НИТУ «МИСиС»

Кафедра **инженерной кибернетики**

Квалификация (степень): **магистр**

Направление подготовки: 09.04.03 «Прикладная информатика»

Профиль (программа) «**Инновационные ИТ проекты**»

Дисциплина: Современные инструментальные средства разработки

КУРСОВАЯ РАБОТА

на тему

«Проектирование и разработка системы для отслеживания уровня заболеваемости COVID-19 по ВУЗу»

Выполнили*:*

Руководитель/архитектор: Горожанкин Вадим

Аналитик-разработчик: Малынковский Олег

Разработчик C#: Хабибулин Марат

Разработчик C#/DBA: Кунафин Инсур

Группа: МПИ20-4-2

Проверил: Тарханов И.А.

Оценка: _____

Дата: _____

Москва 2021

Оглавление

1 Перечень принятых сокращений 4
2 Общие сведения 5
2.1 Полное наименование системы и ее условное обозначение 5
2.2 Цели создания системы 5
2.3 Задачи проекта 5
3 Функциональные требования 6
3.1 Общие требования 6
3.2 Требования к роли «Студент» 8
3.3 Требования к роли «Преподаватель» 9
3.4 Требования к роли «Ректорат/деканат» 10
4 Нефункциональные требования 11
5 Сценарий использования 12
5.1 Ознакомление студента со своей группой 12
5.2 Ознакомление с расписанием 13
5.3 Ознакомление со статистикой заболеваемости 14
5.5 Закрытие кафедры или университета администрацией 16
5.6 Заражение студента и уведомление его группы 17
6 Сценарии тестирования 19
6.1 Регистрация и авторизация студента 19
6.2 Просмотр группы для студента 20
6.3 Просмотр расписания для студента 21
6.4 Просмотр статистики для студента 22
6.5 Авторизация преподавателя или администрации 23
6.6 Ведение занятия для преподавателя или администрации (если пользователь по совместительству преподаватель) 24
6.7 Просмотр расписания для преподавателя или администрации (если пользователь по совместительству преподаватель) 25
6.8 Просмотр статистики для преподавателя 26
6.9 Просмотр статистики для администрации 27
6.10 Заражение студента и уведомление однокурсников об их потенциальном заражении 28
7 Прототипы основных окон системы 30
7.1 Профиль 30

7.2 Статистика ... 31
7.3 Расписание ... 33
7.4 Занятия ... 34
7.5 Группа ... 35
7.6 Авторизация и регистрация ... 36
8 Код 38
8.1 Клиентское приложение ... 38
StudentNavigation ... 38
TeacherNavigation ... 40
Group ... 42
AdditionalInformation ... 45
Registration ... 46
Login ... 50
CheckRegProp ... 54
Profile ... 55
UpdateStatus ... 57
DownloadWindow ... 58
LessonView ... 60
ScheduleView ... 64
StatsView ... 68
8.2 База данных ... 75
Создание базы данных ... 75
Создание тестовых данных ... 79
9 Список литературы ... 86

1 Перечень принятых сокращений

№	Обозначение/сокращение	Определение
1.	COVID-19	Коронавирусное инфекционное заболевание
2.	ФИО	Фамилия, Имя, Отчество
3.	AWS	Amazon Web Services
4.	БД	Базы данных
5.	ПК	Персональный компьютер

2 Общие сведения

2.1 Полное наименование системы и ее условное обозначение

Полное наименование системы: COVID – ВУЗ

Условное обозначение: Система, аналитика, динамика.

2.2 Цели создания системы

Целью создания Системы является отслеживание уровня заболеваемости вирусом COVID-19 среди студентов.

2.3 Задачи проекта

Задачи проекта следующие:

– разработка функционала для ведения учета посещения студентов;

– разработка функционала для определения потенциально зараженных студентов на основании посещения;

– создание функционала для возможности отметки о заражении или здоровом состоянии путем прикрепления подтверждающего документа;

– разработка функционала для предоставления информации о зараженных студентах для преподавателя;

– разработка функционала для возможности рекомендация и закрытиея университета или отдельной кафедры в случае достижения определенного уровня заболеваемости.

