

Колледж космического машиностроения и технологий

**КУРСОВОЙ ПРОЕКТ**

**По ПМ.02 Разработка и администрирование баз данных**

**Тема: «Автоматизация рабочего места менеджера автошколы»**

Выполнил студент

Денисов М. В.

Группа П1-18

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Подпись)

\_\_\_\_\_\_\_\_\_\_\_ (Дата сдачи работы)

Принял преподаватель

Попов В.Н.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Оценка)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Подпись)

Королев, 2022

Оглавление

[Введение 3](#_Toc100368325)

[1. Теоретическая часть 4](#_Toc100368326)

[1.1. Описание предметной области 4](#_Toc100368327)

[1.2. Входные и выходные документы 5](#_Toc100368328)

[1.3. Модель бизнес процесса 5](#_Toc100368329)

[1.4. Сценарий работы клиентского приложения 9](#_Toc100368330)

[1.5. Реализация базы данных 10](#_Toc100368331)

[1.6. Таблицы базы данных 10](#_Toc100368332)

[2. Разработка клиентского приложения 10](#_Toc100368333)

[2.2. Разработка пользовательского интерфейса программного средства 10](#_Toc100368334)

[3. Методы и технологии защиты информации в базах данных 13](#_Toc100368335)

[3.1. Регламент работы с вычислительной техникой 13](#_Toc100368336)

[3.2. Регламента работы с ВТ 15](#_Toc100368337)

[3.3. Список доступа к работе с компьютерами и базой данных 15](#_Toc100368338)

[3.4. Регламент разграничивающий доступ 15](#_Toc100368339)

[3.5. Аппаратные средства защиты информации 15](#_Toc100368340)

[4. Условия эксплуатации 15](#_Toc100368341)

[4.1. Климатические условия эксплуатации 15](#_Toc100368342)

[5. Требования к численности и квалификации 15](#_Toc100368343)

[5.1. Требования к численности и квалификации персонала 15](#_Toc100368344)

[Заключение 16](#_Toc100368345)

# Введение

Курсовой проект представляет собой приложение по контролю учеников в автошколе.

В первой части будут рассмотрены предметная область и существующие продукты по данной теме.

Во второй части будут описаны инструменты разработки программы и модули программы.

В заключительной части будут приведены общие выводы по проекту.

# Теоретическая часть

## Описание предметной области

Университет образован 16 июля 1998 года в форме некоммерческой организации с названием: Негосударственное образовательное учреждение «Королевская академия управления, экономики и социологии».

27 апреля 1999 года наименование было изменено на Негосударственное образовательное учреждение «Королевский институт управления, экономики и социологии».

Постановлением Главы города Королёва от 13 июня 2001 года №1009 Королевский институт управления, экономики и социологии был преобразован в муниципальное образовательное учреждение.

26 декабря 2005 года постановлением Правительства Московской области № 964/52 вуз перешёл в собственность Московской области и получил наименование: Государственное образовательное учреждение высшего профессионального образования Московской области «Королевский институт управления, экономики и социологии». Этим же постановлением учредителем вуза было назначено Министерство образования Московской области.

Постановлением Правительства Московской области от 25.09.2007 года № 725/32 Государственное образовательное учреждение высшего профессионального образования Московской области «Королевский институт управления, экономики и социологии» передано в ведомственное подчинение Министерства финансов Московской области.

В конце декабря 2011 года Приказом Федеральной службы по надзору в сфере образования и науки КИУ­ЭС признан прошедшим проце­дуру государственной аккреди­тации с установлением статуса «академия». 12 апреля 2012 года постановлением Правительства Московской области № 484/13 вуз переименован в Государственное бюджетное образовательное учреждение высшего профессионального образования Московской области «Финансово-технологическая академия».

11 июля 2012 года постановлением Правительства Московской области № 906/24 Академия была реорганизована в форме присоединения к ней государственного бюджетного образовательного учреждения среднего профессионального образования Московской области «Королевский государственный техникум технологии и дизайна одежды» и государственного бюджетного образовательного учреждения среднего профессионального образования Московской области «Королевский колледж космического машиностроения и технологии»

26 декабря 2013 года постановлением Правительства Московской области от 26.12.2013 № 1137/57 «О передаче Государственного бюджетного образовательного учреждения высшего профессионального образования Московской области «Финансово-технологическая академия» в ведомственное подчинение Министерству образования Московской области» Академия была передана в ведомственное подчинение Министерству образования Московской области.

20 января 2015 года постановлением Правительства Московской области от 20.01.2015 № 9/2 Академии присвоен статус «университета» и вуз переименован в Государственное бюджетное образовательное учреждение высшего образования Московской области «Технологический университет».

3 апреля 2015 года Технологическим университетом, Администрацией г.о. Королёв и ведущими предприятиями ракетно-космической отрасли РФ, работающими в наукограде Королёве, ряде городов Московской области и г. Москве, подписано соглашение о создании консорциума «Региональный научно-образовательный кластер «Северо-восток» с целью обеспечения потребностей наукоёмких высокотехнологичных предприятий квалифицированными кадрами.

