Министерство науки и высшего образования РФ

федеральное государственное автономное образовательное учреждение высшего образования «НИТУ МИСиС»

Институт ИТАСУ

Кафедра АСУ

Отчет на тему:

**«Разработка и тестирование клиент-серверной архитектуры приложения “Messenger”»**

по дисциплине:

«Тестирование программных комплексов»

Выполнил: ст. группы МИВТ-19-2-7

Будяков А.И

Проверил: ассистент каф. АСУ

Рзазаде Ульви Азар оглы

Москва 2020

Оглавление

[ОПИСАНИЕ ПРОЕКТА 4](#_Toc41127764)

[ТЕХНИЧЕСКОЕ ЗАДАНИЕ 5](#_Toc41127765)

[Серверная часть: 5](#_Toc41127766)

[Клиентская часть: 5](#_Toc41127767)

[ТЕХНОЛОГИЧЕСКИЙ СТЕК 6](#_Toc41127768)

[АРХИТЕКТУРА 7](#_Toc41127769)

[Класс Client 7](#_Toc41127770)

[Класс Server 8](#_Toc41127771)

[Класс ConnectionDB 9](#_Toc41127772)

[Класс Tests 9](#_Toc41127773)

[ДИАГРАММЫ АРХИТЕКТУРЫ 10](#_Toc41127774)

[Диаграмма классов и компонент QML. 10](#_Toc41127775)

[Диаграмма взаимодействия Backend и БД. 12](#_Toc41127776)

[Диаграмма тестирующего класса на базе фреймворка QTest 13](#_Toc41127777)

[МЕТОД УПРАВЛЕНИЯ ПРОЕКТОМ 13](#_Toc41127778)

[МЕТОД ТЕСТИРОВАНИЯ 14](#_Toc41127779)

[Функциональное тестирование 14](#_Toc41127780)

[Восходящее тестирование. 15](#_Toc41127781)

[ИНТЕРФЕЙС 16](#_Toc41127782)

[Окно входа в приложение. 16](#_Toc41127783)

[Окно регистрации. 17](#_Toc41127784)

[Кнопка выбора собеседника 18](#_Toc41127785)

[Окно отправки сообщения 19](#_Toc41127786)

[РЕЗУЛЬТАТЫ ТЕСТОВ 21](#_Toc41127787)

[ЗАКЛЮЧЕНИЕ 23](#_Toc41127788)

# ОПИСАНИЕ ПРОЕКТА

Данный проект представляет собой приложение “Messenger”, для Linux, предоставляющий возможности передавать сообщения между пользователями в сети интернет.

В проекте участвует один человек по совместительству выполняющий следующие проектные роли:

* Full-stack программист,
* Сетевой программист,
* Тестировщик,
* Менеджер проекта

Этапы разработки программного продукта представляются следующими этапами:

* анализ требований к проекту ;
* проектирование;
* реализация;
* тестирование продукта;
* внедрение и поддержка.

В данном отчете будут представлены этапы разработки программного продукта в той форме, в которой они были реализованы, а так-же даны пояснения к стратегиям разработки и тестирования, которые были выбраны.

# ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Зафиксируем предъявляемыми требованиями к продукту в виде технического задания. Так как проект требует больших трудозатрат и является архитектурно объемным, в первую очередь требуется довести его до *Альфа* версии. Поэтому сейчас и далее, в данном отчете мы будем касаться лишь функциональных требований к разрабатываемому продукту, оставив за рамками такие виды тестирования как нагрузочное и тестирование безопасности, оставив их для *Бета* версии.

ТЗ.

## Серверная часть:

* Многопоточная обработка входящих данных от клиентов и перенаправление сообщений адресату,
* Хранение сообщений, имен пользователей и паролей в базе данных.

## Клиентская часть:

* Красивый интерфейс,
* Возможность отправки сообщений серверу,
* Окно регистрации,
* получения данных о предыдущих переписках с сервера.