3 Функциональные требования

3.1 Общие требования

Номер	Название	Условия/ограничения
1.	Регистрация	Система должна давать возможность каждому пользователю зарегистрироваться в системе (при наличии пропуска). Для регистрации студенту необходимо ввести ФИО, номер студенческого билета, паспортные данные, дату рождения и желаемый пароль. После ввода данные будут отправлены в БД для создания новой записи о пользователе. Ролирование автоматическое в зависимости от первой буквы пропуска: - s - Студент - t - Преподаватель - a - Администрация
2.	Вход в систему	Система должна давать возможность войти в систему при наличии учетной записи данного пользователя. Для входа пользователю необходимо ввести имя пользователя (номер студенческого) и пароль, после чего система отправит запрос в БД для проверки существования данного пользователя в системе: при положительном результате осуществит вход в систему под указанным пользователем, при отрицательном будет выведена ошибка о некорректности введенных данных.
3.	Выход из системы	Система должна давать возможность выйти из системы при необходимости. Для этого пользователю будет необходимо нажать на кнопку справа сверху с соответствующим значком.
4.	Просмотр профиля	Система должна давать пользователю возможность просматривать собственный профиль. На странице профиля человека должна быть отображена информация: ФИО, группа (если студент), дата рождения, контактные данные (телефон, email), студенческий билет, если студент болен – отображаются его дата заболевания и дата возрождения, если выздоровел. Также на данном экране присутствует кнопка, которая отмечает о болезни.
5.	Просмотр статистики заболеваемости Covid	Система должна демонстрировать статистику заболеваемости группы и всех учащихся в виде столбчатой диаграммы. На странице просмотра статистики будет отображена информация об общем количестве зараженных по всему университету с приведенным графиком, на котором по вертикали отображал шкала количества зараженных, по горизонтали – дата, когда был заражен студент. Также будет присутствовать фильтр со следующими параметрами – группа, факультет, университет – для того, чтобы пользователь мог указать, в каком разрезе ему необходимо посмотреть информацию.
6.	Просмотр расписания	Система должна иметь возможность отображать расписание преподавателя и студентов.

		В системе будет создана страница, на которой будет отображено 6 таблиц, которые будут содержать информацию о расписании для студента, под данными которого был осуществлен вход в систему. В таблице будет информация о занятиях, аудитории, в котором проводится занятие, преподавателе, который будет проводить занятие, а также ссылка на онлайн-конференцию, если занятие проводится в Teams.

3.2 Требования к роли «Студент»

Номер	Название	Условия/ограничения
1.	Подключение к сервисам онлайн-обучения	Система должна давать возможность перехода студентами по ссылке к онлайн-занятиям из расписания. Так как в таблице с расписанием присутствует ссылка на онлайн-конференцию, в которой будет проведено занятие, будет реализован переход по ссылке из приложения в браузер для открытия конференции.
2.	Просмотр списка студентов группы	Студент имеет возможность просматривать свою группу для ознакомления с составом. Для этого будет создана отдельная страница, на которой будет выведен номер группы, количество студентов в ней, а также куратор группы при его наличии. Будет создана таблица, в которой будет отображена в виде столбцов следующая информация: ФИО студента, дата рождения. Также при нажатии на студента будет открываться всплывающее окно, в котором будет представлена более подробная информация по выбранному студенту: его телефон и email.
3.	Отметка о болезни	Система должна давать возможность отобразить статус своего Covid-состояния. На экране профиля студента будет реализована кнопка, при нажатии на которую будет открываться всплывающее окно, в котором необходимо указать статус своего заболевания. Для каждого из статусов необходимо будет загрузить подтверждающий данный статус документ, после загрузки которого система оправит данные в БД для подтверждения и изменения статуса студента.
4.	Получение уведомления в личном кабинете	Система должна отображать в ЛК риск заражения Covid и рекомендацию к обследованию. Если в группе какой-либо студент заболел, то для всех студентов, которые посещали с заболевшим пары на протяжении последней недели, придет уведомление о необходимости пройти проверку на заражение и прикрепление скана о полученных результатах в систему по механизму, описанному в предыдущем шаге.

3.3 Требования к роли «Преподаватель»

Номер	Название	Условия/ограничения
1.	Простановка посещаемости	Система должна давать возможность преподавателю проставлять присутствие студента на занятии в определенную дату. Для преподавателя будет реализована отдельная страница с занятием. На нем преподавателю будет предложено выбрать группу и дату занятия, после чего система отправит запрос в БД с указанными фильтрами и выведет в таблицу информацию по найденному занятию по указанным данным. В таблице будет отображена информация: ФИО студента, а также будет возможность проставлять баллы и посещаемость студента. После сохранения данных путем нажатия на кнопку, запрос на изменение БД будет отправлен.
2.	Простановка баллов	Система должна давать возможность преподавателю проставлять целочисленные баллы каждому из студентов, кто присутствовал на занятии. В таблице будет отображена информация: ФИО студента, а также будет возможность проставлять баллы и посещаемость студента. После сохранения данных путем нажатия на кнопку, запрос на изменение БД будет отправлен.
3.	Просмотр заболеваемости	Система должна иметь возможность отображать статус заболевания студента в общем списке для преподавателя. В таблице, в которой преподаватель просматривает информацию о ФИО студента и проставляет его баллы за занятие и посещаемость, цветовой палитрой красного оттенка будут выделены студенты, которые сейчас больны и не могут физически посещать занятия, чтобы не заражать сокурсников.