## Входные и выходные документы

В качестве входных документов для работы с программой необходим документ с данными учеников автошколы.

Документ данных учеников автошколы – это документ в котором описаны фамилия, имя, отчество, дата рождения, количество занятий.

В качестве выходного документа является «Excel файл» отчёт о прогрессе в обучении ученика. Этот документ содержит изменённую оператором информацию.

## Модель бизнес процесса

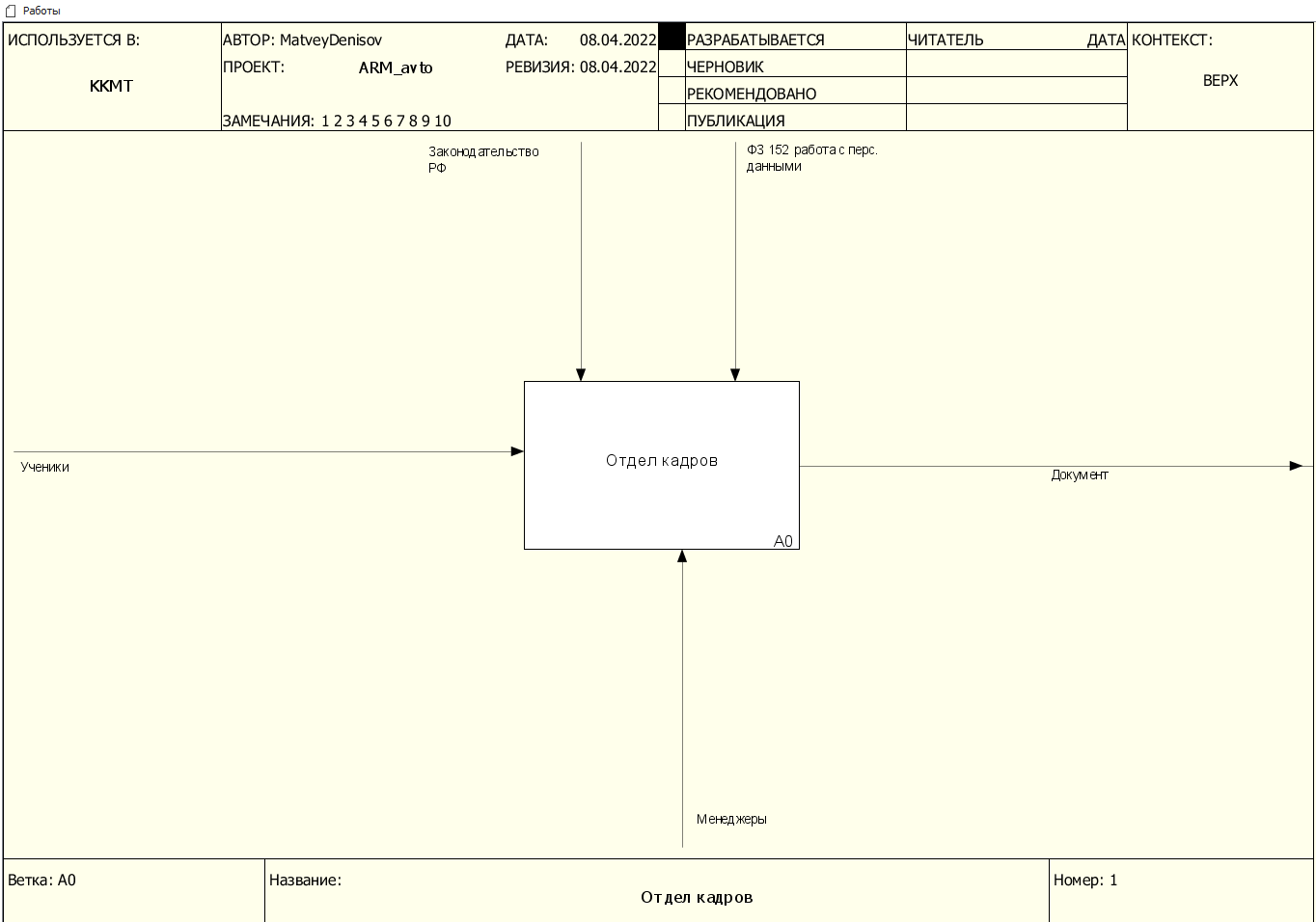


Рисунок 1. Первый уровень.

