# Содержание

1. [Введение](#X81abf0a3d09417f4daac988b8dba10f0ac2ca8c)
2. [Теоритическая часть](#X6316f88a9dca3764ea02fcc74f3117d7bbfe0c0)
   1. [Организация ЧОП](#a66c7beef7ee05ce09ccd305758a4b34)
   2. [Информационные потоки ЧОП](#Xeede502369471251ce873105b877e62664b5064)
   3. [Процессы внутри ЧОП](#bd328bb371cabcf61a023dd89284d430)
      1. [Описание процессов](#X6ba5f9391dcd40c5a9f87c85960590d2214ebf1)
         1. [Составление контракта](#a10e8e66a8fb12520c93c572664604c9)
         2. [Закупка у поставщика оружия](#Xa6d5579cc6cbaff227256e16441675a2b518c01)
         3. [Прием на работу](#Xe475acf7de394b588f3b29ec8edfe21cb684d7a)
         4. [Регистрация и выдача оружия](#a67dfdaec394ee405cbad740a91f9c25)
         5. [Составление отчета об инцеденте на объекте](#Xce85023636bff45809f09574dd26e262b05ec12)
         6. [Выплаты](#X865a7f56be0586e812dcb5d8d56e98bec44da0d)
3. [Описание средстд разработки](#f61c3d8237771dae61110cd55542d40f)
   1. [Утилиты для разработки баз данных](#X482934e5b77bc1e18d5d3de368fa932c4d35989)
      1. [SQLite v3.35.5 stable](#Xbcc6eadd88454e6ce7b5ba1343dd6c75870e1d5)
      2. [SQLiteBrowser v3.12.2](#d9107f160c4cbd3769c9deef66bb16b0)
      3. [DBVisualizer v12.1.8](#c5cd767e2d7140d75b7479628ef2625d)
   2. [Библиотеки](#X76760c012e06613ae0bc017fb2d64aaa04f7110)
      1. [Qt v6.2.4](#X896d42445c381ccb1305b4f0b83decfe5ff2d75)
   3. [Системы сборки и компиляторы](#c41121e7ff1f037c52c40f6800344b60)
      1. [CMake v3.22.3](#d769a93d450808dd2699e4d277b38a96)
      2. [Ninja c1.10.2](#X4523f57e2b99abc79efe245d773210d09ebd0c2)
      3. [clang v13.0.1](#X9aa055228aaa75e6eaf1c99ebed595103e34a4b)
   4. [IDE](#a9d5433c4ced0429602d686763b511c0)
      1. [QtCreator v6.0.2](#Xee9ce94f9bf6cdcdc193fb94ad119408d093f67)
      2. [VIM v8.2](#be86898d7ac3847f7ca9aa4e09d807c7)
         1. [Плагины](#X2e650f6effb50cc9d1e875cd7be4e6cac4351a8)
   5. [Разное](#X987ae0d15d442482ecf3b624fd71853d41cdff9)
      1. [Pandoc](#e9585a13fb650b9f3c9eeef243be6d8e)
      2. [Plantuml v1.2021.16](#X7fb93f4a69db054350dfaef24b16300fdf61851)
      3. [BASH v5.1.16](#a7809f369ead5a4aba1407bab398c327)
   6. [Отладка](#X8e66e1d76d86192d570db0409cb8850956502eb)
      1. [Valgrind v3.18](#X40ebc304fffd7fc86891c1a97c57a21339b113d)
   7. [VCS](#b7249081aeaa3bc1b21f5a8f16eeb88e)
      1. [GIT v2.35.1](#be9c09aacbde6d23d67f89a3fca38c08)
         1. [GitHub](#Xa6634c6045ed67a4172461508e93823e6a0e77a)
4. [Разработы базы данных ЧОП](#eb1fa9a0c02763fe5749637a8d16ac3a)
   1. [Таблицы](#X536ad542a8c325f780bdf28dc0ba757f62e5202)
   2. [Визуализация базы данных](#Xd6768691e4a0ce31f7c063cc24ed591a2505ccc)
5. [Разработка архитектуры приложения](#Xe69fba65165835d15a1f2c48c1eb7a168df2533)
   1. [Основа](#X4031212a876b499a8bcb1288c7f84ac9dd04d42)
      1. [Информационные потоки ЧОП с учетом сервера](#Xf28ab9ba1a570320e89c767a71e971b1a01552d)
   2. [База данных](#X9abc9300d9a058600831bbb05447fb9262c7d53)
      1. [Система безопасности](#Xa917f4ed5d51864ca11cf22573c70e6ec72766e)
         1. [Система идентификации и авторизации](#c6725059f1407364b5c9cae3833a0b92)
         2. [Система команд](#Xc1c1262dce570bc4803a9f483f3a386ed911ed8)
   3. [Архитектура Клиент-Сервер](#Xa0586772ef29fad7756622e46f783864b8c012c)
      1. [Протокол](#X67ea7d5eb453dbad9455719058c90b7b14e8903)
         1. [Формат сообщений](#d601d1cdf1d51beb692dd2b3f43e5038)
      2. [Модель сервера](#c6c1ffce38a4e50926edc47be9e48348)
         1. [Система регистрации](#Xc850ed3e6a259b719cfb1b3efeb3b59f9656a85)
         2. [Модель обработки подключений](#b4a12ae5c66fc7edc3b4150477497e0b)
         3. [Модель обработки данных](#Xd12aaf51ed04bee8418830db3b36902589b5dfd)
      3. [Модель клиента](#X91d65b5ae2e8764997d4f7514be38b2aabb083e)
6. [Реализация](#X4cde7a818dee1253d109edf30ee93710d44d891)
   1. [Пользовательский интерфейс](#X558d2c820ddfdc823d565a7392ddbfcf62840f2)
      1. [PagesManager](#X9a43c218f3630d803fce6d17e6d5d5404e008d2)
      2. [NotifyManager](#Xe552fbc0ae864e0941e5bfa1b6a9a017d47a18b)
   2. [Журналирование](#Xa4191b0f7dc32a52a3d0f06dcc22efaa62a7d1e)
   3. [iiNPack](#X66124fed03ceccccc3b6bf6cfa286a83d1eb359)
   4. [Сервер](#X3023f6ed4a3d1cc84041d12b3447a4afb103868)
      1. [Потоки данных](#Xcafc44f60455987c5b9b7d08adc8fcd31b9cdda)
      2. [Драйвер базы данных](#X5a8cc993b99f89dc67fa8d5982c8ce89e51832e)
         1. [Криптография](#X58496945bada413ba24986e2b42b07db2c3609c)
      3. [Объект “сервер”](#e710c78f3c4dafb8ada8021c7b878969)
         1. [Линк](#X7f2423cdb03775d93a00a09cf76de6bbd81eecf)
         2. [Процессор подключений](#e3d4a75ceaa1be8d285dacee2bfe4bf1)
         3. [Сервис регистрации](#X5020b6bcb95eaeeeabde199aa6eaeccf4de2c15)
   5. [Клиент](#b99b6558c7cde67d398236bf888d33cb)
      1. [Потоки данных](#Xcafc44f60455987c5b9b7d08adc8fcd31b9cdda)
      2. [Service](#X555a2a2f1e0256a514a0a2743fbd72b9c67614d)
      3. [Менеджер групп страниц](#bb5ea4981f0e9a3be08addcddaef373b)
7. [Заключение](#e1c1d994d00e4e7a44fcd5d029d3548d)
8. [Послесловие](#X1912d82b0f165c6c394277cd2f75d16f96ae9cc)
9. [Список литиратуры](#bc5e99ae74607750379b50c9e11d0695)

# Введение

В наше временя оборот информации в бизнес сфере огромен, отцифрованно практически всё. Большая часть информации передаётся по средствам компьютерных сетей, в частности глобальнай – Интернет.

Технология Интернет приспособленна для передачи любого вида закодированной информации, чем пользуется в преимуществе большая часть бизнес отраслей для получения своей прибыли. Но во время развития Интернет появилась такая ниша как Взломщики, люди посягающии на не санкционированное получения доступа к ресурсам подключенных к Интернет. Вследствии начали стремительно развиваться технологии Криптографии , которые существующие еще со времён древнего Рима, и других методов борьбы с Взломщиками и не только.

Взломищакам могут быть интересны любые ресурсы: журналы Бухгалтерии, системы управления предприятеим и т.д.

Подходя к теме о ведении бизнеса в сфере ЧОП многие подозрительные личности могут быть заинтересованн в получении запланированных маршрутов инкосации, адресса жительства сотрудников… Для пресечения перечисленных выше махинаций можно полностью отказаться от ведения своей деятельности в Интернет.

Если это не выход – тогда необходимо развертывание системы защиты, чем мы и займемся в пределах данной работы. Также самым современным решением будет использование Blockchain – децентрализованный метод хранения данных, но данный вариант не будет рассматриваться, так как всё сильно усложняет и, скорее всего, по просту не приемлем.

Решение будет выполненно в несколько шагов:

1. Анализ структуры ЧОП.
2. Анализ проходимых процессов в ЧОП.
3. Описание средств разработки.
4. Составление базы данных ЧОП.
5. Разработка архитектуры приложения для взаимодействия с ЧОП.
6. Реализация архитектурных решений
7. Тестирование.

Отследить прогресс проекта и найти исходный код можно на github: https://github.com/siisgoo/siisty Так же данную курсовую можно прочитать оналйн по адресу: https://siisgoo.github.io/siisty Исходный текст курсовой расположен по адресу: https://github.com/siisgoo/siisty/tree/main/cursed Остальные ресурсы, связанные с документацие проекта: https://github.com/siisgoo/siisty/tree/main/docs

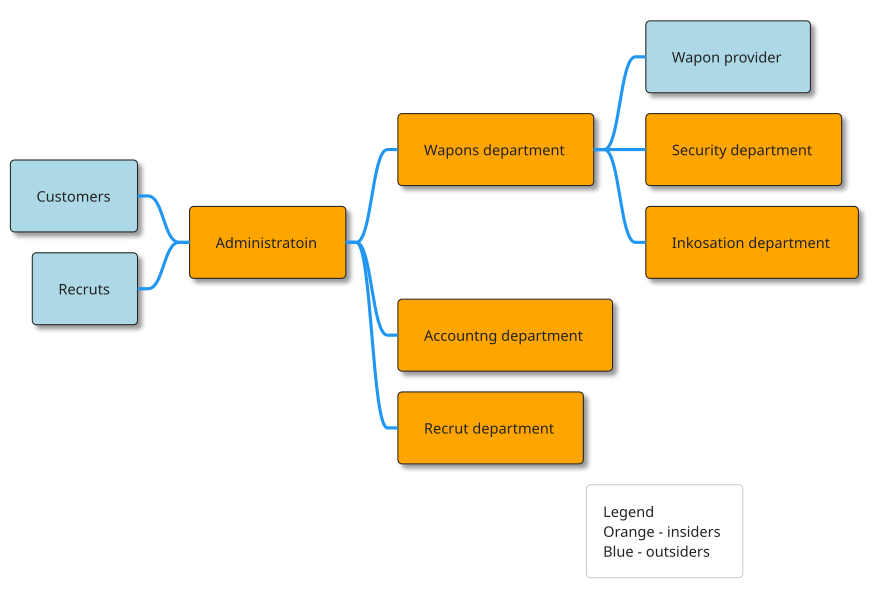
# Теоритическая часть

## Организация ЧОП

Охранное предприятие занимается охраной какой либо частной или государственной собственности.