3.4 Требования к роли «Ректорат/деканат»

Номер	Название	Условия/ограничения
1.	Рекомендация к закрытию университета/кафедры	Система должна иметь возможность выводить уведомление с рекомендацией о закрытии всего университета или кафедры. При достижении уровня заболеваемости свыше 60% система будет выводить для роли администратора сообщение, содержащее рекомендацию к закрытию и переводу кафедры/университета на дистанционную основу.
2.	Закрытие университета/кафедры	Система должна иметь возможность закрытия университета/кафедры и уведомление всех причастных к данному вузу/кафедре студентов о выполненной мере. Будет реализовано посредством нажатия на кнопку, после чего для соответствующего университета или кафедры (смотря что закрывают) проставится соответствующий статус, после чего студенты данной группы будут переведены на дистанционную форму обучения.

4 Нефункциональные требования

Номер	Название	Условия/ограничения
1.	Производительность	Время отклика системы должно не превышать 4 секунд. Количество пользователей, который должны иметь одновременный доступ к системе – 1000.
2.	Масштабируемость	Должна быть возможность горизонтальной масштабируемости для увеличения места для хранения информации при заполнении уже текущих дисков, задействованных для функционирования системы.
3.	Безопасность	Серверная часть системы должна быть расположена на собственных серверах (on-premise) по ряду причин: - потенциальная кража данных, если приложение находится в облачном окружении другого производителя, взлом данной платформы может предоставить доступ также к нашим данным; - стабильность – нарушение работы облачной платформы может повредить возможность функционирования и нашего приложения, в связи с чем может уменьшиться спрос. Защита учетных записей пользователей производится за счет пароля, указываемого при регистрации.
4.	Удобство использования	Интерфейс пользователя должен быть интуитивно понятным. Все элементы в системе должны быть реализованы по принципу «важная информация находится в слева сверху, менее справа снизу». Слева будет реализована статичная панель навигации, в которой будут отображены страницы, которые доступны пользователю для посещения, верхняя часть с информацией об университете, группе, а также студенте, который на текущий момент авторизован в системе также будет статична.
5.	Используемые технологии	WPF (C#), в качестве БД - MySQL (AWS), для обращения к БД будет использован Entity Framework.
6.	Лингвистические требования (поддерживаемые языки)	Русский
7.	Минимальные системные требования клиента	Минимальные: Windows 7 ОЗУ – 4гб Процессор - Pentium 4
8.	Технические требования к серверу	В качестве серверного оборудования будет использован Dell PowerEdge R730xd, предназначенный для обработки больших массивов данных Процессор - 1x/2x Xeon E5-2600 v3 “Haswell” Чипсет - Intel C610 Series Chipset ОЗУ – 64гб ОС: RedHat; SuSe Linux

5 Сценарий использования

5.1 Ознакомление студента со своей группой

Цель.

Ознакомиться со своей группой.

Актеры (действующее лицо).

Студент

Заинтересованные лица (Stakeholders).

Студент

Предварительные условия.

Студент должен иметь учетную запись в системе.

Активаторы.

Появление у студента желания ознакомиться с учебной программой своего направления на текущий семестр и проверить состояние заболеваемости в вузе в целом.

Шаги сценария.

1. Авторизоваться в системе;
2. проверить свой профиль;
3. перейти на страницу группы, ознакомиться со своими одногруппниками и куратором;

Альтернативные пути и дополнения.

При желании студент может посмотреть дополнительную информацию о выбранном одногруппнике или кураторе.

5.2 Ознакомление с расписанием

Цель.

Получение информации об учебном процессе.

Актеры (действующее лицо).

Студент, преподаватель, администрация

Заинтересованные лица (Stakeholders).

Студент, преподаватель, администрация

Предварительные условия.

Студент должен иметь учетную запись в системе.

Активаторы.

Появление у студента желания ознакомиться с учебной программой своего направления на текущий семестр и проверить состояние заболеваемости в вузе в целом.

Шаги сценария.

1. Авторизоваться в системе;
2. проверить свой профиль;
3. перейти на страницу расписания, ознакомиться с планом занятий на текущую неделю.

Альтернативные пути и дополнения.

Отсутствуют.

5.3 Ознакомление со статистикой заболеваемости

Цель.

Получение информации по заболеваемости в вузе.

Актеры (действующее лицо).

Студент, преподаватель, администрация

Заинтересованные лица (Stakeholders).

Студент, преподаватель, администрация

Предварительные условия.

Студент должен иметь учетную запись в системе.

Активаторы.

Появление у студента желания ознакомиться с учебной программой своего направления на текущий семестр и проверить состояние заболеваемости в вузе в целом.

Шаги сценария.

1. Авторизоваться в системе;
2. проверить свой профиль;
3. перейти на страницу статистики для ознакомления со статистикой заболеваемости в вузе:
    a. возможность просмотреть статистику по университету в целом;
    b. возможность просмотреть статистику по кафедре, к которой принадлежит студент

Альтернативные пути и дополнения.