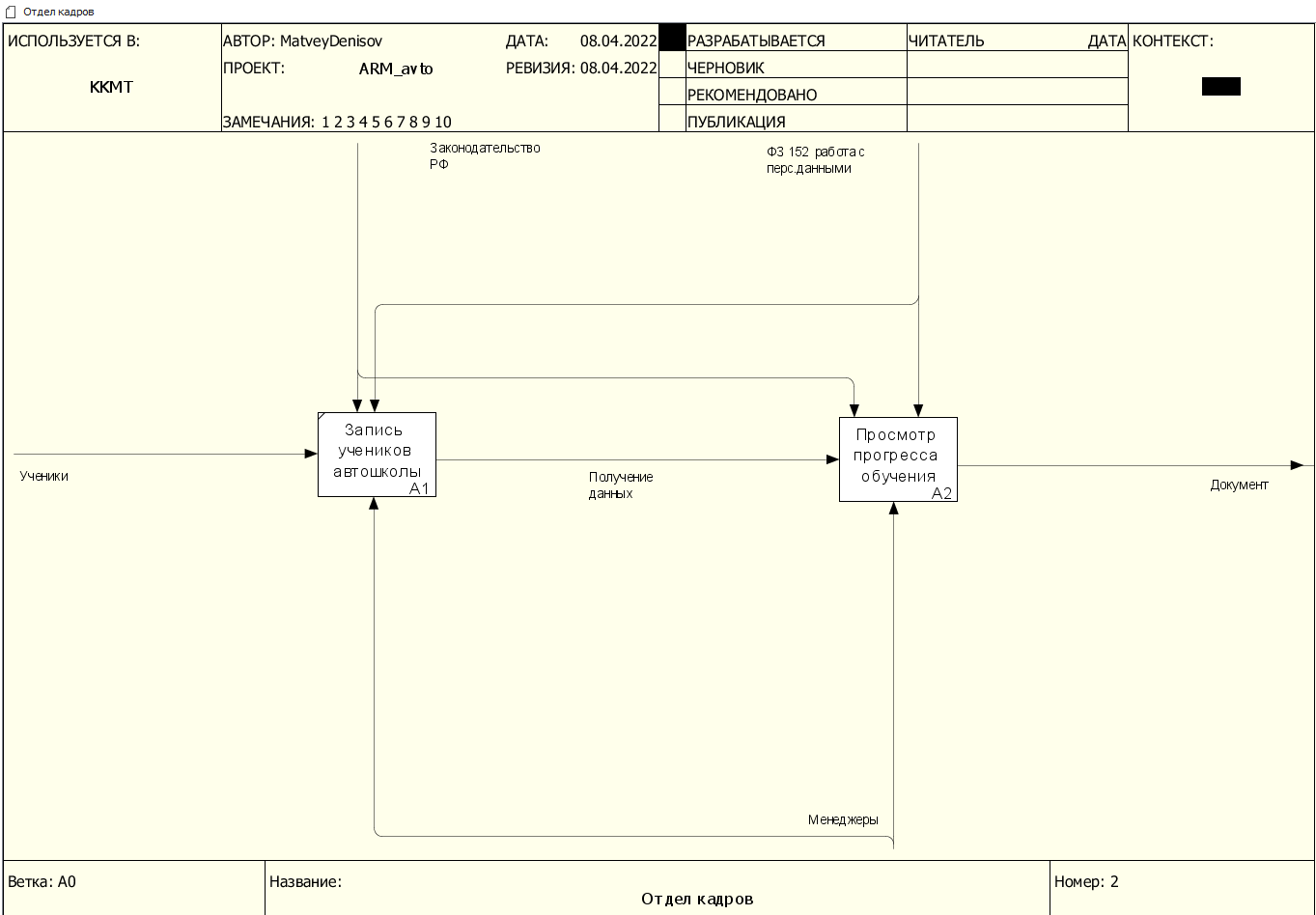


Рисунок 2. Второй уровень.