*Справка:*

Жизненный цикл разработки ПО:

• Пре-альфа

• Альфа

• Бета

• Релиз-кандидат

• Релиз

• Пост-релиз

# ТЕХНОЛОГИЧЕСКИЙ СТЕК

Выбор технологического стека опирался в первую очередь на имеющиеся навыки разработчика и возможность кросс-платформенного портирования приложения в будущем.

В проекте будут применяться следующие технологии:

* *Серверная часть*: **С, C++, SQL**;
* *Клиентская часть*: **С, C++, QML, JS**;
* *База данных*: **MySQL**.

В качестве основного языка программирования был выбран **C++,** обладающий высокой производительностью, что хорошо подходит для серверной части архитектуры (см. раздел архитектура)

Представление пользовательского интерфейса реализовано на базе нового фреймворка и декларативного языка финской компании **Qt Company** – **QML**. В пользу данного интерфейса говорит его легкая портируемость как на десктопные версии программы, так и на мобильные. Стоит так же отметить что данный фреймворк позволяет писать кроссплатформенные решения, тем самым позволяя распространить продукт под все виды современных ОС, такие как Windows, Linux, Mac, IOS, Android. Так же он позволяет делать вставки из **JS** кода, что для фронтэнда является существенным плюсом Более того данный фреймворк легко позволяет встраивать бизнес логику, или модель данных написанную на **C++.**

В качестве модели хранения данных используется база данных поддерживаемая СУБД MySQL, легко в будущем портируема для возможной браузерной версии приложения на базе веб-сервера Apache.

# АРХИТЕКТУРА

Архитектура условно поделена на клиентскую часть, серверную и базу данных.

Опишем основные классы и методы данных, объектных модулей, представляемых на рисунках следующей главы.

## Класс Client

Основной класс бизнес-логики клиента.

Описание методов:

* startClient – подключение к сокету;
* getMessageFromUser – метод включает прослушивание порта, на прием сообщений от других пользователей;
* sendMessageToUser – отправка серверу сообщения для другого клиента;
* userLogin – отправка запроса на вход в аккаунт серверу;

## Класс Server

Основной класс бизнес-логики сервера.

Описание методов:

* Server – конструктор принимающий порт, имя хоста, ip адресс хоста, название таблицы с пользователями БД, название поля с логинами, название поля с паролями БД;
* startListening – цикл в котором создаются потоки обработки клиентов (см функцию listenClientThread);
* establishConnection – начало прослушивания порта;
* listenClientThread – функция потока для обработки сообщений конкретного пользователя;
* findUserByValue – функция для поиска в словаре по значению;
* login – функция-запрос БД, на соответствие логина и пароля присланного пользователем
* registration – функция запрос на добавления пользователя (уникальное поле) в БД
* drop – удаление пользователя в БД
* getOnlineUsers – возвращает массив подключенных пользователей.

## Класс ConnectionDB

Базовый класс класса Server.

Описание методов:

* connectDB – создает соединение с БД посредством API фреймворка библиотеки Qt;

## Класс Tests

Основной класс использующий фреймворк QTest.

Методы:

* TestServerSocket() – тестирование соединение посредством сокета к порту;
* TestConnectionDB() – тестирование соединения с БД;
* TestLoginDB() – Тестирование входа пользователя;
* TestRegistrationDB() – тестирование регистрации пользователя;
* TestDropDB() – тестирование удаления пользователя;
* TestGetOnlineUsersDB() тестирование функции получения текущих вошедших в систему пользователей;