Отсутствуют.

5.4 Проведение занятия

Цель.

Выполнение преподавательской деятельности.

Актеры (действующее лицо).

Преподаватель, администрация

Заинтересованные лица (Stakeholders).

Преподаватель, студенты группы

Предварительные условия.

Преподаватель должен иметь учетную запись в системе.

Активаторы.

Желание ознакомиться со своим расписанием на неделю, провести занятие и ознакомиться со статистикой заболеваемости.

Шаги сценария.

1. Авторизоваться в системе;
2. проверить свой профиль;
3. перейти на страницу занятия:
    a. выбрать группу, у которой сейчас проводится занятие;
    b. выбрать текущую дату;
4. проверить, какие из студентов больны и находятся сейчас на дистанционном обучении;
5. проставить посещаемость студентов на начало занятия;
6. провести занятие и проставить баллы в соответствии с активностью и положительными ответами студентов.

Альтернативные пути и дополнения.

Отсутствуют.

5.5 Закрытие кафедры или университета администрацией

Цель.

Ознакомление администрации с заболеваемостью и закрытие при необходимости.

Актеры (действующее лицо).

Администрация

Заинтересованные лица (Stakeholders).

Преподаватели, студенты

Предварительные условия.

Преподаватель должен иметь учетную запись в системе.

Активаторы.

Желание просмотреть статистику заболеваемости по вузу.

Шаги сценария.

1. Авторизоваться в системе;
2. проверить свой профиль;
3. перейти на страницу со статистикой и проверить статистику заболеваемости по университету и по закрепленной кафедре;
4. в случае повышенного уровня заражения закрыть кафедру или университет (зависит от уровня заболеваемости);
5. студенты и преподаватели закрытой кафедры или всего университета переведены на обучение на удаленной основе.

Альтернативные пути и дополнения.

Отсутствуют.

5.6 Заражение студента и уведомление его группы

Цель.

Уведомить однокурсников больного об их потенциальном заражении.

Актеры (действующее лицо).

Преподаватель, заболевший студент

Заинтересованные лица (Stakeholders).

Одногруппники заболевшего студента

Предварительные условия.

1. Все участники имеют доступ к системе;
2. наличие больного студента;
3. наличие однокурсников, которые посещали пары вместе с больным на протяжении 14 дней до дня заболевания;
4. наличие отметок о посещаемости студентов.

Активаторы.

Заболевание студента группы и загрузка в систему соответствующей справки.

Шаги сценария.

1. Авторизация преподавателя в системе;
2. Посещение страницы занятия для простановки посещаемости группы на текущую дату;
3. Авторизация больного студента в системе;
4. Загрузка справки о состоянии болезни в систему;
5. Перевод заболевшего студента на обучение на удаленную основу;
6. Уведомление одногруппников об их возможном заболевании;
7. Проверка одногруппниками состояния своего здоровья:
    a. Если проверенный одногруппник остался здоров, уведомление в системе снимается, студент посещает пары в штатном режиме;
    b. Если проверенный одногруппник оказался болен, он также переводится на обучение на удаленной основе до момента выздоровления.

Альтернативные пути и дополнения.

Отсутствуют.

6 Сценарии тестирования

6.1 Регистрация и авторизация студента

Номер шага	Действие	Ожидаемый результат
1.	Открыть приложение	Приложение открыто
2.	Ввести данные человека	Всплывает уведомление о том, что введены неверные данные
3.	Нажать кнопку «Регистрация»	Открыто новое окно для ввода данных для регистрации
4.	Вводим данные Нажать кнопку «Зарегистрироваться»	Найден пользователь в БД, присвоен новый пароль согласно форме регистрации. Открывается экран с Авторизацией
5.	Вводим данные пользователя, пароль которого был изменен	Пользователь авторизован, открыт экран Профиля Сверху справа написано ФИО Иванов Иван Иванович с его группой, которая в БД На центральной части введены данные соответственно тем, что указывались при регистрации (см. пункт выше) Дата заболевания и дата выхода скрыты Потенциальное заражение отсутствует Слева в панели видны следующие кнопки: Профиль, Моя группа, Расписание, Статистика COVID

6.2 Просмотр группы для студента

Номер шага	Действие	Ожидаемый результат
1.	Авторизоваться в системе (см. Test case «Регистрация и авторизация студента»)	Пользователь авторизован
2.	Слева нажимает кнопку «Моя группа»	Открывается страница группы Отображается следующая информация: *Номер группы* – ***номер группы*** *Количество студентов – целое число* = ***количеству студентов в таблице ниже*** *Куратор* – ***назначенный куратор*** Ниже представлена таблица со следующими столбцами: ФИО, Дата рождения. Данные таблицы – студенты, имеющие аналогичную группу Авторизованному студенту
3.	Нажать на любого из студентов	Открывает всплывающее окно с названием «Информация о выбранном студенте» Показана информация: ФИО, Телефон, Email выбранного студента
4.	Закрываем всплывающее окно	Окно закрыто, видим прежний экран с группой
5.	Авторизоваться в системе (см. Test case «Регистрация и авторизация студента»)	Пользователь авторизован