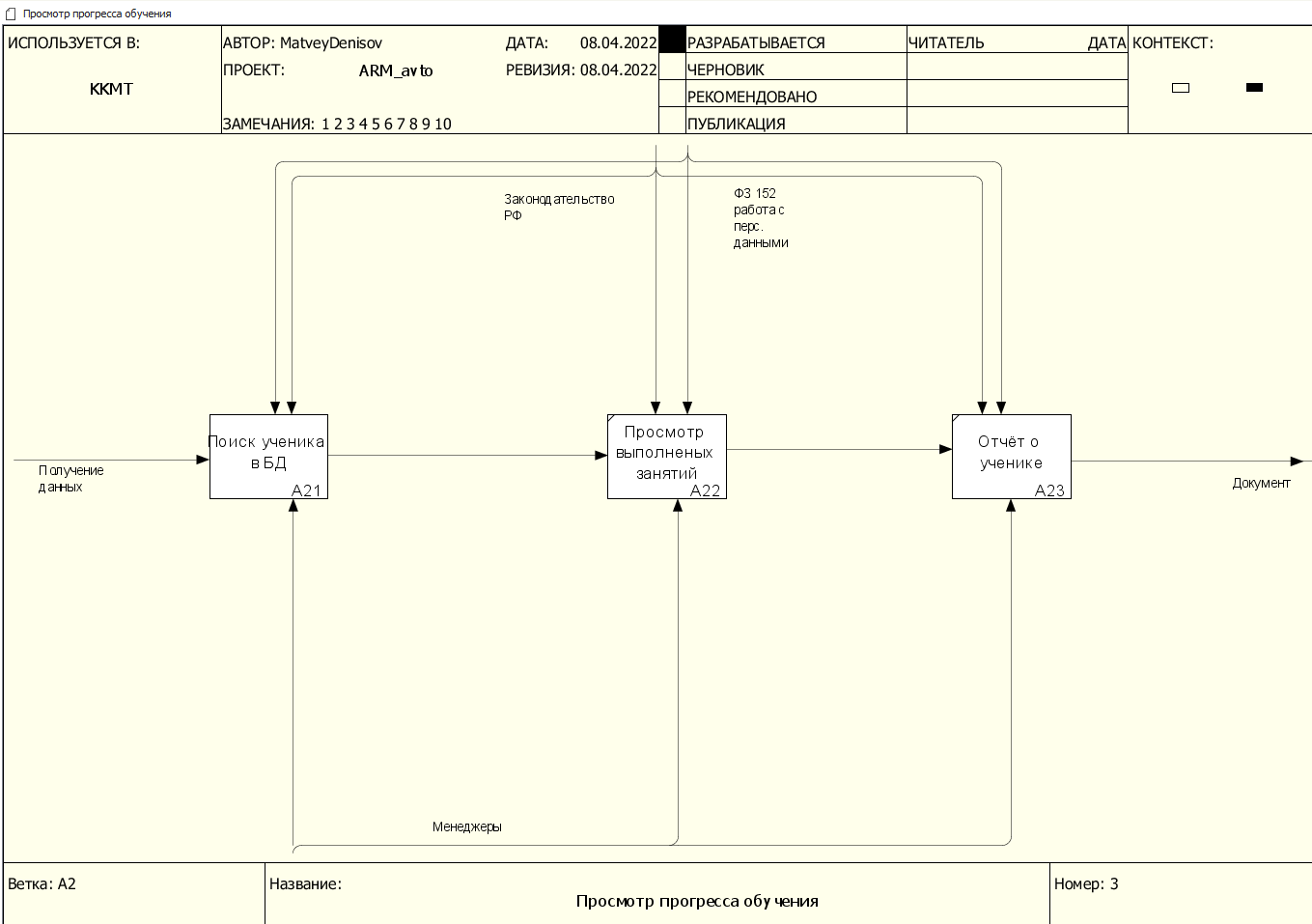


Рисунок 3. Третий уровень.