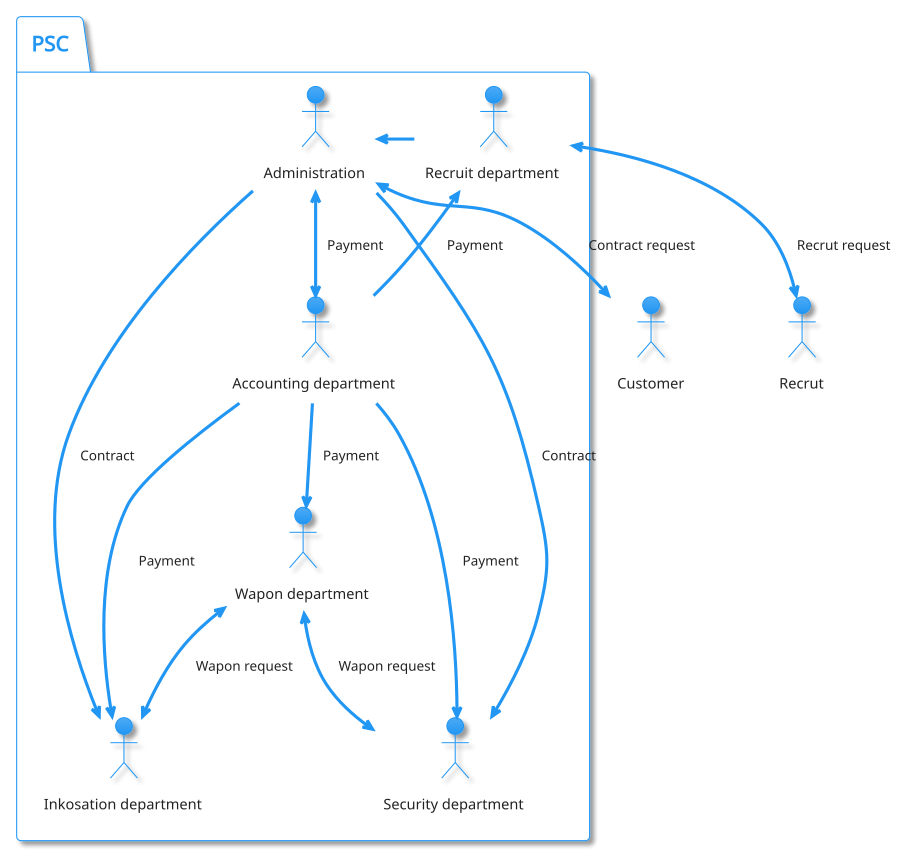
ЧОП, чаще всего, состоит из слудующих сущностей: - Администрация - Бухгалтерия - Отдел кадров - Отдел охраны - Отдел инкосации - Отдел вооружения

Если говорить о предоствляемых услугах более конкретно, то ЧОП для заказчика предлогает: - Охрану объекта - Охрану и транспортировка ценных бумаг или металов(инкосация)



Структура ЧОП

## Информационные потоки ЧОП

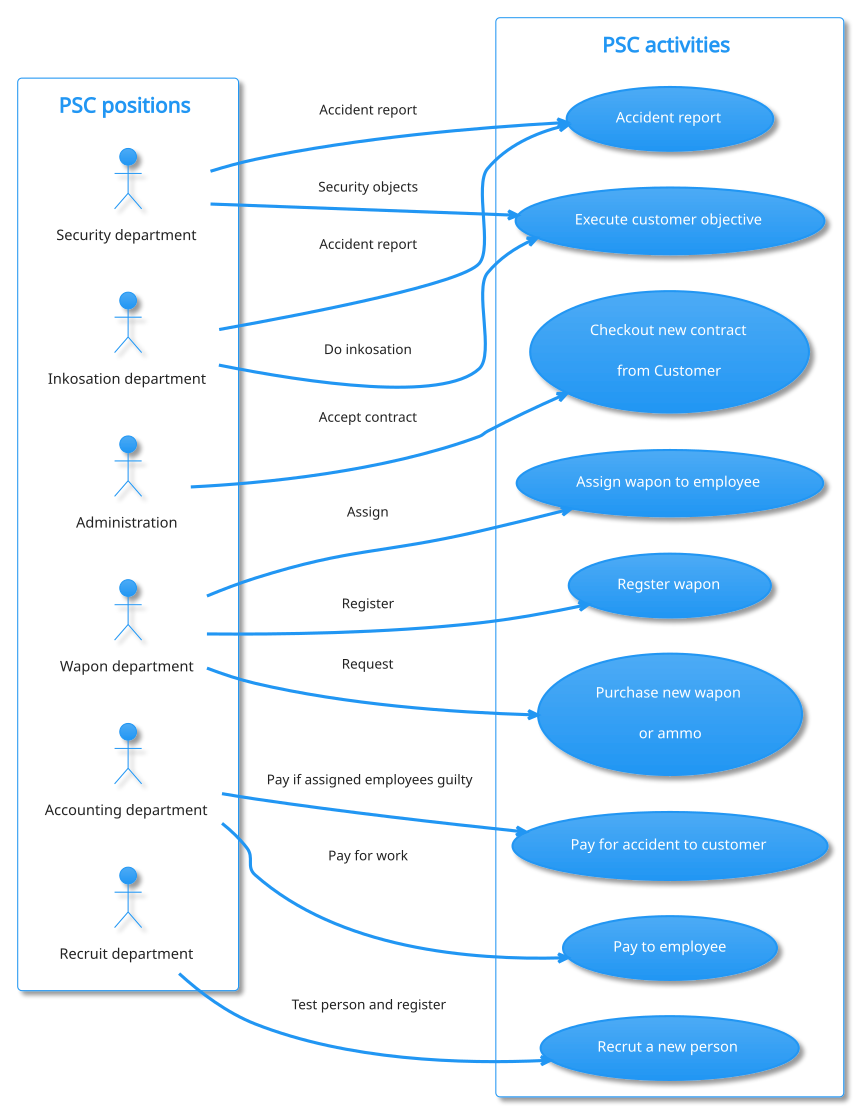
Проанализировав модель ЧОП, я выявил следующие информационные потоки: 

## Процессы внутри ЧОП

В стандартном ЧОП можно рассматривать следующие группы процессов: - Внутренние - Внешние

*Внутренние* - описывают действия, совершаемые в пределах огранизации, между сотрудниками. *Внешние* - действия с лицами из вне.

Лицами из вне, как было показанно на листинге выше, являются сущности “Заказчик” и “Рекрут”. > Под рекрутом понимается желающий вступить в организацию.



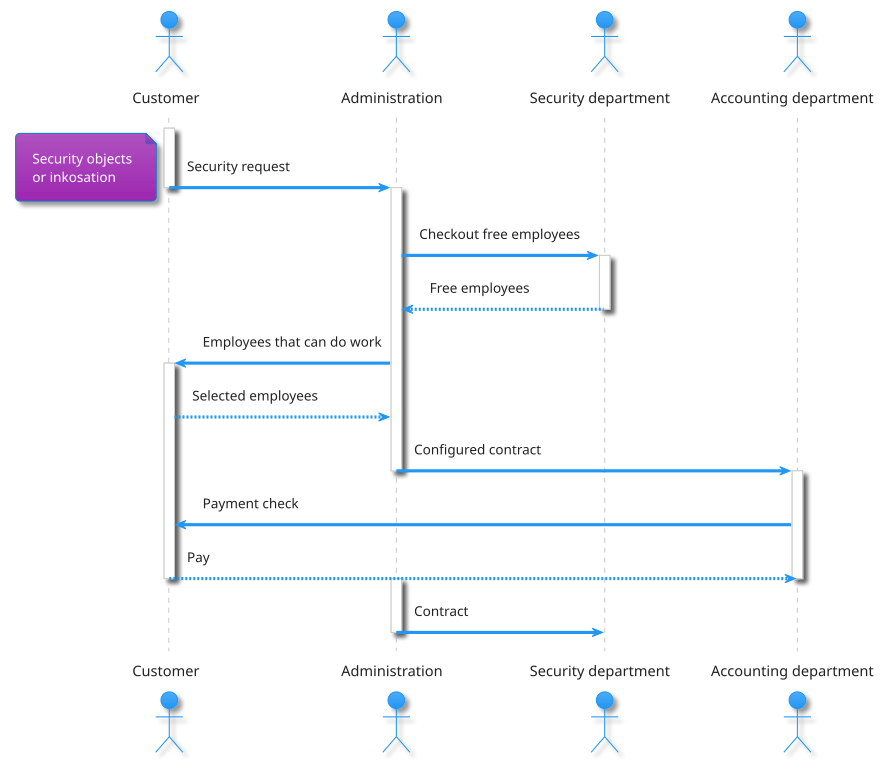
Процессы внутри ЧОП

### Описание процессов

В данном разделе будут более детально рассмотрены вышеперечисленные процессы.

#### Составление контракта

Вся прибыль ЧОП держится на составлении контрактов с заказчиками. Небольшой алгоритм по составлению и подтверждению контракта 1. Для составления контракта заказчик сначала выберает услугу, которую он хочет получить от ЧОП. 2. Затем ЧОП предлогает ему работников, которые могли бы выполнить этот заказ или сообщают о невозможности выполения данного контракта в данный момент. 3. После заказчик выберает из списка сотрудников подходящих или оставляет выбор за ЧОП. 4. И наконец - оплачивает заказ.

В общем виде это выглядит вот так: 

#### Закупка у поставщика оружия

Зачастую, для обороны объектов сотрудников ЧОП используется оружие, но для этого его нужно откуда-то брать. Закупкой будет промышлять Оружейный отдел.

Схема закупки: Диаграма закупки оружия

#### Прием на работу

Прием на работу состоит из этапов подачи заявки на рассмотрение кандидатуры, одобрения депортаментом рекрутинга и оповещения администрации.

#### Регистрация и выдача оружия

Регистрацией оружия занимается профилированый под это дело человек, эти детали будут опущены. Операция регистрации/выдачи оружия выполняется по фактору необходимости наличия оружия при службе на объекте сотрудника депертаментов инкосации и охраны(проверка на умения владения оружием проводится на этапе ркекрутинга), назначенным заказчиком.

#### Составление отчета об инцеденте на объекте

Обязательным предшествующем событием перед составлением отчета является доклад(звонок/письмо/оповещение) заказчика о нарушении сохранности объекта. После администрация выезжает на место и оценивает ущерб(не имеет финансов нанять специолистов), после чего составляется полный отчет и предоставляется заказчику. В случае не 0% вины сотрудника, выполняющего своим обязаности во время инцедента из его зарплаты вычетается сумма ущерба, если сумма ущерба превышает зарплату сотрудника, сумма ущерба вычетается из текущей и последующих X зарплат так, чтобы минимальная реальная зарплата сотрудника не была ниже минимальной установленной зарплаты.

#### Выплаты

Выплаты работникам: На баланс сотрудника переводится сумма, расчитанная через некоторые показатели сотрудника и параметров объекта, на котором он выполняет службу. Выплата совершается, если сотрудник был задействован в любом объекте.

Компенсации за нанесенный ущерб объекту охраны при не нулевом проценте вины сотрудника: Сумма ущерба переводится одной транзакцией на счет клиента. Денежная сумма выплаты либо сразу взымается из текущей зарплаты виновного сотрудника/виновных сотрудников, либо берется из резервов, если сумма выплаты превышает текущую зарплату сотрудника/сотрудников.

# Описание средстд разработки

В данном разделе будут рассмотрены средства разработки, используемые мной при создании ПС для ЧОП.

## Утилиты для разработки баз данных

### SQLite v3.35.5 stable



logo

SQLite — компактная встраиваемая СУБД. Исходный код библиотеки передан в общественное достояние. Данная СУБД работает в безсерверной конфигурации. Если сравнивать с другими СУБД, то в равных условиях запись SQLite осуществляет медленее на 20-30% чем другие СУБД, но чтение превосходит другие на 40-50%. SQLite не имеет привелегий, только систему авторизации, но это и не нужно в моем проекте, об этом будет сказано позже. > Примичание: > *Библиотека SQLite не будет использована в чистом виде, а в составе Qt v6.2.4*

### SQLiteBrowser v3.12.2



logo

Удобный FOSS браузер баз данных SQLite, использованный для отладки.

### DBVisualizer v12.1.8



logo

Проприетарная утилита для работы с разными СУБД, использован для генерации графа таблиц составленной базы данных ЧОП.

## Библиотеки

### Qt v6.2.4



logo

Qt - один из самых популярных и больших фреймворков c++ на рынке. Важная характеристика Qt - переносимость, т.к. я работаю на Linux. Содержит все необходимые компоненты для создания приложения любой сложности. Имеет одну из самых мощьных систем создания пользовательских интерфейсов, таких как QML+QtQuick и систему QWidget.