# ДИАГРАММЫ АРХИТЕКТУРЫ

## Диаграмма классов и компонент QML.

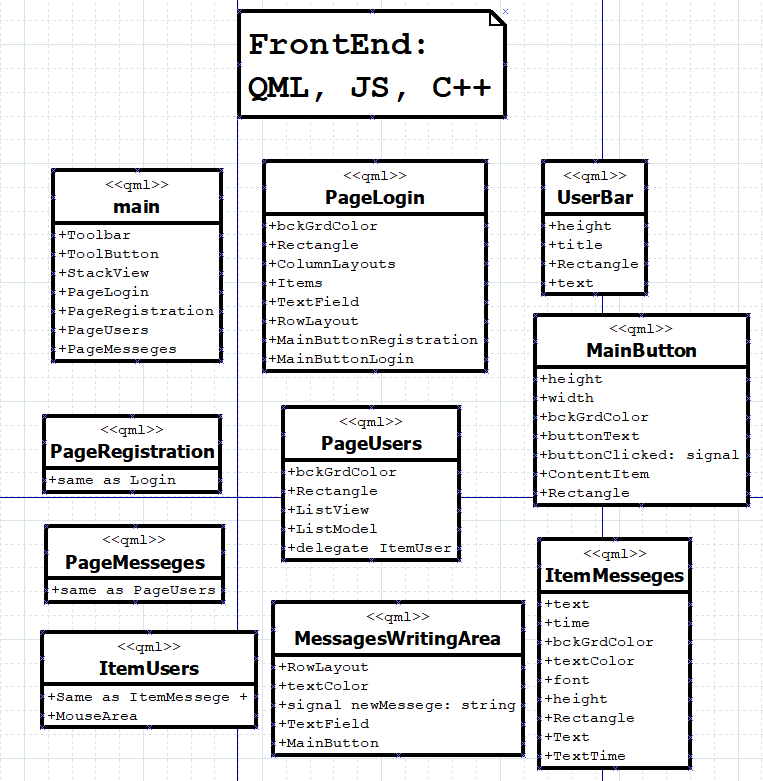


Рисунок 1. Диаграмма классов и компонент QML.Диаграмма классов и методов Backend части.

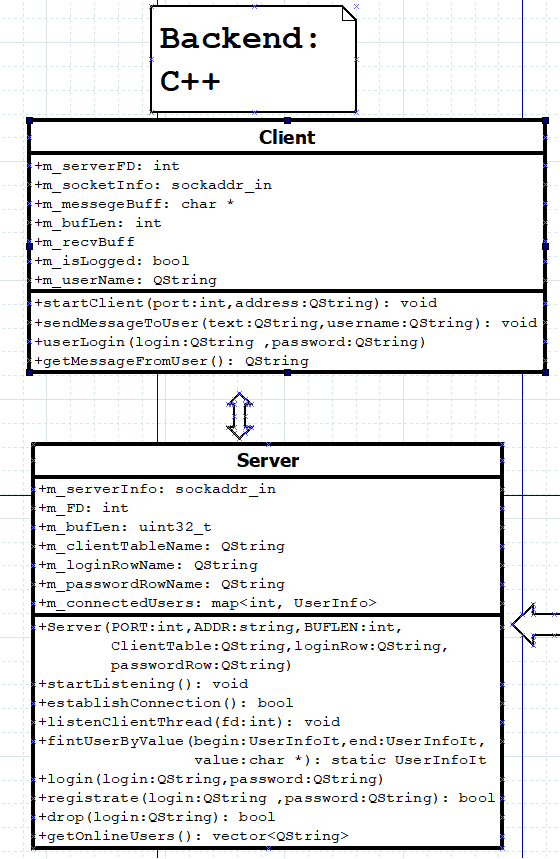


Рисунок 2. Диаграмма классов и методов Backend части.

## Диаграмма взаимодействия Backend и БД.

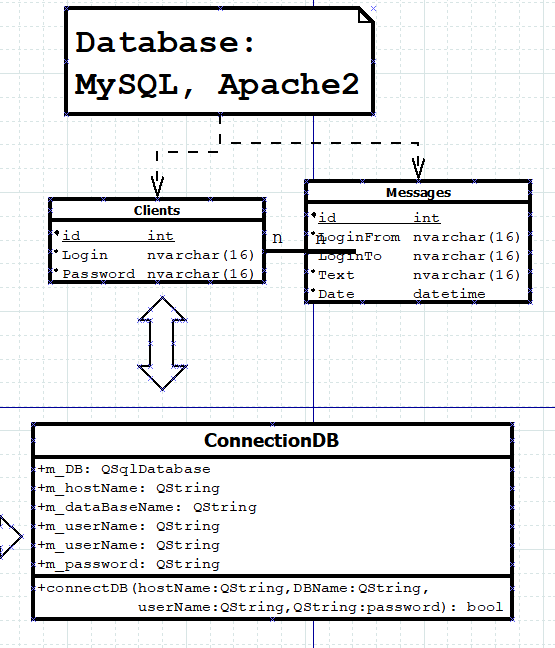


Рисунок 3. Диаграмма взаимодействия Backend и БД.