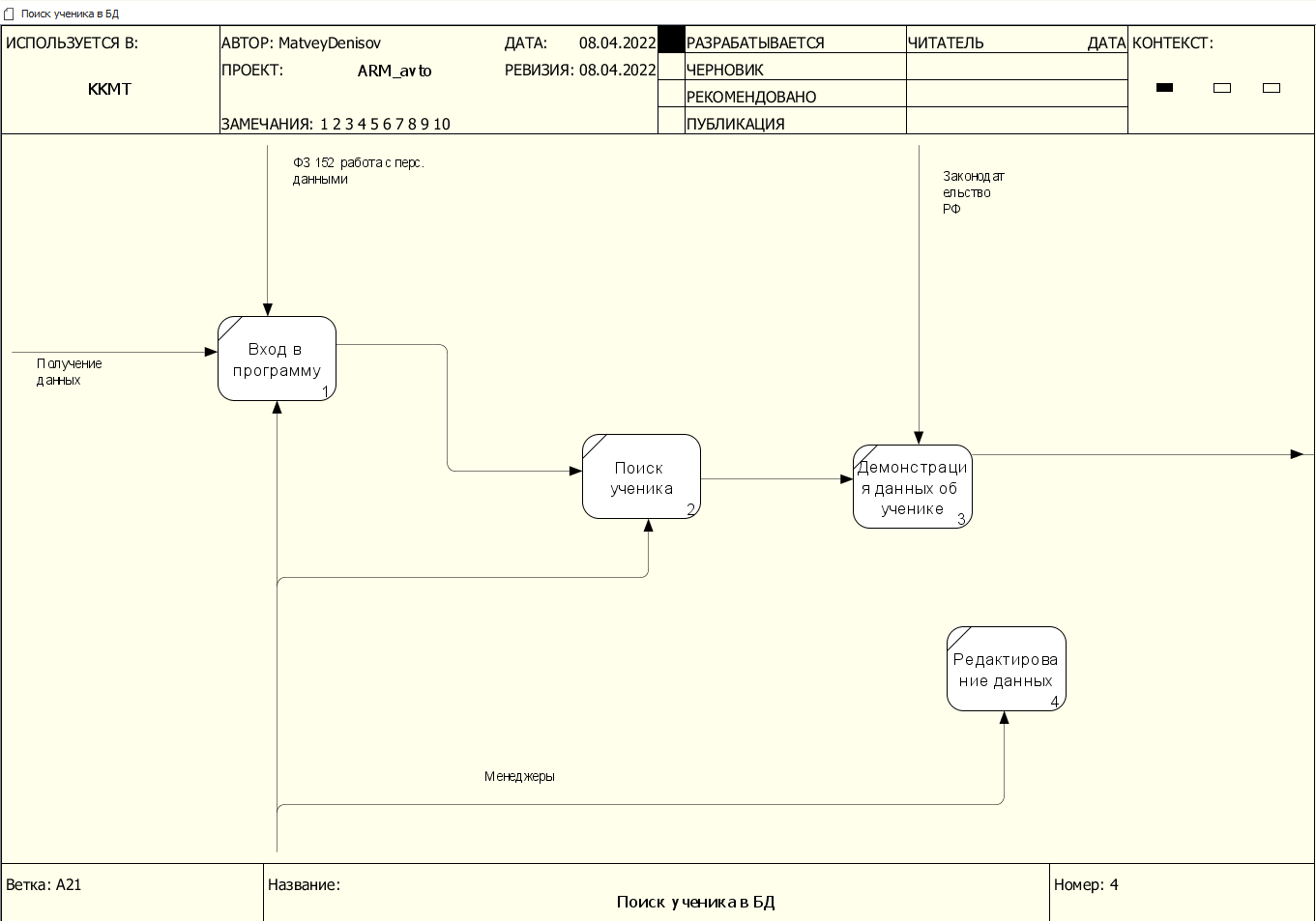


Рисунок 4. DFD уровень.

## Сценарий работы клиентского приложения

В начале работы, после входа в программу, оператору необходимо ввести свой логин и пароль в поля для ввода. После того как вы ввели корректный логин и пароль, перед вами откроется главное окно для работы с таблицей.

Затем оператор вводит ФИО ученика для поиска определённого человека. Все данные будут доступны оператору после входа.

После того как пользователь получил данные об ученике он может запросить программу, о создании Excel файла.

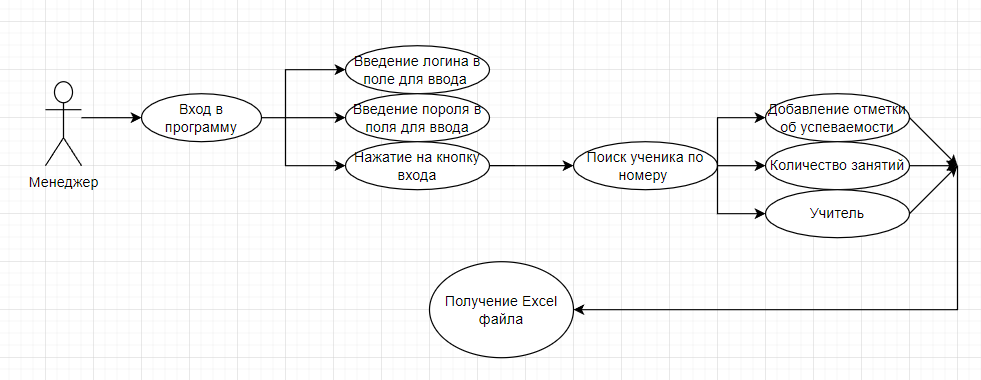


Рисунок 5. Сценарий работы клиентского приложения.

Для того чтобы изменить данные об ученике надо просто нажать двойным кликом по ячейке таблицы, вписать нужные данные о учении и нажать на кнопку «сохранить».

Для того чтобы добавить ученика менеджер нажимает на кнопку «добавить», перед менеджером откроется окно с полями для ввода, которые необходимо заполнить.

После полной записи изменений для, получения данных нажать на кнопку «Excel».

## Реализация базы данных

В БД присутствуют:

1. Таблицы
2. Первичные ключи
3. Внешние ключи
4. Индексы Последовательности
5. Представления

## Таблицы базы данных

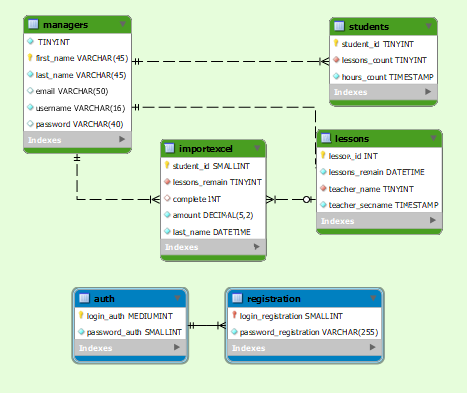


Рисунок 6. Схема базы данных.

# Разработка клиентского приложения

## 2.2. Разработка пользовательского интерфейса программного средства

После запуска программы вас встретит окно с авторизацией под аккаунт.

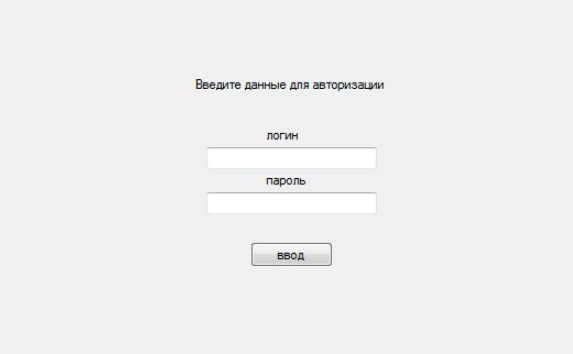


Рисунок 7. Окно авторизации.