6.3 Просмотр расписания для студента

Номер шага	Действие	Ожидаемый результат
1.	Авторизоваться в системе (см. Test case «Регистрация и авторизация студента»)	Пользователь авторизован
2.	Нажимаем кнопку «Расписание»	Открыта страница с расписанием Видно 6 таблиц с понедельника по субботу В каждой таблице следующие столбцы: Предмет, ФИО преподавателя, Кабинет, Время начала и окончания Данные таблицы заполнены согласно связанному расписанию для группы Авторизованного студента

6.4 Просмотр статистики для студента

Номер шага	Действие	Ожидаемый результат
1.	Авторизоваться в системе (см. Test case «Регистрация и авторизация студента»)	Пользователь авторизован
2.	Нажимаем кнопку «Статистика COVID»	Открыта страница со статистикой Сверху экрана расположена надпись «Заражено сегодня:» - значение зависит от дня захода в систему По центру экрана расположен график: по горизонтали – дата, по вертикали – количество зараженных на дату Справа поле с раскрывающимся списком, по умолчанию – «Университет»
3.	Нажимаем на поле раскрывающегося списка	Список раскрыт, видны значения: - кафедры - университет
4.	Нажимаем на параметр кафедры	Параметр выбран и показан в поле Поле с выбором сворачивается График изменяется, отображая статистику заболевания непосредственно по группе Количество зараженных сегодня меняется на соответствующее значение
5.	Нажимаем на поле раскрывающегося списка	Список раскрыт, видны значения: - кафедры - университет
6.	Нажимаем на параметр «университет»	Значение зараженных сегодня и вид графика возвращается в положение, в котором они были при открытии страницы
7.	Авторизоваться в системе (см. Test case «Регистрация и авторизация студента»)	Пользователь авторизован
8.	Нажимаем кнопку «Статистика COVID»	Открыта страница со статистикой Сверху экрана расположена надпись «Заражено сегодня:» - значение зависит от дня захода в систему По центру экрана расположен график: по горизонтали – дата, по вертикали – количество зараженных на дату Справа поле с раскрывающимся списком, по умолчанию – «Университет»

6.5 Авторизация преподавателя или администрации

Номер шага	Действие	Ожидаемый результат
1.	Открыть приложение	Приложение открыто
2.	Ввести данные неизвестного человека	Всплывает уведомление о том, что введены неверные данные
3.	Вводим верные данные	Пользователь авторизован, открыт экран Профиля Сверху справа написано ФИО Иванов Иван Иванович На центральной части введены данные соответственно тем, что указывались при регистрации (см. пункт выше) Слева в панели видны следующие кнопки: Профиль, Занятие, Расписание, Статистика COVID

6.6 Ведение занятия для преподавателя или администрации (если пользователь по совместительству преподаватель)

Номер шага	Действие	Ожидаемый результат
1.	Авторизоваться в системе	Пользователь авторизован
2.	Нажать на кнопку «Занятия»	Открыто окно занятия Представлены 2 поля с выпадающим списком: группа и дата. По умолчанию – пустые По центру таблица, содержащая следующие столбцы: Студент, присутствие, баллы По умолчанию таблица пустая
3.	Открываем поле с выпадающим списком группы	Отображены все группы, занятия у которых ведет данный преподаватель
4.	Выбираем любую группу	Группа выбрана, поле с раскрывающимся списком свернуто Таблица по-прежнему пустая
5.	Открываем поле с выпадающим списком даты	Отображены все даты, на которые у преподавателя заполнено расписание
6.	Выбираем любую дату	Дата выбрана, поле с раскрывающимся списком свернуто Таблица заполняется студентами группы, которая была найдена согласно фильтрам, выбранным по полям с раскрывающимся списком Строки без заливки отображают здоровых студентов, строки с красной заливкой отображают больных студентов
7.	Нажимаем на чек-боксы студентов для проставления их присутствия	После каждого выбранного чек-бокса данные автоматически отправляются в БД Чек-бокс визуально меняет свое значение
8.	Снимаем чек-бокс с пары студентов	В БД отправлены новые данные Чек-бокс визуально меняет свое значение
9.	Проставляем баллы студентам	После каждого заполненного значения данные отправляются в БД
10.	Уберем баллы у пары студентов	В БД отправлены новые данные
11.	Нажимаем «Сохранить»	Показано количество изменений, значения сохранены

6.7 Просмотр расписания для преподавателя или администрации (если пользователь по совместительству преподаватель)