## Диаграмма тестирующего класса на базе фреймворка QTest

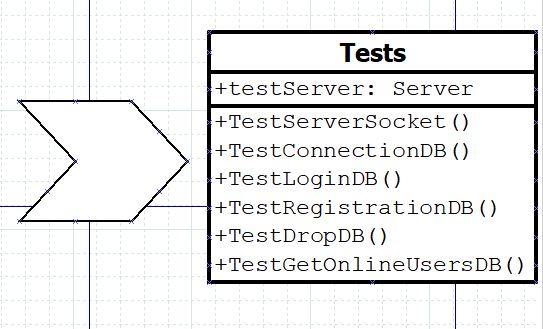


Рисунок 4. Тестирующий класс на базе фреймворка QTest

# МЕТОД УПРАВЛЕНИЯ ПРОЕКТОМ

Так как проект реализуется одним человеком, исключается накладные на взаимодействие разработчиков в команде, однако увеличивается объем работы и время разработки. Однако при проектировании сложной архитектуры обойти стороной тестирование просто невозможно. В какой-то момент после написания очередной функции, было принято решение что, дальнейшая разработка, без уверенности в правильности уже реализованного функционала, просто нельзя (см. следующую главу). Был изучена и применена в той или иной степени методика Agile, так как походу реализации проекта, становилось ясно, что реализация полнофункционального мессенджера, задача не из легких, и приходилось жертвовать некоторыми изначальными требованиями тз, в угоду реальности окончания проекта.

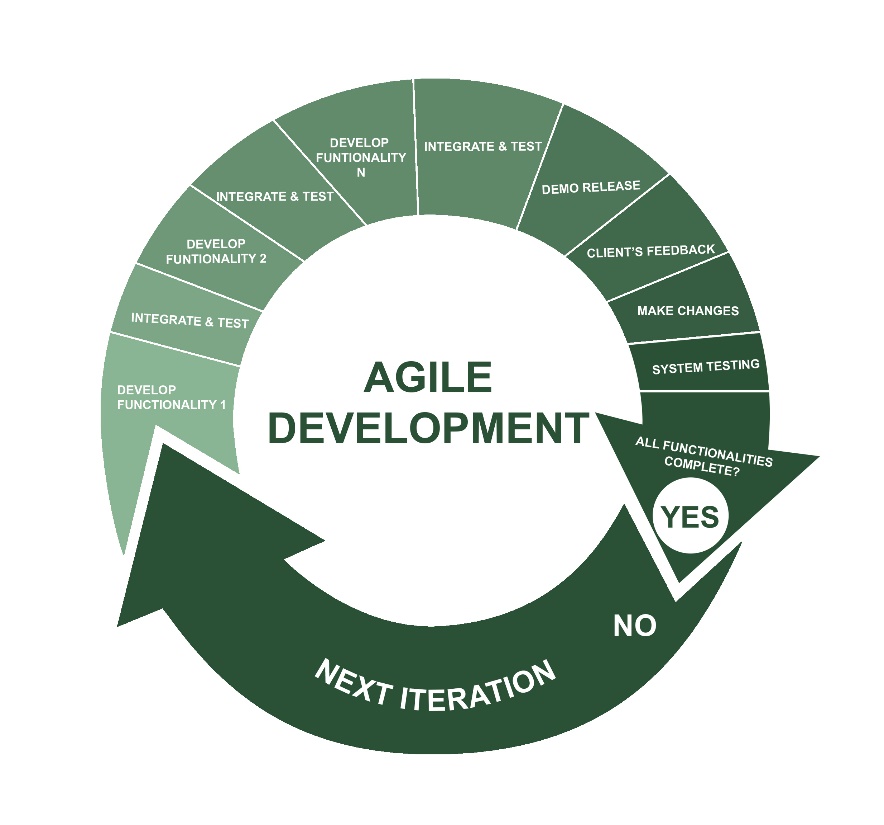


Рисунок 5. Схематичное изображение итераций в Agile методике.