## Системы сборки и компиляторы

### CMake v3.22.3

logo Система сборки c++. Сборка проекта и передача более низкоуровневому средству.

### Ninja c1.10.2

 Еще одна система сборки, только уже более низкого уровня, чем CMake. Передача исходного кода на компиляцию.

### clang v13.0.1

 Компилятор семейства C.

## IDE

### QtCreator v6.0.2



logo

IDE от компинии The Qt Company, использованный только как средство отладки и создания скелетов форм пользовательского интерфейса.

### VIM v8.2



VIM logo

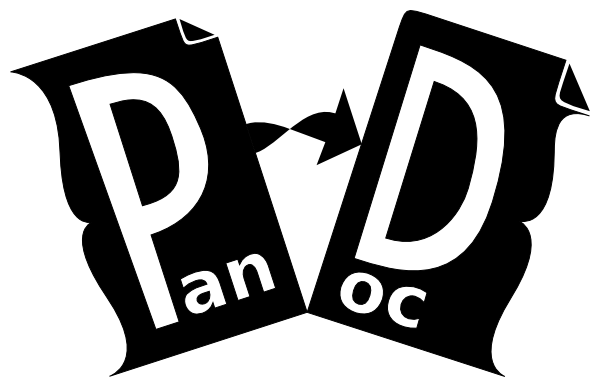
Моя любимая IDE, в своей основе так же прост как и каноничный “блокнот” в Windows, только с максимальной степенью кастомизации. Главное достоинство, по моему мнению, - это управление без использования мыши и возможность настройки управляющих комбинаций максимально удобно, что сокращает время на бесполезное перемещение рук по рабочему пространству.

#### Плагины

Plug 'https://github.com/xolox/vim-misc' " auto load  
Plug 'https://github.com/xolox/vim-session' " session manager  
Plug 'wakatime/vim-wakatime' " wakatime.com  
Plug 'SirVer/ultisnips' " snippets  
Plug 'https://github.com/honza/vim-snippets' " snippents files  
Plug 'https://github.com/pangloss/vim-javascript.git' " javascript extension  
Plug 'vim-airline/vim-airline' " status line  
Plug 'vim-airline/vim-airline-themes' " themes  
Plug 'vim-scripts/AfterColors.vim' " themes  
Plug 'rafi/awesome-vim-colorschemes' " themes  
Plug 'sonph/onehalf', { 'rtp': 'vim' } " theme  
Plug 'https://github.com/sjl/badwolf' " theme  
Plug 'https://github.com/joshdick/onedark.vim' " theme  
Plug 'https://github.com/plasticboy/vim-markdown' " markdown format support  
Plug 'ryanoasis/vim-devicons' " icons support  
Plug 'https://github.com/Yggdroot/indentLine'  
Plug 'ycm-core/YouCompleteMe' " code completer  
Plug 'rdnetto/YCM-Generator', { 'branch': 'stable'}  
Plug 'tpope/vim-commentary' " commentary shortcuts  
Plug 'm-pilia/vim-pkgbuild' " archlinux AUR PKGBUILD files support  
Plug 'https://github.com/tpope/vim-surround' " html-like tags handle utils  
Plug 'https://github.com/octol/vim-cpp-enhanced-highlight' " cpp syntax hilightinght  
Plug 'ctrlpvim/ctrlp.vim' " file finder  
Plug 'preservim/nerdtree' " dir tree dock  
Plug 'https://github.com/preservim/tagbar'  
Plug 'vim-scripts/bufkill.vim' " exit buffers without exiting vim  
Plug 'jreybert/vimagit' " git support  
Plug 'https://github.com/ap/vim-css-color' " HEX-colors hilighting  
Plug 'https://github.com/matze/vim-move' " code moving  
Plug 'https://github.com/junegunn/vim-easy-align' " fast text aligning  
Plug 'https://github.com/ervandew/supertab'  
Plug 'https://github.com/jiangmiao/auto-pairs' " completing pairs  
Plug 'https://github.com/rhysd/vim-clang-format' " auto formating  
Plug 'honza/vim-snippets' " set of snippets  
Plug 'https://github.com/godlygeek/tabular' " tab extender  
Plug 'https://github.com/junegunn/vim-easy-align'  
Plug 'https://github.com/fadein/vim-FIGlet' " figlet  
Plug 'https://github.com/scrooloose/syntastic'

## Разное

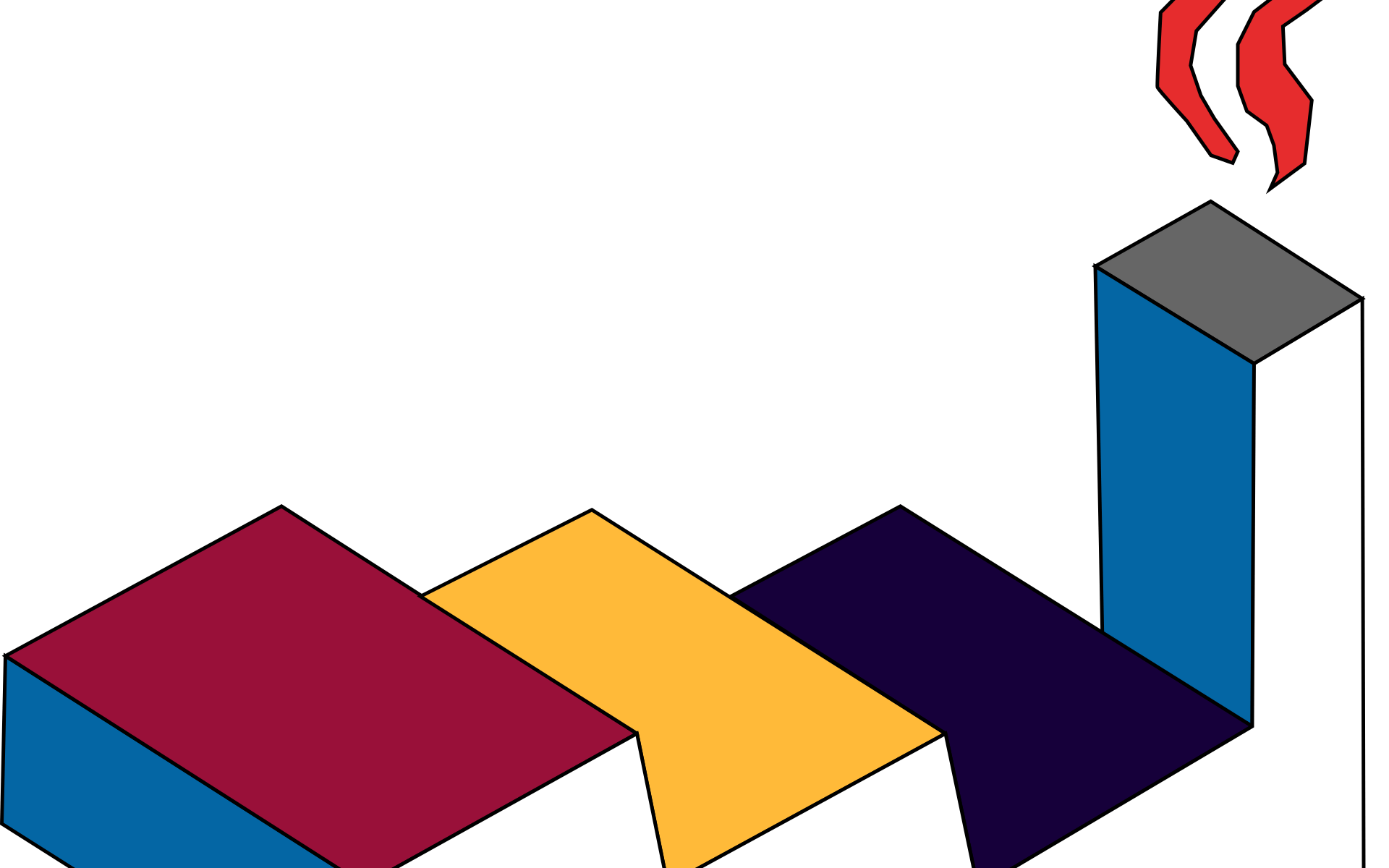
### Pandoc



logo

Работа была трансирована в други форматы с помощью данной утилиты.

### Plantuml v1.2021.16



logo

Средство создания UML диаграм. Использовано для визуализации объектов и просецссов.

### BASH v5.1.16



logo

Bourne Again Shell - интерпритатор, использован для автоматизации некотерых процессов.

Оглавления данной работы было автоматически сгенерерованно данным bash-скриптом:

function indexGen() {  
 out="${1/.md/}".indexed.md  
 cp "$1" "$out"  
 > index  
  
 tab='\t'  
 i=(-1 1)  
 prevLen=0  
 while read -r line; do  
 hash="$(md5sum <<< "$line" | cut -d ' ' -f 1)"  
 printf "<a id=\"%s\"/>\n\n%s\n\n" "$hash" "$line" > tmp  
 sed "/$line/ {  
 x  
 r tmp  
 }" "$out" > "${out}.tmp"  
 mv "${out}.tmp" "$out"  
 hdrLen=$(awk -F'#' '{print NF-1}' <<< "$line")  
 hdrTxt="${line//#/}"  
 (( hdrLen > 1 )) && for (( j=1; j<hdrLen; j++ )); do echo -en "$tab"; done  
 (( prevLen < hdrLen )) && i[$hdrLen]=1  
 printf "%d. [%s ](#%s)\n" ${i[$hdrLen]} "$hdrTxt" "$hash"  
 prevLen=$hdrLen  
 let i[$hdrLen]++  
 done <<< "$(grep --color=no -E "^#+ " "$1")" > index  
  
 cat "$out" > "$out".tmp  
 printf "# Содержание\n" > "$out"  
 cat index >> "$out"  
 cat "$out".tmp >> "$out"  
  
 rm index  
 rm tmp  
 rm "$out".tmp  
  
 mv "$out" "$2"  
}

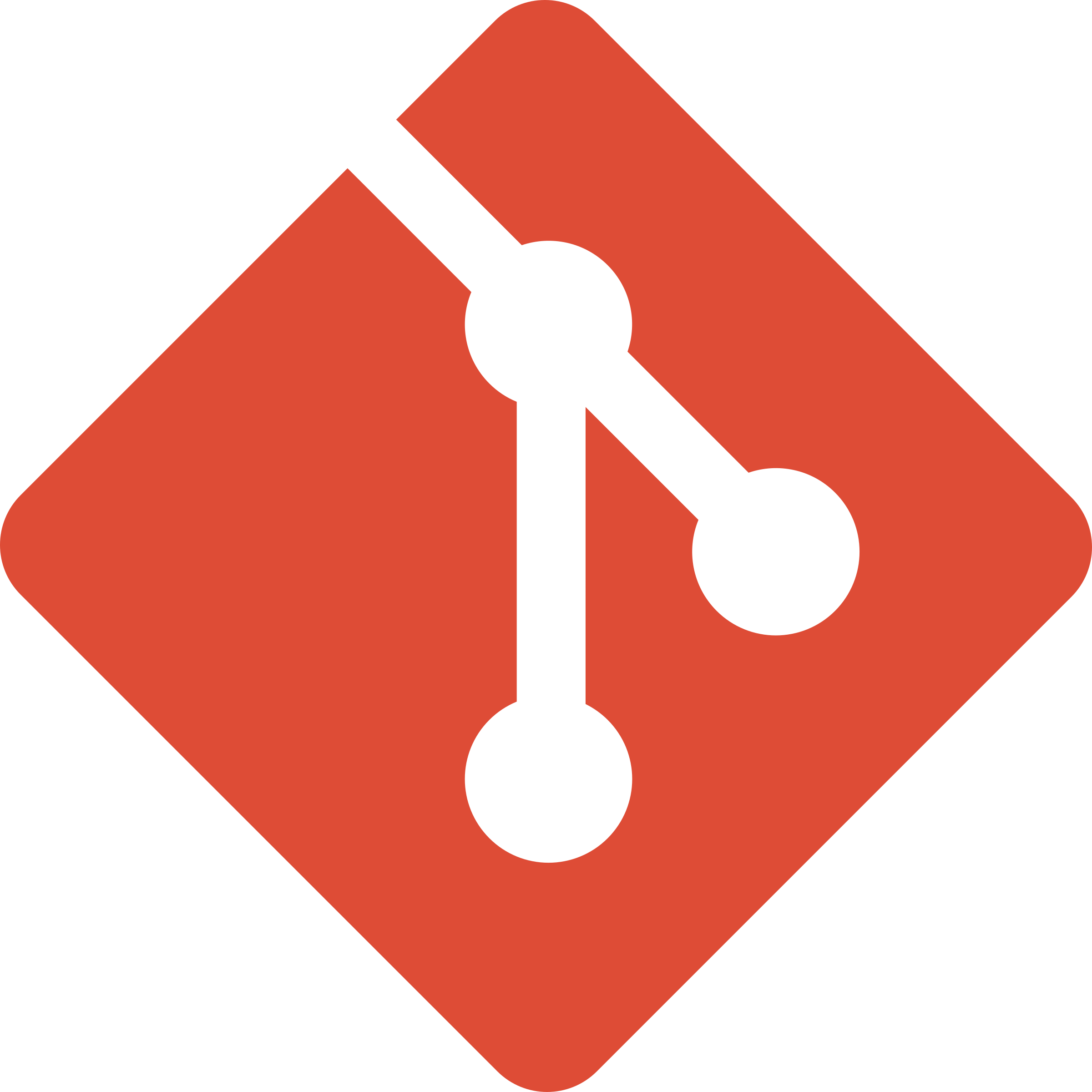
## Отладка

### Valgrind v3.18

Утилита профилирования и отладки программы, использовано в основном для обнаружения утечек памяти.