Далее вы будете в главном меню программы, здесь вы сможете найти клиента, и добавить ему данные о учёбе.

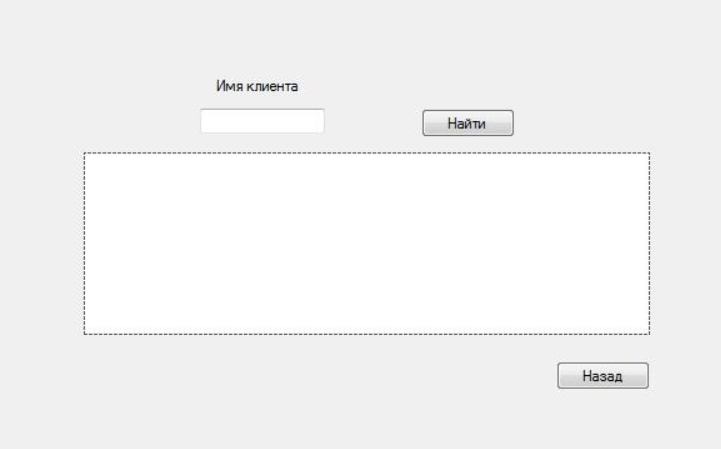


Рисунок 8. Окно нахождения ученика.

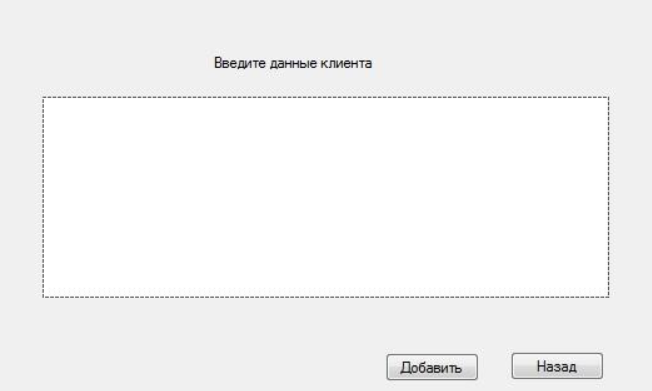


Рисунок 9. Добавление данных о клиенте.

После добавления и редактирования всех данных о клиенте, вы можете запросить выдачу Excel файла.

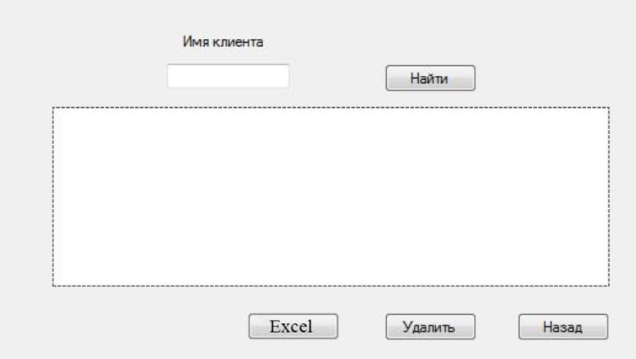


Рисунок 10. Возможность запроса на выдачу Excel.

# Методы и технологии защиты информации в базах данных

## Регламент работы с вычислительной техникой

Настоящая инструкция распространяется на персонал, эксплуатирующий средства вычислительной техники и периферийное оборудование. Инструкция содержит общие указания по безопасному применению электрооборудования в учреждении. Требования настоящей инструкции являются обязательными, отступления от нее не допускаются. К самостоятельной эксплуатации электроаппаратуры допускается только специально обученный персонал не моложе 18 лет, пригодный по состоянию здоровья и квалификации к выполнению указанных работ.

Перед началом работы следует убедиться в исправности электропроводки, выключателей, штепсельных розеток, при помощи которых оборудование включается в сеть, наличии заземления компьютера, его работоспособности.

Для снижения или предотвращения влияния опасных и вредных факторов необходимо соблюдать? Санитарные правила и нормы. гигиенические требования к видеодисплейным терминалам, персональным электронно-вычислительным машинам и организации работы¦ (Утверждено Постановлением Госкомсанэпиднадзора России от 14 июля 1996 г. N 14 СанПиН 2.2.2.542-96), и Приложение 1,2

Во избежание повреждения изоляции проводов и возникновения коротких замыканий не разрешается: вешать что-либо на провода, закрашивать и белить шнуры и провода, закладывать провода и шнуры за газовые и водопроводные трубы, за батареи отопительной системы, выдергивать штепсельную вилку из розетки за шнур, усилие должно быть приложено к корпусу вилки.

Для исключения поражения электрическим током запрещается: часто включать и выключать компьютер без необходимости, прикасаться к экрану и к тыльной стороне блоков компьютера, работать на средствах вычислительной техники и периферийном оборудовании мокрыми руками, работать на средствах вычислительной техники и периферийном оборудовании, имеющих нарушения целостности корпуса, нарушения изоляции проводов, неисправную индикацию включения питания, с признаками электрического напряжения на корпусе, класть на средства вычислительной техники и периферийном оборудовании посторонние предметы.