Номер шага	Действие	Ожидаемый результат
1.	Авторизоваться в системе	Пользователь авторизован
2.	Нажимаем кнопку «Расписание»	Открыта страница с расписанием Видно 6 таблиц с понедельника по субботу В каждой таблице следующие столбцы: Предмет, Группа, Кабинет, Время начала и окончания Данные таблицы заполнены согласно связанным парам для Авторизованного преподавателя

6.8 Просмотр статистики для преподавателя

Номер шага	Действие	Ожидаемый результат
1.	Авторизоваться в системе	Пользователь авторизован
2.	Нажимаем кнопку «Статистика COVID»	Открыта страница со статистикой Сверху экрана расположена надпись «Заражено сегодня:» - значение зависит от дня захода в систему По центру экрана расположен график: по горизонтали – дата, по вертикали – количество зараженных на дату Справа поле с раскрывающимся списком, по умолчанию – «Университет»
3.	Нажимаем на поле раскрывающегося списка	Список раскрыт, видны значения: - кафедра - университет
4.	Нажимаем на параметр «кафедра»	Параметр выбран и показан в поле Поле с выбором сворачивается График изменяется, отображая статистику заболевания по кафедре, к которой привязан студент Количество зараженных сегодня меняется на соответствующее значение
5.	Нажимаем на поле раскрывающегося списка	Список раскрыт, видны значения: - кафедра - университет
6.	Нажимаем на параметр «университет»	Значение зараженных сегодня и вид графика возвращается в положение, в котором они были при открытии страницы

6.9 Просмотр статистики для администрации

Номер шага	Действие	Ожидаемый результат
1.	Авторизоваться в системе	Пользователь авторизован
2.	Нажимаем кнопку «Статистика COVID»	Открыта страница со статистикой Сверху экрана расположена надпись «Заражено сегодня:» - значение зависит от дня захода в систему По центру экрана расположен график: по горизонтали – дата, по вертикали – количество зараженных на дату Справа поле с раскрывающимся списком, по умолчанию – «Университет»
3.	Нажимаем на поле раскрывающегося списка	Список раскрыт, видны значения: - кафедра - университет
4.	Нажимаем на параметр «кафедра»	Параметр выбран и показан в поле Поле с выбором сворачивается График изменяется, отображая статистику заболевания по кафедре, к которой привязан студент Количество зараженных сегодня меняется на соответствующее значение
5.	Нажимаем на поле раскрывающегося списка	Список раскрыт, видны значения: - кафедра - университет
6.	Нажимаем на параметр «университет»	Значение зараженных сегодня и вид графика возвращается в положение, в котором они были при открытии страницы

6.10 Заражение студента и уведомление однокурсников об их потенциальном заражении

Номер шага	Действие	Ожидаемый результат
1.	Авторизоваться в системе под преподавателем	Пользователь преподавателя авторизован
2.	Нажимаем на кнопку «Занятие»	Открыт экран Занятия
3.	Выбираем дату и группу	Дата и группа выбраны В таблице отображены студенты выбранной группы на занятие выбранной даты
4.	Проставить посещение для Студента 1 и Студента 2	
5.	Выйти из приложения	Открыт экран Авторизации
6.	Авторизоваться в системе под пользователем Студента 1	Пользователь Студента 1 авторизован
7.	В профиле нажать на кнопку «Обновить статус заболевания»	Открыто всплывающее окно, названное «Обновление статуса заболевания COVID» В центре отображен график заболеваемости по всему университету По горизонтали – дата, по вертикали – количество зараженных Также видны 2 кнопки: «Я здоров» и «Я болен»
8.	Нажимаем на кнопку «Я болен»	Открывается всплывающее окно, названное «Загрузка документов». Видна надпись «Вы загружаете следующие документы:», список документов пуст Присутствует видимые кнопки «Выбрать документы»
9.	Нажимаем на кнопку «Выбрать документы»	Открывается системное окно для выбора документов для загрузки
10.	Выбираем документ «я болен.pdf»	Документ загружен, его имя появляется в списке выбранных документов Появляется кнопка «Отправить»
11.	Нажимаем на кнопку «Отправить»	Студент получает статус больного в БД, всем однокурсникам проставляется необходимость в проверке Всплывающие окна закрываются, видны надписи «Дата заболевания» - сегодня, «Дата выхода» - пусто
12.	Выйти из приложения	Открыт экран Авторизации
13.	Авторизоваться в системе под пользователем	Пользователь Студента 2 авторизован В окне открывающегося профиля сверху видна надпись «Потенциальное заражение»