## VCS

### GIT v2.35.1

 GIT - система контроля версий, сомо о себе говрит. Использовался в основном для перенесения кода между машинами и как средство дистрибъюции.

#### GitHub

 Ресурс, на котором была размещена работа.

# Разработы базы данных ЧОП

База данных ЧОП в пределах данной работы - головная сущность, вокрук которой будет строится весь функционал. База данных будет существовать под управлением SQLite.

## Таблицы

В компанию, как известно, входит некоторое количество сотрудников, по этому, я создаю таблицу Users. Название выбрано таковым, потому что она будет содержать данные учетных записей сотрудников и клиентов ЧОП.

CREATE TABLE  
 "Users"("id" INTEGER NOT NULL UNIQUE,  
 "name" TEXT NOT NULL,  
 "entryDate" TEXT NOT NULL,  
 "role\_id" INTEGER NOT NULL,  
 "wapon\_id" INTEGER,  
 "email" TEXT UNIQUE,  
 "login" TEXT NOT NULL UNIQUE,  
 "password" BLOB NOT NULL UNIQUE,  
 "salt" BLOB NOT NULL,  
 "image" BLOB,  
 FOREIGN KEY("wapon\_id") REFERENCES "Wapons"("id")ON DELETE RESTRICT,  
 FOREIGN KEY("role\_id") REFERENCES "Roles"("id")ON DELETE RESTRICT,  
 PRIMARY KEY("id" AUTOINCREMENT))

Таблица содержит данные для идентификации:

* login
* password
* salt

Пароль не храниться в открытов виде, а зашифорован с использованием динамической соли по алгоритму “Prefered salt algorithm”, более подробно будет расмотрен далее.

Как видно, таблица Users зависит от таблиц Roles и Wapons, собственно вот они:

CREATE TABLE  
 "Roles"("id" INTEGER NOT NULL UNIQUE,  
 "name" TEXT NOT NULL UNIQUE,  
 "commands\_id" INTEGER NOT NULL,  
 "payMultipler" DECIMAL(10, 3) NOT NULL,  
 "payPeriod" INTEGER NOT NULL,  
 FOREIGN KEY("commands\_id") REFERENCES  
 "roleCommands"("role\_id")ON DELETE RESTRICT,  
 PRIMARY KEY("id"))

Роль определяет какие данные и соответственно команды можешь выполнять на сервере. Ссылается на таблицу roleComands - это SQL массив с ID команд, которые может выполнять пользователь с данной ролью, по поэтому я и отказался от других, более тяжелых СУБД, т.к. все необходимые действия делегируются на Сервер, что будет рассмотрено далее, от СУБД требуется только хранить данные и извлекать их. Связаная таблица roleCommands:

CREATE TABLE  
 "roleCommands"("role\_id" INTEGER NOT NULL,  
 "command\_id" INTEGER NOT NULL,  
 FOREIGN KEY("role\_id") REFERENCES  
 "Roles"("id")ON DELETE RESTRICT)

Таблица Wapons:

CREATE TABLE  
 "Wapons"("id" INTEGER NOT NULL UNIQUE,  
 "employee\_id" INTEGER UNIQUE,  
 "name" TEXT NOT NULL,  
 "ammo" INTEGER NOT NULL,  
 "price" DECIMAL(10, 3) NOT NULL,  
 "ammoPrice" DECIMAL(10, 3) NOT NULL,  
 "image" BLOB,  
 FOREIGN KEY("employee\_id") REFERENCES  
 "Users"("id")ON DELETE RESTRICT,  
 PRIMARY KEY("id"))

Каждая запись в таблице Wapons - это еденица зарегестрированного оружия, в полной мере описывающяя необходимые характеристики для ЧОП.

Так как организация имеет свои расходы и доходы, нам нужно сохранять эти данные. Таблица Accounting:

CREATE TABLE  
 "Accounting" ("id" INTEGER NOT NULL UNIQUE,  
 "accountingType\_id" INTEGER NOT NULL,  
 "pay" DECIMAL(10, 3) NOT NULL,  
 "date" TEXT NOT NULL,  
 FOREIGN KEY("accountingType\_id")  
 REFERENCES "AccountingType"("id") ON DELETE RESTRICT,  
 PRIMARY KEY("id") )

Зависит от таблицы AccountingType, описывающей какого рода транзакция была совершена.

CREATE TABLE  
 "AccountingType" ("id" INTEGER NOT NULL UNIQUE,  
 "name" TEXT NOT NULL UNIQUE,  
 PRIMARY KEY("id","name") )

ЧОП получает доход от контрактов, по этому была составлена таблица Contracts:

CREATE TABLE  
 "Contracts"("id" INTEGER NOT NULL UNIQUE,  
 "assignedEmployees\_id" INTEGER NOT NULL,  
 "customer\_id" INTEGER NOT NULL,  
 "objectType\_id" INTEGER NOT NULL,  
 "objectAddress" TEXT NOT NULL,  
 "objectWayPoint" TEXT, "date" TEXT NOT NULL,  
 "expirationDate" TEXT NOT NULL,  
 "weekends" TEXT NOT NULL,  
 FOREIGN KEY("customer\_id") REFERENCES  
 "Users"("id")ON DELETE RESTRICT,  
 PRIMARY KEY("id"),  
 FOREIGN KEY("assignedEmployees\_id") REFERENCES  
 "AssignedEmployees"("employee\_id") ON DELETE RESTRICT,  
 FOREIGN KEY("objectType\_id") REFERENCES  
 "objectType"("id")ON DELETE RESTRICT)

Запись в таблице Contracts это сделка вида, описанного в связаной таблице objetType. Таблица objetType:

CREATE TABLE  
 "objectType" ( "id" INTEGER NOT NULL UNIQUE,  
 "name" TEXT NOT NULL UNIQUE,  
 "price" DECIMAL(10, 3) NOT NULL,  
 PRIMARY KEY("id") )

Запись в данной таблице описывает объект контракта, где указывается базавая цена за период оплаты(payPeriod) исполнителя/исполнителей контракта(assignedEmployees\_id). Для привязки нескольких сотрудников, была создана еще одна таблица AssignedEmployees, являющейся массивом.

CREATE TABLE  
 "AssignedEmployees" ( "id" INTEGER NOT NULL,  
 "employee\_id" INTEGER NOT NULL,  
 "guiltyPercent" DECIMAL(10, 3) NOT NULL,  
 "usedAmmo" INTEGER,  
 FOREIGN KEY("employee\_id") REFERENCES  
 "Users"("id") ON DELETE RESTRICT,  
 PRIMARY KEY("id") )

Для создания контракта в данной таблице нужно только 2 поля - id, employee\_id. Так так во время исполнения может произойти какой то ицедент, то была создана таблица Accidents:

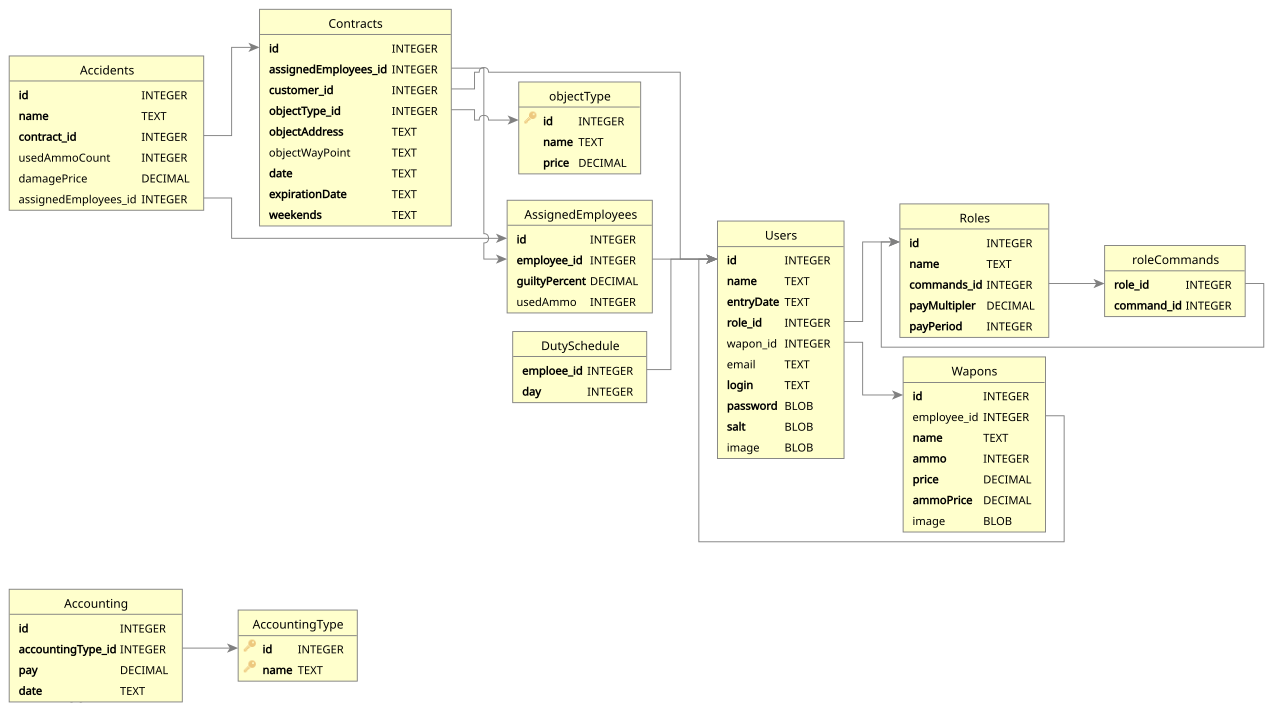
CREATE TABLE  
 "Accidents" ("id" INTEGER NOT NULL UNIQUE,  
 "name" TEXT NOT NULL,  
 "contract\_id"  
 INTEGER NOT NULL,  
 "usedAmmoCount" INTEGER,  
 "damagePrice" DECIMAL(10, 3),  
 "assignedEmployees\_id" INTEGER,  
 FOREIGN KEY("contract\_id") REFERENCES "Contracts"("id") ON DELETE RESTRICT,  
 FOREIGN KEY("assignedEmployees\_id") REFERENCES "AssignedEmployees"("id") ON DELETE RESTRICT,  
 PRIMARY KEY("id") )

Описывает проишествия, произошедшие во время исполнения контракта. Если у нас есть контракты, описывающие некую деятельность с учетом выходных и рамок начала и окончания службы, то можно было бы ускорить вычисление рабочего времени по дням с помощью препроцессинга данных из записи Contracts. Таблицей, в которую сохраняются транслированные данные является - DutySchedule:

CREATE TABLE  
 "DutySchedule" ( "emploee\_id" INTEGER NOT NULL,  
 "day" INTEGER NOT NULL  
 FOREIGN KEY("emploee\_id") REFERENCES "Users"("id") ON DELETE RESTRICT )

day - это 64х битная цыфра со знаком в формате UNIX time(secs since epoch).