# МЕТОД ТЕСТИРОВАНИЯ

## Функциональное тестирование

На текущий момент большинство функций серверной части покрыты тестами.

Как уже было сказано ввиду особенностей командной составляющей в текущем отчете на момент написания рассмотрено лишь функциональное тестированиe.

*Справка.*

Функциональное тестирование рассматривает заранее указанное поведение и основывается на анализе спецификаций функциональности компонента или системы в целом.

Подразделяется на:

* Тестирование пользовательского интерфейса (GUI Testing)
* Тестирование безопасности (Security and Access Control Testing)
* Тестирование взаимодействия (Interoperability Testing)

## Восходящее тестирование.

Как уже было сказано в предыдущем модуле, в какой-то момент, тесты стали необходимостью для практически каждого модуля, разрабатываемой системы. Такой метод называется **восходящим**. Стоит отметить что использовалось Unit- тестирование, позволяющее, как раз разбив программу на составляющие создавать тесты к каждому компоненту системы.

*Справка.*

При восходящем тестировании низкоуровневые модули, процедуры или функции собираются воедино и затем тестируются. После чего собирается следующий уровень модулей для проведения интеграционного тестирования.

# ИНТЕРФЕЙС

## Окно входа в приложение.

Два текстовых поля, и две кнопки.

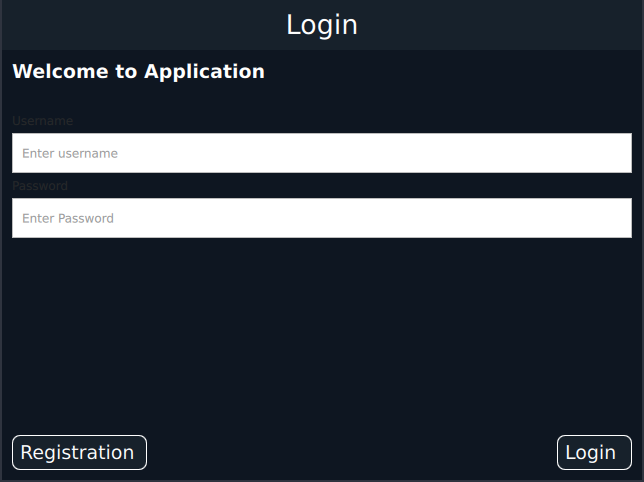


Рисунок 6. Интерфейс страницы входа

## Окно регистрации.

Кнопка возврата сверху, два текстовых поля и кнопка подтверждения регистрации.

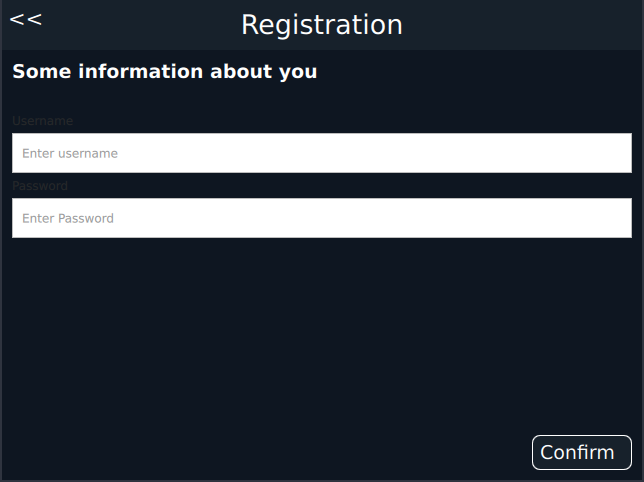


Рисунок 7. Окно регистрации

## Кнопка выбора собеседника

Список пользователей, на которых можно нажать и перейти к переписке, кнопка возврата.

