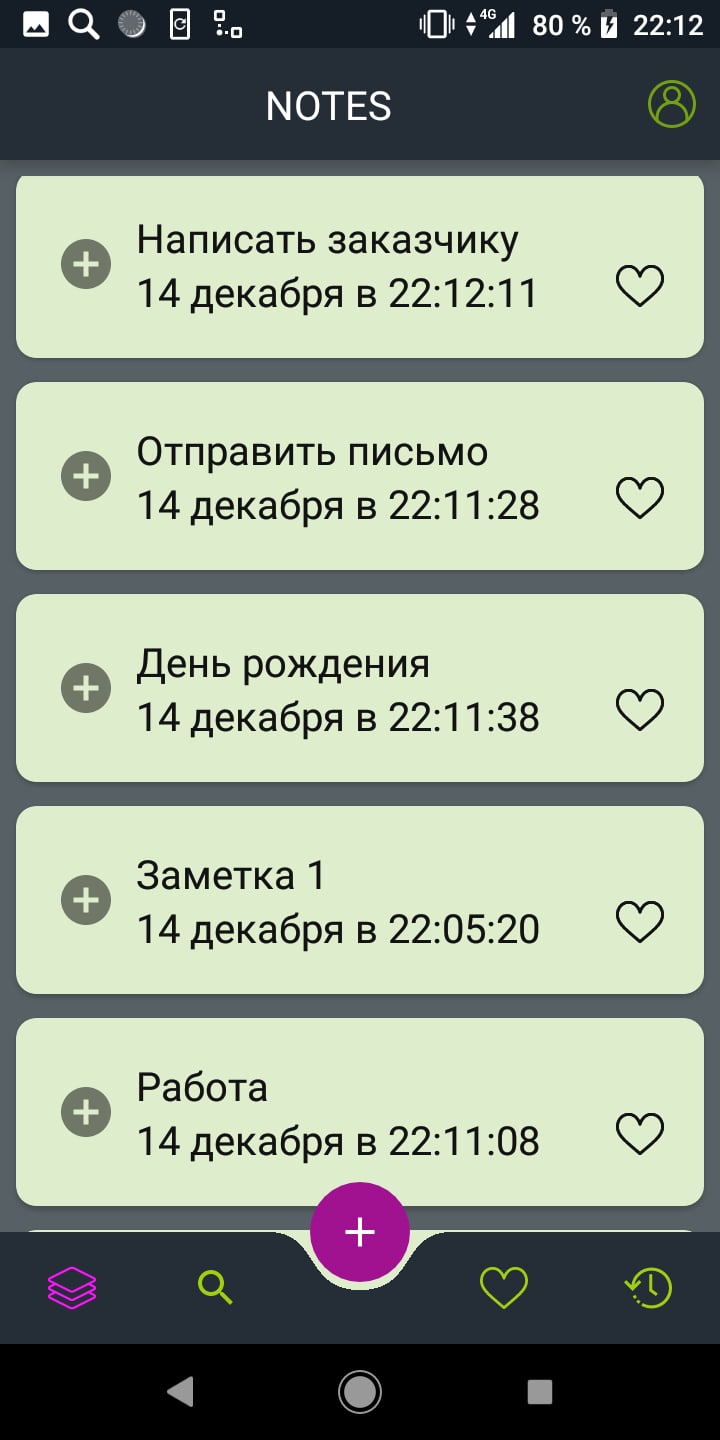


# **Скриншоты работы приложения**

1. Окно авторизации и регистрации



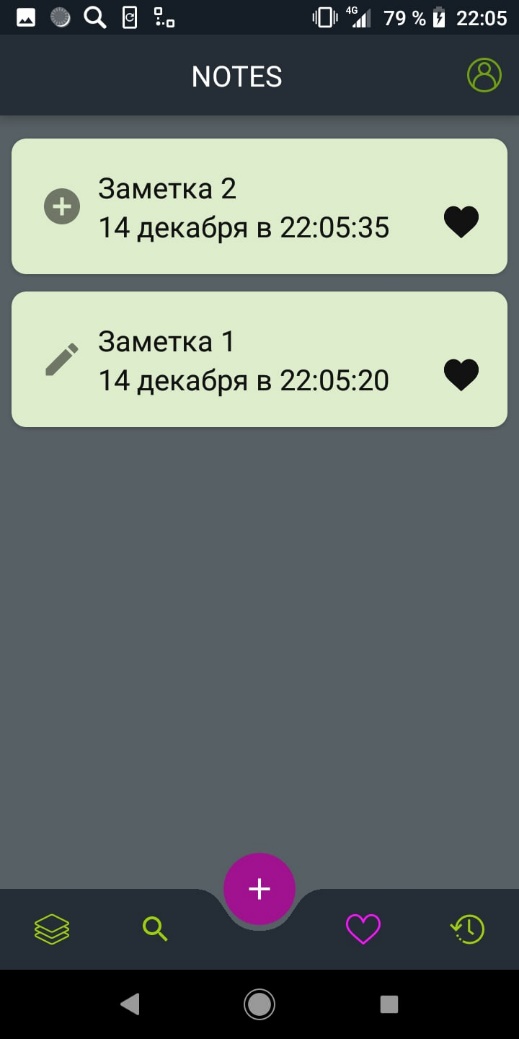
1. Главное окно, в котором отображены все существующие заявки



1. Окно редактирования заметок и добавления новых



1. Окно избранных заметок



1. Окно поиска