## Визуализация базы данных



Datbase visualization

# Разработка архитектуры приложения

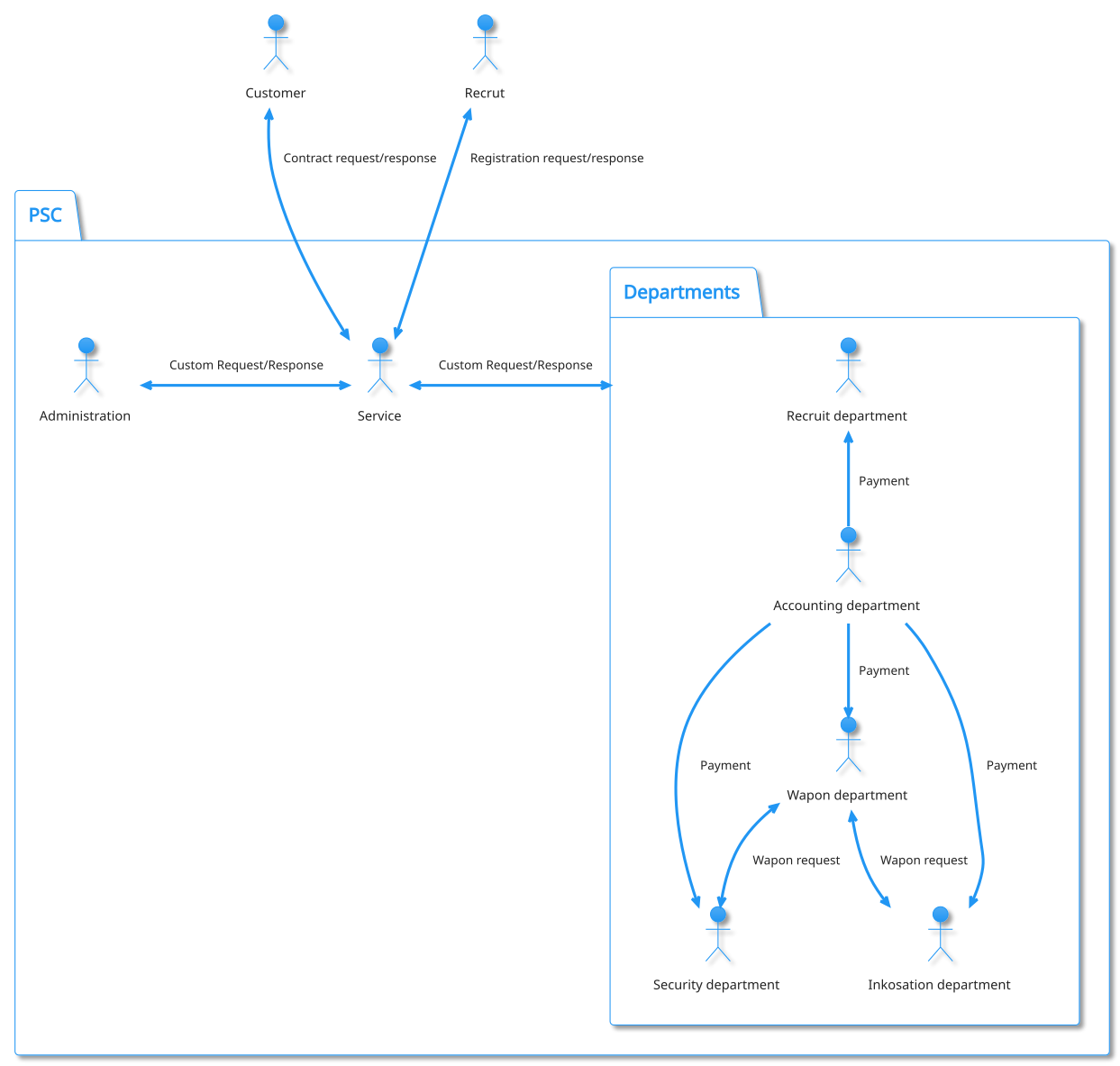
Приложение для взаимодействия с моделью ЧОП, построенной ранее, должно предоставлять функционал для реализации всех описанных процессов взаимодействия с моделью ЧОП.

## Основа

Основной функцией приложения, как понятно из темы курсовой работы, будет обеспечение безопасности данных. Исходя из этого в голову приходит идея организовать клиент-серверную архитектуру приложения, но это не главная причина почему выбрана такая архитектура. Основная причина - необходимость принимать заказы от клиентов, предоставить им функционал для удобного взаимодействия с персоналом ЧОП, но по большей части он будет взаимодействовать с клавиатурой. И для персонала ЧОП тоже будет намного удобнее и быстрее использовать унифицированные методы итерации с базой данных и самой организацией.

## Информационные потоки ЧОП

## Информационные потоки ЧОП

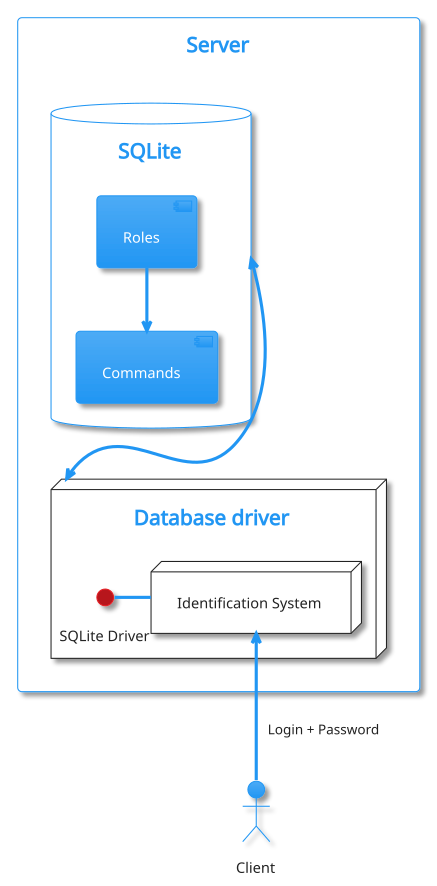


PSC with server info flows

## База данных

Как было сказано ранее, СУБД не будет управлять системой привелегий, этим будет заниматься другой код. Система, которую я разработал основана на ролях, так же как и в обычных “умных” СУБД, к которым привязано некоторое количество возможных к исполнению команд.

База данных должна будет сама себя обслуживать и проверять целостность обязательный записей, который вшиты в исходный код.



### Система безопасности

Для обеспечения более высокого уровня защиты команды и роли будут вшиты в программу. При каждом старте будет проверятся валидность первоначальных данных и имеющихся на данный момент в базе данных. Также, как мера безопасности к ПС будут предложены хеш суммы исполняемых файлов.

#### Система идентификации и авторизации

При регистрации нового пользователя создается новая запись в таблице Users, все поля, короме поля password сохраняются в неизменнов виде. password сохраняется в захешированном виде при использовании хэш функции sha512 с использование динамической соли.

Алгоритм создания соли:

1. Подсчет количества символов разных типов в пароле.
2. Выявление слабых сторон пароля.
3. Формирование алфавита для создания соли.
4. С помощью “strog random” функции выборка символов из составленого алфавита.
5. Запись результата.

Алгоритм хэширования пароля:

1. Генерация соли.
2. Взятие первичного хеша с пароля.
3. Итерационно выбрать по байтно методом XOR младших битов первичного хеша и переданного пароля места вставки соли.
4. Хеширование получившейся строки.
5. Запись результата.

Данный алгорит и выбраная хэш функция ограничивают максимальную длину пароля до 64 символов. Длина соли была вабрана 32 байтная. Соль сохраняется вместе с паролем, чтобы обеспечить возможность идентификации.

Для идентификации проводится сравнивание значение пароля из базы данных с переданным в функцию шифрования с солью данной записи пользователя пароля.

#### Система команд

Проанализировав информационные потоки и требования ЧОП я составил 22 команды, в исходном коде объявление команд выглядит вот так:

// id, name, executor  
#define COMMANDS\_MAP(XX) \  
 XX( 0, MAKE\_CONTRACT, exec\_make\_contract ) \  
 XX( 1, MAKE\_DUTY\_SCHEDULE, exec\_make\_duty\_schedule ) \  
 XX( 2, REGISTER\_ACCIDENT, exec\_register\_accident ) \  
 XX( 3, REGISTER\_EMPLOYEE, exec\_register\_employee ) \  
 XX( 4, REGISTER\_CUSTOMER, exec\_register\_customer ) \  
 XX( 5, REGISTER\_OBJECT\_TYPE, exec\_register\_object\_type) \  
 XX( 6, REGISTER\_WAPON, exec\_register\_wapon ) \  
 XX( 7, ASSIGN\_WAPON, exec\_assign\_wapon ) \  
 XX( 8, PAY\_AMMO, exec\_pay\_ammo ) \  
 XX( 9, PAY\_EMPLOYEE, exec\_pay\_employee ) \  
 XX( 10, PAY\_ACCIDENT, exec\_pay\_accident ) \  
 XX( 11, EDIT\_OBJECT\_TYPE, exec\_edit\_object\_type ) \  
 XX( 12, UPDATE\_ROLE, exec\_update\_role ) \  
 XX( 13, GET\_USER\_INFO, exec\_get\_user\_info ) \  
 XX( 14, GET\_ACCIDENT\_DETAILS, exec\_get\_accident\_details) \  
 XX( 15, GET\_ACCOUNTING\_ENTRY, exec\_get\_accounting\_entry) \  
 XX( 16, GET\_OBJECT\_DETAILS, exec\_get\_object\_detalils) \  
 XX( 17, GET\_ROLE\_DETAILS, exec\_get\_role\_details ) \  
 XX( 18, GET\_WAPON\_DETAILS, exec\_get\_wapon\_details ) \  
 XX( 19, GET\_DUTY\_SCHEDULE, exec\_get\_duty\_schedule ) \  
 XX( 20, CREATE\_TABLE, exec\_create\_table ) \  
 XX( 21, IDENTIFY, exec\_identify ) \

## Архитектура Клиент-Сервер

Архитектура Клиен-Сервер была выбрана не только для обеспечения безопасности базы данных, но и для реализвации удаленного унифицированного доступа к услугам и возможностям ЧОП. Так для сотрудников упрощается способ комуникации с начальством, подчиненными. А заказчики смогут после регистрации в системе могут удаленно составить контракт по охране.

### Протокол

Общение клиентов с сервером будет осуществяться по протоколу TCP опиционально с использованием SSL/TLS(опциональность введена для более удобного тестирования).

#### Формат сообщений

Поверх формата TCP пакета я ввел для ПС дополнительные поля.

Состоит из заголовка и нагрузки.