Запрещается под напряжением очищать от пыли и загрязнения электрооборудование.

Запрещается проверять работоспособность электрооборудования в неприспособленных для эксплуатации помещениях с токопроводящими полами, сырых, не позволяющих заземлить доступные металлические части.

Недопустимо под напряжением проводить ремонт средств вычислительной техники и периферийного оборудования. Ремонт электроаппаратуры производится только специалистами-техниками с соблюдением необходимых технических требований.

Во избежание поражения электрическим током, при пользовании электроприборами нельзя касаться одновременно каких-либо трубопроводов, батарей отопления, металлических конструкций, соединенных с землей.

При пользовании электроэнергией в сырых помещениях соблюдать особую осторожность.

При обнаружении неисправности немедленно обесточить электрооборудование, оповестить администрацию. Продолжение работы возможно только после устранения неисправности.

При обнаружении оборвавшегося провода необходимо немедленно сообщить об этом администрации, принять меры по исключению контакта с ним людей. Прикосновение к проводу опасно для жизни.

Во всех случаях поражения человека электрическим током немедленно вызывают врача. До прибытия врача нужно, не теряя времени, приступить к оказанию первой помощи пострадавшему.

Необходимо немедленно начать производить искусственное дыхание, наиболее эффективным из которых является метод рот в рот или рот в нос, а также наружный массаж сердца.

Искусственное дыхание пораженному электрическим током производится вплоть до прибытия врача.

На рабочем месте запрещается иметь огнеопасные вещества

В помещениях запрещается:

1. зажигать огонь;
2. включать электрооборудование, если в помещении пахнет газом;
3. курить;
4. сушить что-либо на отопительных приборах;
5. закрывать вентиляционные отверстия в электроаппаратуре

Источниками воспламенения являются:

1. искра при разряде статического электричества
2. искры от электрооборудования
3. искры от удара и трения
4. открытое пламя

При возникновении пожароопасной ситуации или пожара персонал должен немедленно принять необходимые меры для его ликвидации, одновременно оповестить о пожаре администрацию.

Помещения с электрооборудованием должны быть оснащены огнетушителями типа ОУ-2 или ОУБ-3.

После окончания работы необходимо обесточить все средства вычислительной техники и периферийное оборудование. В случае непрерывного производственного процесса необходимо оставить включенными только необходимое оборудование.

## Регламента работы с ВТ

Не требуется

## Список доступа к работе с компьютерами и базой данных

Доступ к работе с базой данных имеется лишь у менеджеров.

## Регламент разграничивающий доступ

Администратор занимается:

1. Управлением БД

## Аппаратные средства защиты информации

В виде аппаратных средств защиты информации выступает роутер с Firewall.

# Условия эксплуатации

Программа запускается на любой системе, позволяющей открыть браузер. Окно программы должно быть открыто на весь экран, должна быть возможность свернуть приложение или запустить любое стороннее программное обеспечение.

## Климатические условия эксплуатации

Специальные условия не требуются.

# Требования к численности и квалификации

Программа не требует проведения каких-либо видов обслуживания.

## Требования к численности и квалификации персонала

Специальные условия не требуются.

# Заключение

Данный курсовой проект был разработан в целях автоматизации работы с базой данных. Данный проект может быть использован в учебных целях для обучения программированию на Golang.

Листинг кода. Main.go

package main

import (

"fmt"

"fyne.io/fyne"

"fyne.io/fyne/app"

"fyne.io/fyne/layout"

"fyne.io/fyne/theme"

"fyne.io/fyne/widget"

"io"

"log"

"net"

"os"

)

func main() {

cfg, err := LoadConfig("config.toml")

if err != nil {

log.Fatal(err)

}

logFile, err := os.OpenFile(cfg.Log, os.O\_RDWR|os.O\_CREATE|os.O\_APPEND, 0666)

if err != nil {

log.Fatal(err)

}

defer logFile.Close()

log.SetOutput(logFile)

log.Println("Server addr: ", cfg.Srv)

a := app.New()

w := a.NewWindow("Client")

planEvent := widget.NewEntry()

executor := widget.NewEntry()

count := widget.NewEntry()

stateLabel := widget.NewLabel("")

form := widget.NewForm(

widget.NewFormItem("План. мероприятия", planEvent),

widget.NewFormItem("Исполнители", executor),

widget.NewFormItem("Количество обуч.", count),

widget.NewFormItem("Статус", stateLabel),

)

btnSave := widget.NewButton("OK", func() {

dataSend := fmt.Sprintf("%s %s %s", planEvent.Text, executor.Text, count.Text)

if err := sendData(cfg.Srv, dataSend); err != nil {

log.Println(err)

stateLabel.SetText(err.Error())

} else {

stateLabel.SetText("данные сохранены")

}

})

btnClear := widget.NewButton("Очистить", func() {

planEvent.SetText("")

executor.SetText("")

count.SetText("")

stateLabel.SetText("")

})

btnExit := widget.NewButton("Выход", func() {

a.Quit()