	Студента 2	
14.	В профиле нажать на кнопку «Обновить статус заболевания»	Открыто всплывающее окно, названное «Обновление статуса заболевания COVID» В центре отображен график заболеваемости по всему университету По горизонтали – дата, по вертикали – количество зараженных Также видны 2 кнопки: «Я здоров» и «Я болен»
15.	Нажимаем на кнопку «Я здоров»	Открывается всплывающее окно, названное «Загрузка документов». Видна надпись «Вы загружаете следующие документы:», список документов пуст Присутствует видимые кнопки «Выбрать документы»
16.	Нажимаем на кнопку «Выбрать документы»	Открывается системное окно для выбора документов для загрузки
17.	Выбираем документ «я здоров.pdf»	Документ загружен, его имя появляется в списке выбранных документов Появляется кнопка «Отправить»
18.	Нажимаем на кнопку «Отправить»	Студент получает статус здорового в БД

7 Прототипы основных окон системы

7.1 Профиль

Данная страница (см. рисунок 2) предназначена для просмотра профиля пользователя системы (студент, преподаватель и т. д.). На данной странице он может увидеть основную информацию о себе в рамках университета, а также ознакомиться с номером своей группы.

Также студенту доступна кнопка отметки о болезни, которую необходимо использовать при первом получении положительного теста на COVID. В случае, если один из студентов был заражен, его одногруппнику в верхней части экрана появится уведомление о необходимости пройти проверку (уведомление с красной заливкой). После прохождение проверки и загрузки документов, уведомление пропадает с интерфейса.

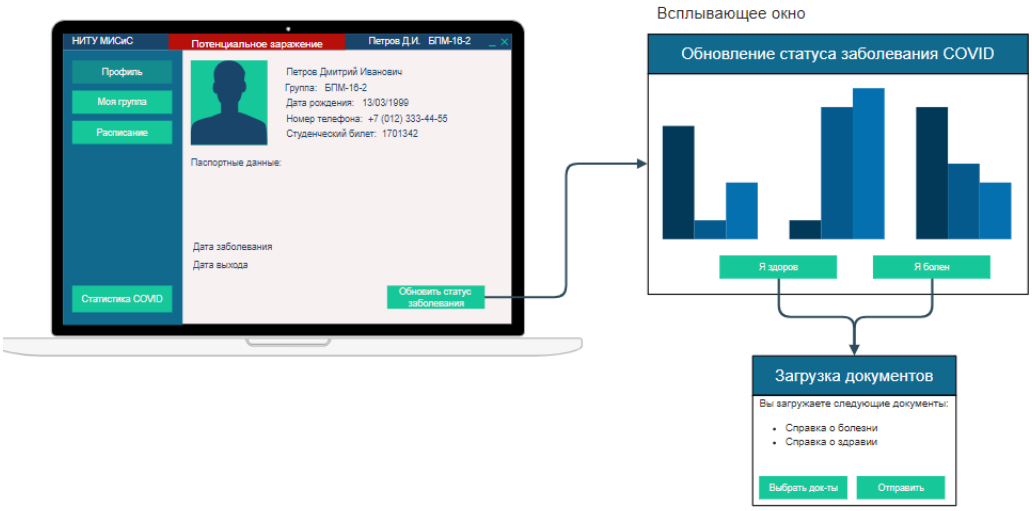

Рисунок 2 - Страница Профиля пользователя для всех ролей

7.2 Статистика

Данная страница предназначена для просмотра степени заболеваемости на разных уровнях. В зависимости от роли меняется наличие уровней, которые пользователь системы может просматривать, а также отображается общее количество зараженных по всему университету. Для студента (см. рисунок 3) доступен просмотр следующих уровней: группа, кафедра, университет.

Рисунок 3 - Страница Статистики для роли Студент

Для ролей Преподаватель и Ректорат/деканат (см. рисунок 4) доступен просмотр следующих уровней: кафедра и университет. Группа исключена, так как данная роль не привязана к какой-либо конкретной группе.

Рисунок 4 - Просмотр статистики для ролей Преподаватель и Ректорат/деканат

7.3 Расписание

Данная страница (см. рисунок 5) отображает расписание пользователя на текущую неделю. В зависимости от роли отображается в следующем виде:

- для роли Студента отображается расписание для пользователя на ближайшую неделю с отображением преподавателя;
- для роли Преподавателя отображается его расписание на ближайшую неделю с группами, у которых необходимо провести занятие.

Рисунок 5 - Страница просмотра Расписания для всех ролей

7.4 Занятия

Данная страница (см. рисунок 6) предназначена для Преподавателя, заполняется в процессе проведения семинаров или лекций.

Прежде всего, на интерфейсе находится два выпадающих списка, которые определяют фильтр для отображения в таблице группы, у которой будет проведено занятие:

- группа - для выбора группы, у которой проводим занятие;
- дата - дата проведения занятия.

В отображенной таблице показываются студенты группы. Красным цветом выделены студенты, зараженные COVID и обучающиеся на удаленной основе. Есть возможность проставить посещение полем с галочкой и проставить баллы в целочисленном значении каждому из студентов.