*Заголовок* Длина заголовка - 176 бит или 22 байта. Кончно, можно было бы спокойно использовать все 192 бита(3 машинных слова), чтобы выравнять заголовок, но это не сильно повлияет на какой либо процесс.

| Байты | Название | Описание |
| --- | --- | --- |
| 0x0 - 0x3 | Size | Суммарный размер сообщения(байт) |
| 0x4 - 0x11 | ServerStamp | Время оправки сообщения сервером |
| 0x12 - 0x19 | ClientStamp | Время отправки сообщения клиентом |
| 0x20 - 0x21 | Type | Тип сообщения |
| 0x22 - 0x23 | Format | Формат назгрузки сообщения |

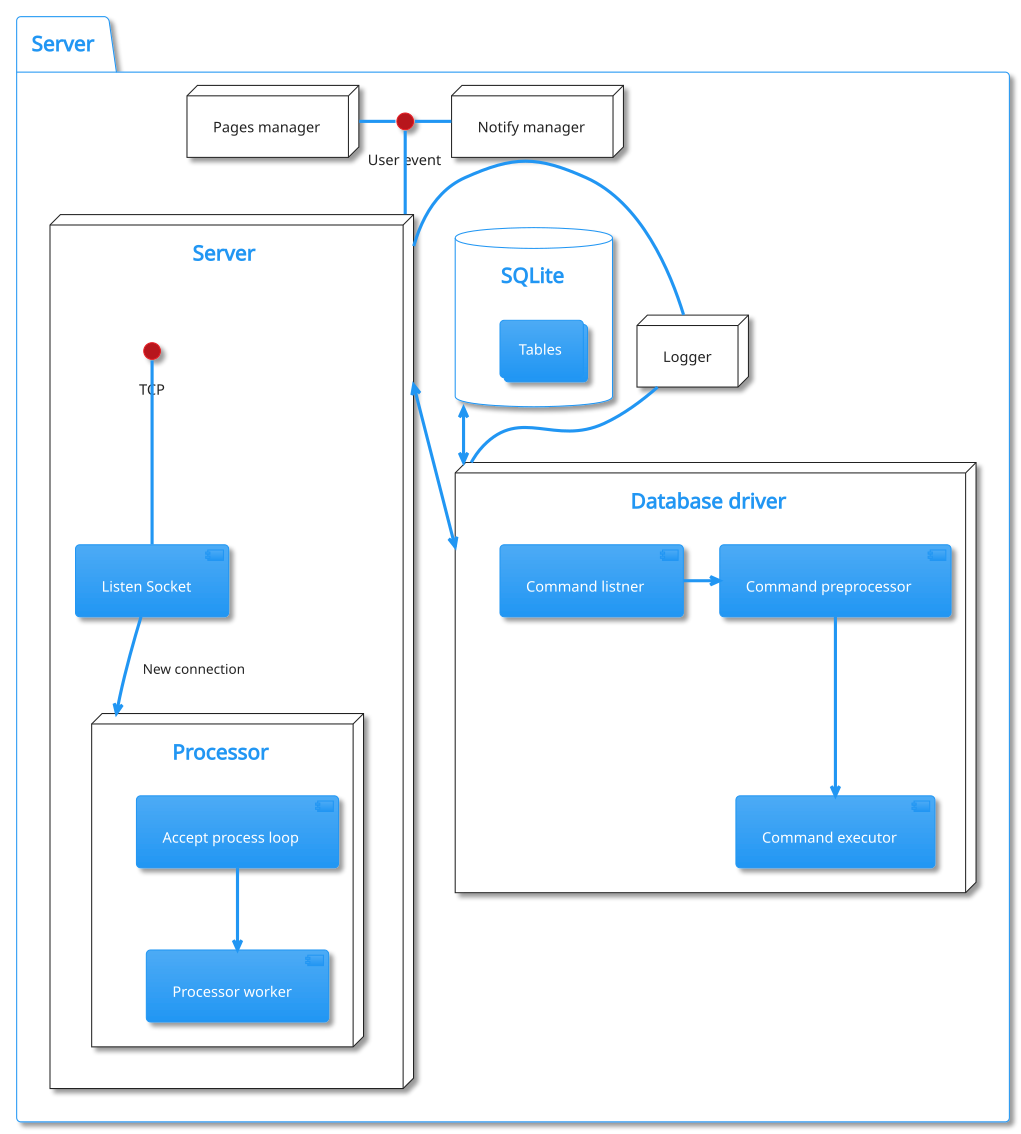
Таким образом, поле Size позволяет считывать последовательность байт в единое сообщение из нескольких пакетов TCP. Поля xxxStamp используются как некий идентификатор сообщения. Тип пакета и формат нагрузки определяют собственно тип нагрузки и сообщения в целом.

### Модель сервера

Основные задачи сервера:

* обслуживании базы данных
* реализации механизма идентификации и авторизации, также регистрации
* принятие входящих запросов на подключение
* работа в режиме Запрос => Ответ
* Мониторин протекающих процессов

Также, развертывание сервера будет означать создание первого пользователя, для управления организационными процессами ЧОП. Следовательно у сервера должен быть интерфейс для создания пользователей, как и у клиента.



Server abstract model

#### Система регистрации

Идентификацие и авторизацией пользоваетлей занимается база данных, а регистрацией новых пользователей типа “Заказчик” и “Рекрут” будет сервер т.к. процесс регистрации будет содержать помимо создания новой записи в таблице Users, но и подтверждение личности через электронную почьту.

#### Модель обработки подключений

Server processing model

Server processing model

#### Модель обработки данных

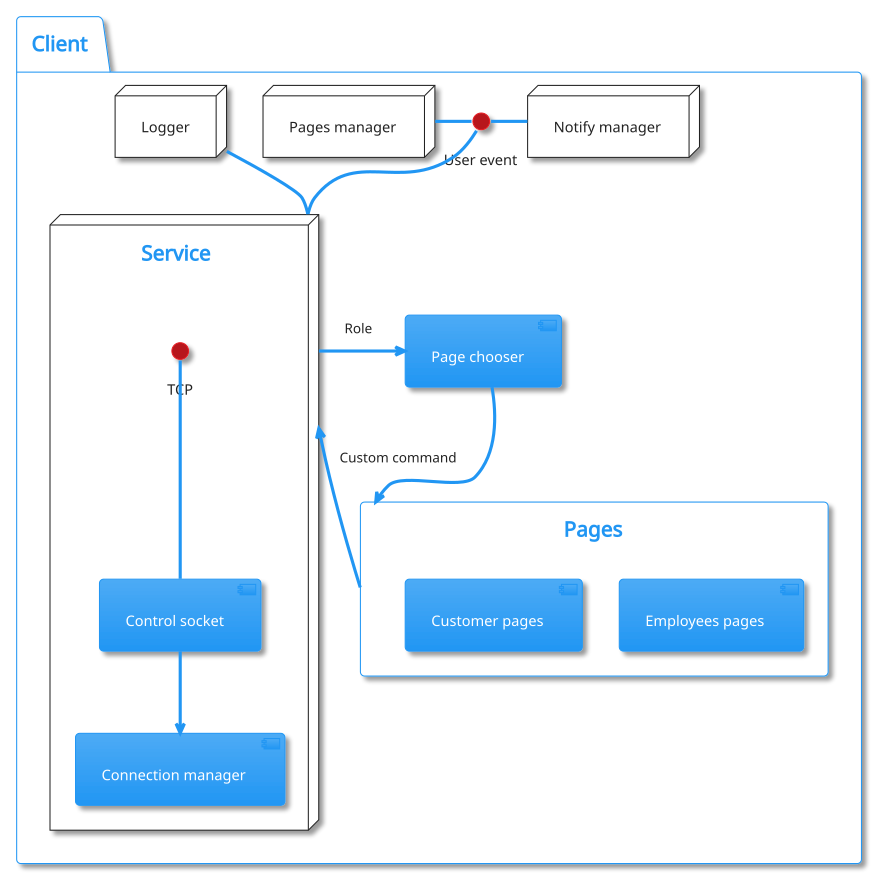
Server processing data model

Server processing data model

### Модель клиента

Клиент - это программа, предоставляющая некотерый функционал, в который в любом случае входит пользовательский интерфейс, после успешного входа в систему, функционал будет разниться в зависимости от роли вошедшего в систему.

Все что требуется от клиента - это показывать пользовательский интерфейс, предоставить некотерый метод для входа в систему сервера и отправлять команды серверу, и ждать ответа. В частности, когда пользватель, который воспользовался клиентом является заказчик - клиент должен предоставить метод регистрации в систему.



Client abstract mode

# Реализация

Для реализации проекта был выбран язык C++ и фреймвокр Qt6. Выбор лег в сторону C++ только потому что это мной наиболее изученный язык, на много проще было бо всё реализовать на nodejs с express, а Qt - это лучшый кросплатформенный фреймворк лично для меня, он упростил разработку процентов на 60% и сократил зависимости проекта на 100%(используются только библиотеки из пакета Qt). Сборкой проекта будет заниматься CMake с делегацией компиляции Ninja, C++ компилятор - clang. Для отладки буду использовать GDB в составе QtCreator и Valgrind. Создания шаблонов форм так же будет осуществляться в QtCreator в модуле QtDisigner.

При разработке будет применен модульный принцып програмирования - отдельным модуль - динамическая бибилиотека или библиотека типа MODULE, загружаемая dlopen-like методом, т.к. пользовательский интерфейс для клиента тоже будет находиться в подмодулях основного приложения клиента и таким образом можно будет сократить кол-во использования памяти.

Список модулей: - Crypto - Database - iiServer - Logger - Math - Network - PagesManager - NotifyManager - Widgets

Использование динамических библиотек обусловленно уменьшением времени компиляции и идеологией разработки на Linux. Большая чать приложния будет реализованна с использованием ООП, только модуль Database будет наполовин использвать чистое функциональное c-style програмирование.

Т.к. разработка ведется в передлах курсовой работы, релиз приложения не будет выходить дальше Pre-alpha и для работы на платформе Linux.

## Пользовательский интерфейс

За отрисовку пользовательского интерфейса будет отвечать модуль Qt - QtGUI.

Пользовательский интерфейс будут составлять 5 основных сущностей:

* Бар состояния
* Меню бар
* Тул бар
* Пейджер(PagesManager)
* Менеджер уведомлений(NotifyManager)

### PagesManager

Данный класс отвечает за агрегацию “страниц” GUI и их переключение. > Страница - это отдельный виджет(QWidget) или объект-наследник. Сам класс PagesManager - это тоже виджет, наследуемый от QFrame(он тоже наследник QWidget).

Содержит в себе именованный массив страниц:

struct Page {  
 QWidget \* widget = nullptr;  
 int navId = -1;  
 QVector<QString> edges = {};  
};  
  
QMap<QString, Page> \_pages;

Как видно из данного участка кода, одна страница может быть связана с другими по имени(можно было бы связывать напрямую с другим объектом Page, но выбранный мной подход более прост в реализации). Страница добавляется в общий массив методом:

void  
PagesManager::addPage(const QString& name, QWidget\* wp,  
 const QVector<QString>& edges)  
{  
 if (!wp) {  
 throw QString("empty widgt passed");  
 }  
  
 wp->setObjectName(name);  
 \_pages[name] = {wp, -1, edges};  
 \_view->addWidget(wp);  
}

view - это QStacketWidget - место размещения страниц и является основной сущностью пейджера. Также, класс содержит перегруженный метод добавления корневой страницы. Она необходима для построения путей к страницы.

За построение пути отвечает агрегируемый класс PagePathFrame, наследуемый от QFrame, является второй основной сущность пейджера. Основным методом класса явяется методод:

void  
PagePathFrame::changePath(const QVector<QString>& path)  
{  
 reset();  
 for (auto node : path) {  
 QWidget \* lbl = new QLabel(\_delemiter, this);  
 lbl->setFont(QFont("Jet Brains Mono", 9));  
 lbl->setSizePolicy(QSizePolicy::Maximum, QSizePolicy::Maximum);  
 \_layout->addWidget(lbl);  
 ClickableLabel \* clbl = new ClickableLabel(node, this);  
 clbl->setSizePolicy(QSizePolicy::Maximum, QSizePolicy::Maximum);  
 clbl->setCursor(QCursor(Qt::CursorShape::PointingHandCursor));  
 clbl->setStyleSheet("color: #b78620");  
 connect(clbl, &ClickableLabel::clicked, [this, node] { Q\_EMIT clicked(node); });  
 \_layout->addWidget(clbl);  
 }  
 adjustSize();  
}

Путь к страницы вычисляется в методе пейджера:

QVector<QString>  
PagesManager::pagePath(const QString& page)  
{  
 QVector<QString> path { page };  
 QString search = page;  
 bool done = false;  
 int tries = 0;  
  
 while (!done && tries < \_pages.size() + 1) {  
 for (auto i : \_pages) {  
 if (search == \_root) {  
 done = true;  
 break;  
 }  
 for (auto node : i.edges) {  
 if (node == search) {  
 search = i.widget->objectName();  
 path.push\_front(search);  
 // exit outer?  
 break;  
 }  
 }  
 }  
 tries++;  
 }  
 return path;  
}

Не самый эффективный метод, можно было бы хранить сразу все возможные пути в массиве. Метод основан на поиск в глубину в графе, если я не ошибаюсь.