})

w.SetContent(

widget.NewVBox(

form,

layout.NewSpacer(),

widget.NewHBox(layout.NewSpacer(), btnClear, btnSave, btnExit),

))

w.Resize(fyne.NewSize(480, 200))

a.Settings().SetTheme(theme.LightTheme())

w.ShowAndRun()

}

func sendData(addr, data string) error {

conn, err := net.Dial("tcp", addr)

if err != nil {

return fmt.Errorf("Error connect to server: %v\n", err)

}

if \_, err := conn.Write([]byte(data)); err != nil {

return fmt.Errorf("Error send data to server: %v\n", err)

}

log.Printf("Send data: %v\n", data)

resp := make([]byte, 512)

respLen, err := conn.Read(resp)

if err != nil && err != io.EOF {

return fmt.Errorf("Response error: %v\n", err)

}

log.Println("Packet processing: ", string(resp[:respLen]))

return conn.Close()

}

Листинг кода. Config.go

package main

import (

"github.com/BurntSushi/toml"

)

type Settings struct {

Srv string

Log string

}

//LoadConfig load configuration from file

func LoadConfig(confPath string) (Settings, error) {

c := Settings{}

\_, err := toml.DecodeFile(confPath, &c)

return c, err

}

Листинг кода. Db.go

package main

import (

"database/sql"

"fmt"

\_ "github.com/mattn/go-sqlite3"

)

type Store interface {

Insert([]string) error

}

type SQLLiteStore struct {

db \*sql.DB

}

func (s \*SQLLiteStore) Connect(dbName string) error {

var err error

s.db, err = sql.Open("sqlite3", dbName)

if err != nil {

return fmt.Errorf("Open db error: %v\n", err)

}

if err = s.db.Ping(); err != nil {

return fmt.Errorf("Connection db error: %v\n", err)

}

\_, err = s.db.Exec(`create table IF NOT EXISTS schedule (event TEXT, executor TEXT, count TEXT,

created TIMESTAMP DEFAULT CURRENT\_TIMESTAMP)`)

if err != nil {

return fmt.Errorf("Init table error: %v\n", err)

}

return err

}

func (s \*SQLLiteStore) Insert(rec []string) error {

var err error

if len(rec) <= 3 {

\_, err = s.db.Exec("insert into schedule (event, executor, count) values ($1, $2, $3)",

rec[0], rec[1], rec[2])

} else {

err = fmt.Errorf("Incorrect record len\n")

}

return err

}

Листинг кода. Main.go

package main

import (

"fyne.io/fyne/app"

"fyne.io/fyne/layout"

"fyne.io/fyne/theme"

"fyne.io/fyne/widget"

"log"

"os"

)

func main() {

cfg, err := LoadConfig("config.toml")

if err != nil {

log.Fatal(err)

}

logFile, err := os.OpenFile(cfg.Log, os.O\_RDWR|os.O\_CREATE|os.O\_APPEND, 0666)

if err != nil {

log.Fatal(err)

}

defer logFile.Close()

log.SetOutput(logFile)

store := SQLLiteStore{}

if err = store.Connect(cfg.Db); err != nil {

log.Fatal(err)

}

s := Server{}

if err = s.Init(cfg.Srv, &store); err != nil {

log.Fatal(err)

}

go s.Run()

a := app.New()

w := a.NewWindow("Server")

btnSave := widget.NewButton("Выгрузка", func() {

})

btnExit := widget.NewButton("Выход", func() {

a.Quit()

})

w.SetContent(

widget.NewVBox(

widget.NewHBox(layout.NewSpacer(), btnSave, btnExit),

))

a.Settings().SetTheme(theme.LightTheme())

w.ShowAndRun()

}

Листинг кода. Server.go

package main

import (

"fmt"

"io"

"log"

"net"

"strings"

)

type Server struct {

l net.Listener

db Store

}

func (s \*Server) Init(addr string, db Store) error {

var err error

s.l, err = net.Listen("tcp", addr)

if err != nil {

return fmt.Errorf("Error start server: %v\n", err)

}

log.Println("Server start ", addr)

if db == nil {

return fmt.Errorf("Store not initializate\n")

}

s.db = db

return err

}

func (s \*Server) Run() {

for {

conn, err := s.l.Accept()

if err != nil {

log.Printf("Accept error: %v\n", err)

} else {

go s.handlerConn(conn)

}

}

}

func (s \*Server) handlerConn(c net.Conn) {

defer c.Close()

buf := make([]byte, 2048)

rcvPacketSize, err := c.Read(buf)

if err != nil && err != io.EOF {

log.Println("Read error: ", err)

return

}

data := buf[:rcvPacketSize]

rec := strings.Split(string(data), " ")

log.Println("Received data: ", rec)

if len(rec) <= 3 {

if err := s.db.Insert(rec); err != nil {

log.Printf("Insert error: %v\n", err)

}

log.Printf("Save record in DB: %v\n", rec)

if \_, err = c.Write([]byte("OK")); err != nil {

log.Printf("Response send error: %v\n", err)

}

}

}