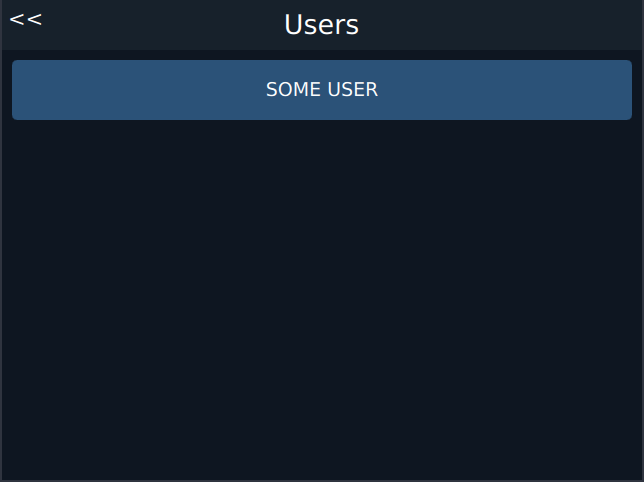


Рисунок 8. Окно выбора собеседника

## Окно отправки сообщения

Кнопка возварта, отправки, и список сообщений, бар показывающий имя пользователя, которму отправляется сообщение

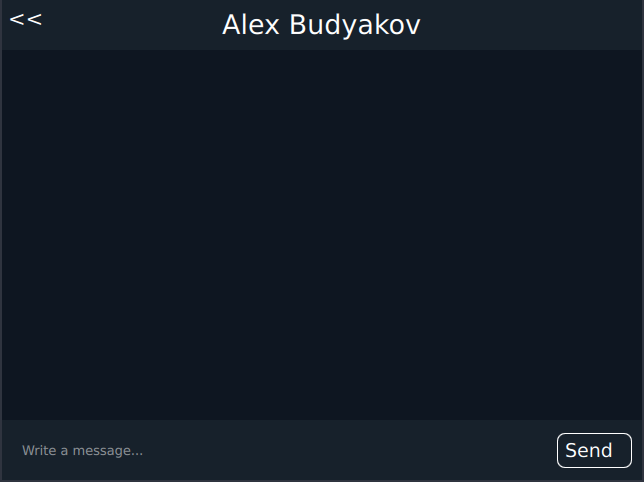


Рисунок 9. Окно отправки сообщения

Окно отправки сообщения с сообщениями

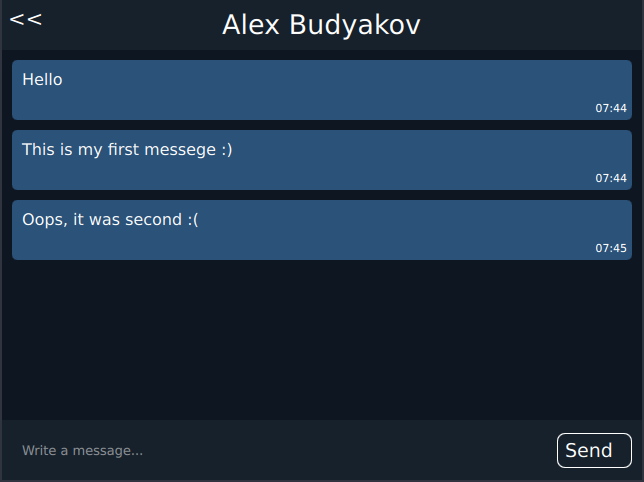


Рисунок 10. Окно отправки сообщений с сообщениями.