Как видно, каждый элемент фрейма, при нажатии генерирует сигнал о нажатии, для отправки PagesManager. При получении пейджер меняет страницу.

Третьей сущностью явялется навигационная панель - NavWidget. Содержит связаные с данной страницей ссылки в виде кнопок. Гланый метод добавления навигационного меню:

int  
NavWidget::addNav(const QVector<QString>& pages, bool createBack)  
{  
 NavFrame \* fnav = new NavFrame(pages, createBack, this);  
 this->addWidget(fnav);  
 connect(fnav, SIGNAL(clicked(QString)), this, SIGNAL(clicked(QString)));  
 adjustSize();  
 return this->count()-1;  
}

Так же при нажатии меняет страницу.

Для того, чтобы связать страницы и навигационное меню используется рекурсивный метод:

void  
PagesManager::bindPages(const QString& parent, const QVector<QString>& childs)  
{  
 int nid = \_nav->addNav(childs, parent != \_root);  
 \_pages[parent].navId = nid;  
 for (auto child : childs) {  
 if (\_pages[child].edges.length() > 0) {  
 bindPages(child, \_pages[child].edges);  
 } else {  
 \_pages[child].navId = nid;  
 }  
 }  
}  
  
void  
PagesManager::finalize()  
{  
 if (\_root == QString()) {  
 throw "Cannot finalize PagesManager without root page";  
 }  
 bindPages(\_root, \_pages[\_root].edges);  
 changePage(\_root);  
}

### NotifyManager

Сущность выполняющая динамическое позиционирование всплывающих уведомлений разного типа. В своей основе - это лейаут поверх всего рабочего пространства приложения и процессор, обрабатывающий запросы на создание новых уведомлений и управления существующими, уведомление - это разновидность класса NotifyItem. Главная причина почему этот класс существует - возможность создавать thread-safe уведомления из любого потока. Т. к. любой виджет вне потока существования виджета-родителя не будет напрямую связан с его петлей событий. Как известно, объект начинает существовать там, где он был создан с помощью оператора new. Поэтому для передачи на обработку NotifyManager’у используюся фабрики NotifyItemFactory.

Таким образом, мне получилось создать простой интерфейс для создания всплывающих уведомлей:

void setItemPropery(int uid, const QByteArray& name, const QVariant& value);  
void createNotifyItem(NotifyItemFactory\*, int& uid);

Обращение к созданому уведомлению происходит по выделеному uid, который возвращает createNotifyItem.

Все существующие уведомления храняться в именованом массиве:

QMap<int, NotifyItem\*> \_items;

Но лучше было бы хранить их в связаном списке, так бы сохронялся их порядок появления и упрощалась перегрупировка с возможностью более быстрой обработки анимации движения появления и скрытия, пока что единственная доступная анимация - изменение прозрачности.

Существующие уведомления перегрупировываются при заверении, изменении размера окна. При изменении максимальной ширины уведобления

Для управления поп-апами из вне используется система QProperty в мета объекте.

Класс является *thread-safe*.

## Журналирование

Логер реализован по простой модели работы в отдельном потоке. Использует систмему уровней уведомления, для фильтрации вывода. Для вывода в лог используется система событий Qt. Для фильтрации доступны уровни:

enum LoggingLevel {  
 Trace = 0,  
 Debug,  
 Info,  
 Warning,  
 Error,  
 Fatal,  
};

## iiNPack

Класс реализующий протокол описанный выше, включает методы упаковки данных для упрощения процесса передачи данных.

Главные определения в классе:

enum PacketType : quint8  
{  
 AUTORIZATION\_REQUEST,  
 REQUEST,  
 RESPONSE,  
 ERROR\_MESSAGE,  
};  
  
enum PacketLoadType : quint8  
{  
 JSON = 0,  
 XML,  
 RAW,  
};  
  
// TODO create error map  
enum ResponseError : quint8  
{  
 ACCESS\_DENIED = 0,  
 NETWORK\_ERROR,  
 REQUEST\_ERROR,  
 UNSUPPORTED\_FORMAT,  
 UNSUPPORTED\_TYPE,  
 PARSE\_ERROR,  
};  
  
struct Header  
{  
/\* 0x0 - 0x3 \*/ quint32 Size; /\* Overall packet load size in bytes \*/  
/\* 0x4 - 0x11 \*/ qint64 ServerStamp; /\* Send time on server; using QDateTime SecsSinceEpoch \*/  
/\* 0x12 - 0x19 \*/ qint64 ClientStamp; /\* Send time on client; using QDateTime SecsSinceEpoch \*/  
/\* 0x20 - 0x21 \*/ quint8 PacketType; /\* Type of packet; enum class PacketType \*/  
/\* 0x22 - 0x23 \*/ quint8 PacketLoadType; /\* Load format, see enum class PacketLoadType \*/  
};

## Сервер

В данном подразделе будут описанны пути реализации приложения-сервера.

### Потоки данных

**TODO**

### Драйвер базы данных

По сути, драйвер базы данных - это декоратор QSqlDatabase класса преднозначенный для работы в отдельном потоке. По мимо доступа к базе данных, класс реализует систему идентификации и атунтификации.

Т.к. драйвер будет работать в отдельном потоке, чтобы не тормозить дргие потоки, из которых вызываются методы драйвера, он имеет в себе очередь команд на выполнение:

struct DatabaseCmd {  
 int executorRole;  
 QJsonObject data;  
 DriverAssistant \* waiter;  
};  
  
DatabaseCmd cmd = \_cmdQueue.dequeue();

Т.к. передача команды в очередь на исполнение драйверу реализована с помощью сигналов, данный объект DatabaseCmd необходимо ввести в систему мета компиляции:

Q\_DECLARE\_METATYPE(Database::DatabaseCmd)

Обработка происходит в такой незамысловатой петле:

void  
Driver::worker()  
{  
 QMutexLocker lock(&\_queueMtx);  
 if (\_cmdQueue.length()) {  
 DatabaseCmd cmd = \_cmdQueue.dequeue();  
 this->executeCommand((RoleId)cmd.executorRole, cmd.data, cmd.waiter);  
 }  
  
 if (\_running) {  
 QTimer::singleShot((\_cmdQueue.length() ? 100, 0), this, SLOT(worker()));  
 }  
}

И наконец, функция которая реализует механизм аутентификации:

void  
Driver::executeCommand(Database::RoleId role, QJsonObject obj, DriverAssistant\* waiter) {  
 if (!waiter) {  
 throw QString("Driver::" + QString(\_\_func\_\_) + ": Null waiter passed!");  
 }  
  
 if (role != ROLE\_AUTO) {  
 if (role > ROLES\_COUNT || role < (RoleId)0) {  
 waiter->Failed(CmdError(AccessDenied, "Invalid Role ID passed"));  
 return;  
 }  
 }  
  
 int command\_n;  
 if (auto val = obj["command"]; val.isDouble()) {  
 command\_n = val.toInt();  
 } else {  
 waiter->Failed(CmdError(InvalidCommand, "No command passed"));  
 return;  
 }  
  
 if (command\_n > COMMANDS\_COUNT || command\_n < 0) {  
 waiter->Failed(CmdError(InvalidCommand, "Command not exists"));  
 return;  
 }  
  
 if (role != ROLE\_AUTO) {  
 // check permission for execute command  
 if (!\_roles[role].commands.contains(command\_n)) {  
 waiter->Failed(CmdError(AccessDenied, "You not have access to execute this command"));  
 return;  
 }  
 }  
  
 if (auto val = obj["arg"]; val.isObject()) {  
 QJsonObject target = val.toObject();  
 auto cmd = \_commands[command\_n];  
 CmdError rc = cmd.executor(target);  
 if (rc.Ok()) {  
 waiter->Success(target);  
 } else {  
 waiter->Failed(rc);  
 }  
 } else {  
 waiter->Failed(CmdError(InvalidParam, "No parameters passed"));  
 }  
}

DriverAssistant - это отдельный класс, который уведомляет объект, который ожидает данные от драйвера, содержить два метода и два сигнала, описывающих успешное или не успешное завершение выполнение команды.

Роли и команды храняться статично в объекте драйвера в объектах:

struct role\_set {  
 int id; // equal to index  
 const char \* name;  
 QVector<CommandId> commands;  
};  
  
struct command\_set {  
 int id; // equal to index  
 const char \* name;  
 command\_exec\_t executor;  
 bool sendback;  
};

Команды храняться в массиве, роли - именованом массиве. И заполняются в конструкторе драйвера таким образом:

#define XX(num, name, query) { TABLE\_ ##name, QUOTE(name), query },  
 \_tables = { TABLES\_MAP(XX) };  
#undef XX  
#define XX(id, val, cmds) { ROLE\_ ##val, { ROLE\_ ##val, QUOTE(val), cmds } },  
 \_roles = { ROLE\_MAP(XX) };  
#undef XX  
#define XX(id, n, exe) { CMD\_ ##n, QUOTE(n), exe },  
 \_commands = { COMMANDS\_MAP(XX) };  
#undef XX

Данные ролей и команд:

#define AdminCommands { \  
 CMD\_EDIT\_OBJECT\_TYPE, \  
 CMD\_MAKE\_DUTY\_SCHEDULE, \  
 CMD\_REGISTER\_ACCIDENT, \  
 CMD\_REGISTER\_EMPLOYEE, \  
 CMD\_GET\_USER\_INFO, \  
 CMD\_REGISTER\_OBJECT\_TYPE, \  
 CMD\_UPDATE\_ROLE, \  
 CMD\_GET\_ROLE\_DETAILS, \  
 CMD\_GET\_OBJECT\_DETAILS, \  
 CMD\_GET\_DUTY\_SCHEDULE, \  
 }  
  
#define SecurityCommands { \  
 CMD\_PAY\_ACCIDENT, \  
  
...  
  
// id, name, commands, privilegyID, payMult, payPeriod  
// AUTO - only initiated by server logic automaticaly and have all permissions  
#define ROLE\_MAP(XX) \  
 XX( -1, AUTO, {} ) \  
 XX( 0, Admin, AdminCommands ) \  
 XX( 1, Security, SecurityCommands ) \  
 XX( 2, Inkosor, InkosorCommands ) \  
 XX( 3, Recruter, RecruterCommands ) \  
 XX( 4, WaponManager, WaponManagerCommands ) \  
 XX( 5, Customer, CusomerCommands ) \  
  
#define XX(id, name, commands) ROLE\_ ##name = id,  
 enum RoleId {  
 ROLE\_MAP(XX)  
 ROLES\_COUNT  
 };  
#undef XX

// id, name, executor  
#define COMMANDS\_MAP(XX) \  
 XX( 0, MAKE\_CONTRACT, exec\_make\_contract ) \  
 XX( 1, MAKE\_DUTY\_SCHEDULE, exec\_make\_duty\_schedule ) \  
 XX( 2, REGISTER\_ACCIDENT, exec\_register\_accident ) \  
 XX( 3, REGISTER\_EMPLOYEE, exec\_register\_employee ) \  
 XX( 4, REGISTER\_CUSTOMER, exec\_register\_customer ) \  
 XX( 5, REGISTER\_OBJECT\_TYPE, exec\_register\_object\_type ) \  
 XX( 6, REGISTER\_WAPON, exec\_register\_wapon ) \  
 XX( 7, ASSIGN\_WAPON, exec\_assign\_wapon ) \  
 XX( 8, PAY\_AMMO, exec\_pay\_ammo ) \  
 XX( 9, PAY\_EMPLOYEE, exec\_pay\_employee ) \  
 XX( 10, PAY\_ACCIDENT, exec\_pay\_accident ) \  
 XX( 11, EDIT\_OBJECT\_TYPE, exec\_edit\_object\_type ) \  
 XX( 12, UPDATE\_ROLE, exec\_update\_role ) \  
 XX( 13, GET\_USER\_INFO, exec\_get\_user\_info ) \  
 XX( 14, GET\_ACCIDENT\_DETAILS, exec\_get\_accident\_details ) \  
 XX( 15, GET\_ACCOUNTING\_ENTRY, exec\_get\_accounting\_entry ) \  
 XX( 16, GET\_OBJECT\_DETAILS, exec\_get\_object\_detalils ) \  
 XX( 17, GET\_ROLE\_DETAILS, exec\_get\_role\_details ) \  
 XX( 18, GET\_WAPON\_DETAILS, exec\_get\_wapon\_details ) \  
 XX( 19, GET\_DUTY\_SCHEDULE, exec\_get\_duty\_schedule ) \  
 XX( 20, CREATE\_TABLE, exec\_create\_table ) \  
 XX( 21, IDENTIFY, exec\_identify ) \  
  