# РЕЗУЛЬТАТЫ ТЕСТОВ

Результаты тестов серверной части:

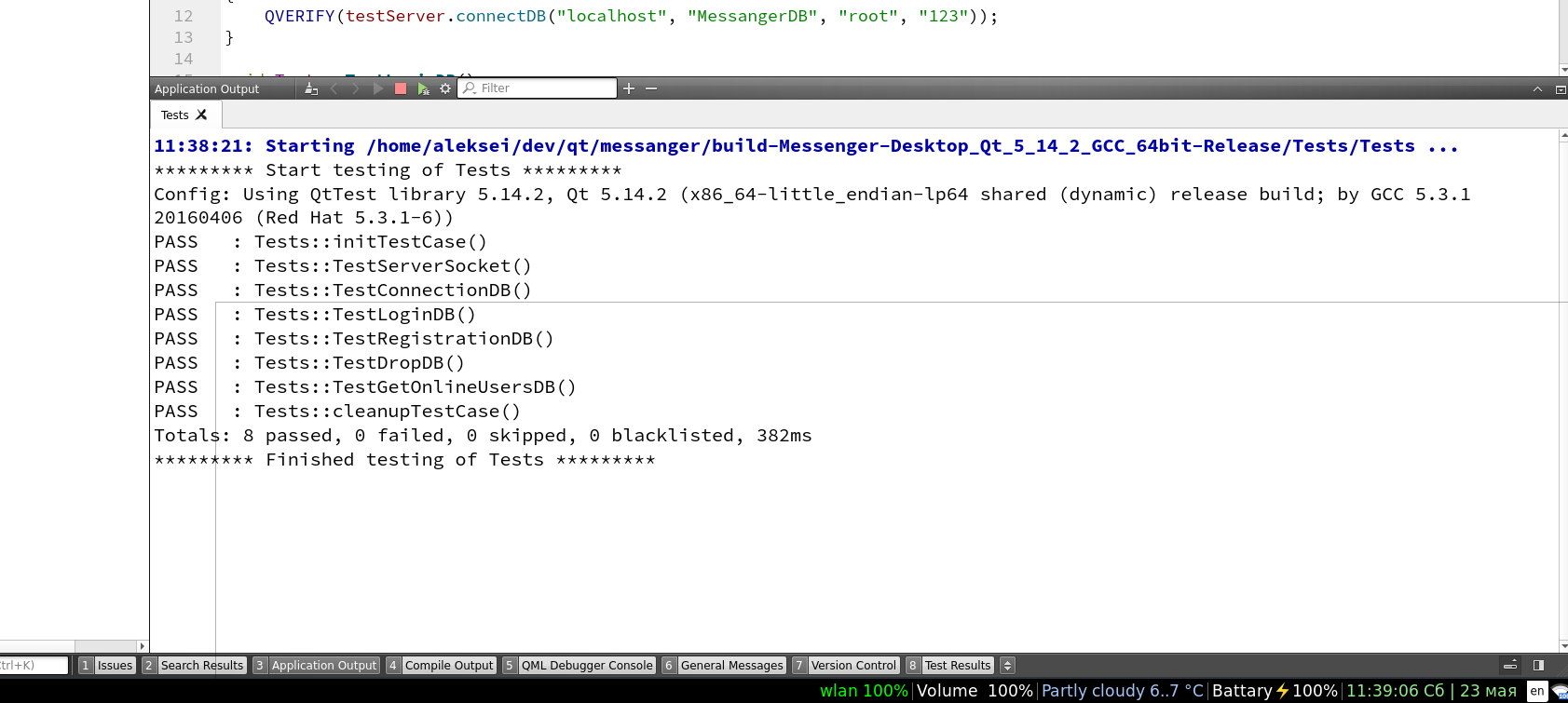


Рисунок 11. Окно вывода тестов

Допустим намеренно ошибку в пароле подключения к БД.



Рисунок 12. Окно вывода тестов с ошибкой

# ЗАКЛЮЧЕНИЕ

В проекте удалось реализовать многопоточный сервер взаимодействующий с базой данных на C++, а так же интерфейс приложения на QML и большую часть бизнес логики клиентской части. Почти полностью была покрыта тестами серверная часть приложения.

К сожалению, на текущий момент командой разработчиков приложение еще не было завершено, и аспекты взаимодействия интерфейсной части фронт-енда с бекендом остаются в процессе разработки. Но так как дедлайн закончился отправляется пре-альфа версия приложения. best regards

Исходный код,

<https://github.com/artofnothingness/Messenger>

Весь материал по диаграммам и отчету в папке /Content