#define XX(id, name, e) CMD\_ ##name = id,  
enum CommandId {  
 COMMANDS\_MAP(XX)  
 COMMANDS\_COUNT  
};  
#undef XX

Сами команды, они же executor, - это обычные функции c-style:

// AUTO only commands  
CmdError exec\_create\_table(QJsonObject& obj);  
CmdError exec\_identify(QJsonObject& obj);  
  
// Role avalible commands  
CmdError exec\_make\_contract(QJsonObject& obj);  
CmdError exec\_make\_duty\_schedule(QJsonObject& obj);  
CmdError exec\_register\_accident(QJsonObject& obj);

Команды(executor) напрямую получают данные переданные клиент-программами, и возвращают значение в том же переданном аргументе, практически без обработки, таким образом на них ложиться задача валидации переданных аргументов и само исполнение. Пример команды:

/\*  
 \* login: string  
 \* password: string  
 \*/  
CmdError  
exec\_identify(QJsonObject& obj)  
{  
 QSqlQuery q;  
 QString login;  
 QString password;  
  
 login = obj.take("login").toString();  
 password = obj.take("password").toString();  
  
 if (!login.length() || !password.length()) {  
 return CmdError(InvalidParam, "Passed empty parameters");  
 }  
  
 q.prepare("SELECT id, name, role\_id, password, salt FROM Users "  
 "WHERE login = :login");  
 q.bindValue(":login", login);  
  
 if (!q.exec()) {  
 return CmdError(SQLError, q.lastQuery() + " " + q.lastError().text());  
 }  
 if (!q.next()) {  
 return CmdError(AccessDenied, "No user registreted with login: " + login);  
 }  
  
 QByteArray salt = q.record().value("salt").toByteArray();  
 QByteArray real\_passwordHash = q.record().value("password").toByteArray();  
 QByteArray passed\_passwordHash = encryptPassword(password.toLatin1(), salt);  
  
 if (real\_passwordHash != passed\_passwordHash) {  
 return CmdError(AccessDenied, "Invalid password");  
 }  
  
 obj["role\_id"] = q.record().value("role").toInt();  
 obj["name"] = q.record().value("name").toString();  
 obj["id"] = q.record().value("id").toString();  
  
 return CmdError();  
}

Факт: на этом этапе я обнаружил баг, который заставил меня сидеть час с надутым лицом. Значение “id” сохраняется как строка, а в програме клиенте я разархивировал это значение как целочисленое, в итоге получал стандартное значение “0”.

#### Криптография

При регистрации пользователя, как было сказано ранее, используется функция хэширования пароля, для большей безопасности учетных записей. Можно было бы использовать просто функцию хэширования, но я выбрал более сложный и надежный путь.

Кроме того, что при хешировании используется динамическая соль, так она еще и “умная”, написана так, чтобы максимально увеличивать энтропию пароля, но и “умная” вставка соли в строку пароля. Алгоритм был найден мной в журнале [IAENG International Journal of Computer Science](http://www.iaeng.org/IJCS/issues_v43/issue_1/IJCS_43_1_04.pdf) 2016 года.

Суть алгоритма в том, чтобы приводить пароль к максимальной энтропии с использованием динамической умной соли и использования особого метода вставки соли в пароль по одному из 4 или 5 правил на выбор. Умная вставка, по мнению автора, должна свести радужные таблицы к эффективному минимуму, что при тестах он и продемонстрировал.

Чтобы определить слабые стороны пароля мы просто перебераем символы пароля в 3 группы:

* Буквы
* Цыфры
* Специальные знаки

и исходя из из процентного содержания в исходной строке выбираем алфавит для генерации соли:

static void countChars(int& spec, int& dig, int& alph, const char \* str) {  
 spec = dig = alph = 0;  
 for (int i = 0; i < strlen(str); i++) {  
 char ch = str[i];  
 if (isalpha(ch)) { alph++;  
 } else if (isdigit(ch)) { dig++;  
 } else if (isgraph(ch) || isspace(ch)) { spec++;  
 }  
 }  
}  
  
quint8 passWeaknesses(const QByteArray& data)  
{  
 int len = data.length();  
 int spec, nums, alph;  
 countChars(spec, nums, alph, data.data());  
 double spec\_c = static\_cast<double>(spec)/len,  
 nums\_c = static\_cast<double>(nums)/len,  
 alph\_c = static\_cast<double>(alph)/len;  
  
 // weakness determinission algo:  
 // 1. find max coef char type - mostch  
 // 2. if mostch count grater then 50% of password mean that weak in both other char types;  
 // 3. if diff of other char types grater then 10% mean that weak only in one min char type;  
 // 4. if diff less or eq to 10% - weak in both;  
 auto determWeakness = [](double f, double s, double t, quint8 wf, quint8 ws, quint8 wt) -> quint8 {  
 if (f > s && f > t) {  
 if (f > 0.5) {  
 return ws | wt;  
 } else {  
 if (std::abs(s - t) > 0.1) {  
 return (s > t ? wt : ws);  
 } else {  
 return ws | wt;  
 }  
 }  
 }  
 return 0;  
 };  
  
 quint8 w1 = determWeakness(spec\_c, nums\_c, alph\_c, Special, Digit, Alpha);  
 quint8 w2 = determWeakness(nums\_c, spec\_c, alph\_c, Digit, Special, Alpha);  
 quint8 w3 = determWeakness(alph\_c, spec\_c, nums\_c, Alpha, Special, Digit);  
  
 return std::max(w1, std::max(w2, w3)); // only one gr then 0  
}

Метод создания динамической соли:

static quint32 strongRand(quint32 min,  
 quint32 max = std::numeric\_limits<quint32>::max())  
{  
 return QRandomGenerator::securelySeeded().generate() % (max+1 - min) + min;  
}  
  
static char randCharFrom(const QLatin1String& d, quint32 rand32) { return d[rand32 % d.size()].toLatin1(); }  
  
enum charWeakness : quint8 {  
 Alpha = 0x1,  
 Digit = 0x2,  
 Special = 0x4,  
};  
  
static QMap<quint8, std::function<char(quint32)>> saltCharGen({  
 { Alpha, [](quint32 rand32) { return randCharFrom(alphabet, rand32); } },  
 { Digit, [](quint32 rand32) { return randCharFrom(digits, rand32); } },  
 { Special, [](quint32 rand32) { return randCharFrom(specials, rand32); } },  
 { Alpha | Digit, [](quint32 rand32) { return (strongRand(0, 1) ? randCharFrom(alphabet, rand32) : randCharFrom(digits, rand32)); } },  
 { Alpha | Special, [](quint32 rand32) { return (strongRand(0, 1) ? randCharFrom(alphabet, rand32) : randCharFrom(specials, rand32)); } },  
 { Special | Digit, [](quint32 rand32) { return (strongRand(0, 1) ? randCharFrom(specials, rand32) : randCharFrom(digits, rand32)); } },  
 });  
  
QByteArray saltGen(quint8 w)  
{  
 QByteArray salt(salt\_length, '\0');  
 QVector<quint32> rand;  
 rand.resize(salt\_length);  
 QRandomGenerator::securelySeeded().fillRange(rand.data(), rand.size());  
 auto genf = saltCharGen[w];  
  
 for (int i = 0; i < salt\_length; i++) {  
 salt[i] = genf(rand[i]);  
 }  
  
 return salt;  
}

И заключающим акордом является сама функция генерации хэша пароля:

static void setBit(unsigned long& num, unsigned long bit) { num |= (1 << bit); }  
static int getBit(unsigned long num, unsigned long bit) { return (num & ( 1 << bit )) >> bit; }  
  
// There is no check bit or little endiang, may currupt on db export to another machine :) im not care :)  
// Implmented by Algorithm struct published in "Proposed Algorithm from IAENG International Journal of Computer Science, 43:1, IJCS\_43\_1\_04"  
QByteArray encryptPassword(const QByteArray& pass, const QByteArray& salt, QCryptographicHash::Algorithm hashAlgo)  
{  
 QByteArray toCrypt;  
 QByteArray hash = QCryptographicHash::hash(pass, hashAlgo);  
  
 int prev = -1;  
 int cur;  
 int inserted = 0;  
 for (int i = 0; i < pass.length(); i++) {  
 toCrypt.push\_back(pass[i]);  
 // push real password char  
  
 // used Rule №2  
 if (inserted < salt.length()) {  
 cur = getBit(pass[i], sizeof(pass[i])) ^  
 getBit(hash[i], sizeof(hash[i]));  
  
 if (cur == 1) {  
 toCrypt.push\_back(salt[inserted++]);  
 } else if (prev == 0) { // two consecutive zeros  
 toCrypt.push\_back(salt[inserted++]);  
 toCrypt.push\_back(salt[inserted++]);  
 i++; //skip 2  
 }  
 prev = cur;  
 }  
 }  
  
 // append remaining salt  
 if (inserted < salt\_length) {  
 for (int i = inserted; i < salt\_length; i++) {  
 toCrypt.push\_back(salt[i]);  
 }  
 }  
  
 return QCryptographicHash::hash(toCrypt, hashAlgo);  
}

### Объект “сервер”

Сервер - это класс содержащий стандартные методы создания серверного слушающего и клиентских сокетов и обработки “сообщений” от клиентских сокетов.

#### Линк

Линк - объект содержащий клиентский сокет, созданый сервером и содержащий информацию о подключенном клиенте siisty, методы по парсингу типизированных “сообщений” iiNPack и отсылке “сообщений”. Генерирует сигналы о изменении состояния сокета и о приеме сообщения.

#### Процессор подключений

Обработка всех TCP линков обслуживается в многопоточном режиме с использованием “диспетчера уведомлений”. На каждое подключение создается линк и помещается в связаный список линков.

При получении сигнала типа “сообщение” от линка создается воркер в пуле потоков, в котором и происходит обработка сообщения, откуда с помощью событийной модели данные передаются в другие модули и это является бутылочным горлышком данной модели, когда из потока данных всех линков все сливается в один, для обеспечения SOLID стиля кода.

**TODO UML MODEL**

#### Сервис регистрации

Т.к. для подачи заявки на вступление в ЧОП и составление контракта клиентом необходимо иметь обратную связь с инициатором события, приложению сервера необходимо иметь какой то сервис, обеспечивающий регистрацию, в нашем случае регистрация будет с подтверждением по e-mail.

Для регистрации нужно послать определенное сообщение на сервер и после отправить еще одно с кодом подтверждения из письма на указанный e-mail. После этого пользователь будет зарегестрирована в системе как Альфа-рекрут или Клиент и с ним можно будт связывать с помощью внутренних сервисов siisty-server.

## Клиент

В данном подразделе будут описанны пути реализации приложения-клиента.

### Потоки данных

### Потоки данных

### Потоки данных

### Потоки данных

**TODO**

### Service

Это объект на подобии Линк, описанного выше, организует интерфейс для доступа к сервисам ЧОП.

### Менеджер групп страниц

Менеджер групп страниц - это простой ассоциативный массив, ключем является роль, а данными является функция развертывающая необходимый UI.

# Заключение

# Послесловие

# Список литиратуры

1. [https://wiki.qt.io](документация%20Qt)
2. [https://www.sqlite.org](документация%20SQLite)
3. [http://www.iaeng.org/IJCS/issues\_v43/issue\_1/IJCS\_43\_1\_04.pdf](Computer%20science%20journal)
4. Unix Network Programming: The Sockets Networking Api - Addison-Wesley Professional; Subsequent edition (November 1, 2003) - by W. Richard Stevens (Author), Bill Fenner (Author), Andrew M. Rudoff (Author)