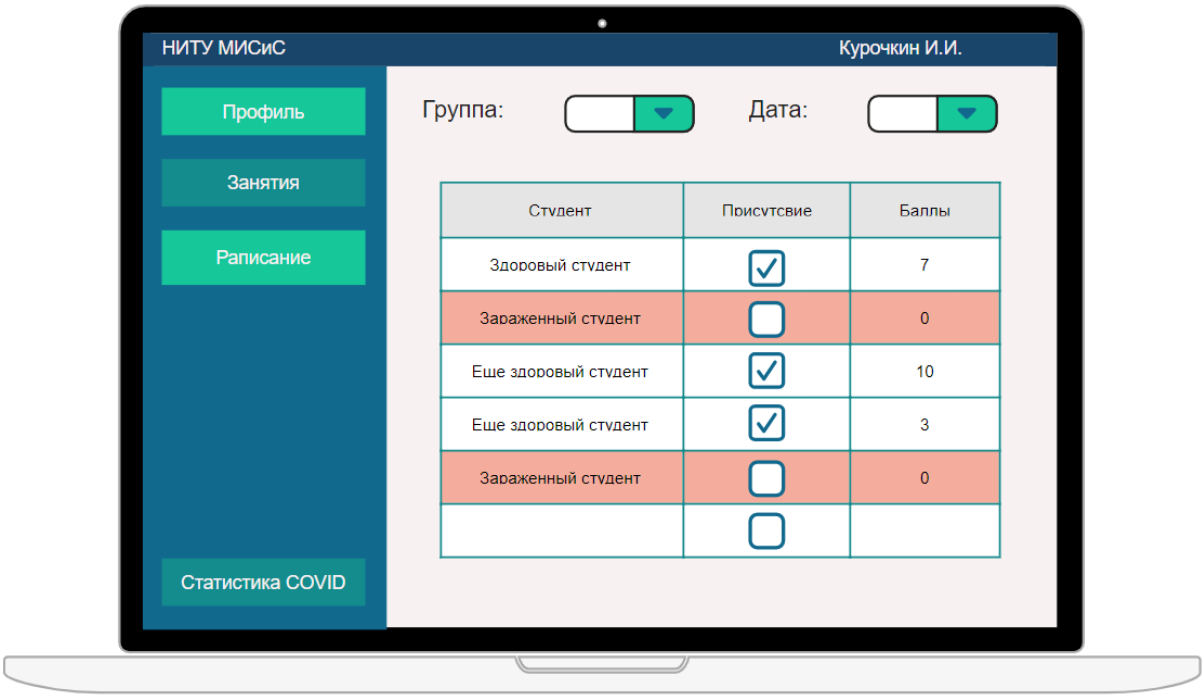

Рисунок 6 - Страница проведения Занятия для ролей Преподаватель и Ректорат/деканат

7.5 Группа

Данная страница (см. рисунок 7) необходима для отображения одногруппников пользователя с ролью Студент, общего количества учащихся в данной группе и куратора. При нажатии на одногруппника или куратора открывается всплывающее окно, в котором отображается основная информация о выбранном пользователе.

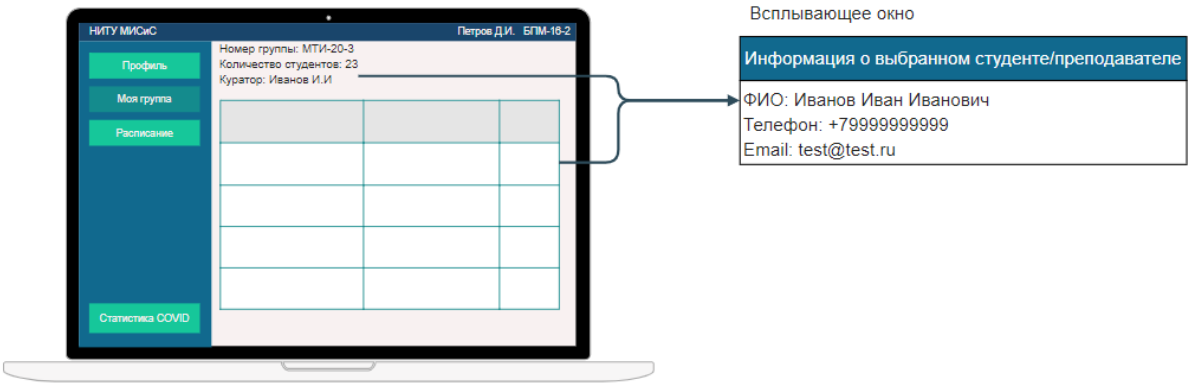

Рисунок 7 - Просмотр Группы для роли Студент

7.6 Авторизация и регистрация

На странице Авторизации (см. рисунок 8) пользователю предлагается ввести данные учетной записи для входа в систему и выполнения необходимых действий, а также предлагается зарегистрироваться в системе в случае отсутствия аккаунта. В случае введения неверных пользовательских данных, будет выведено сообщение об ошибке.

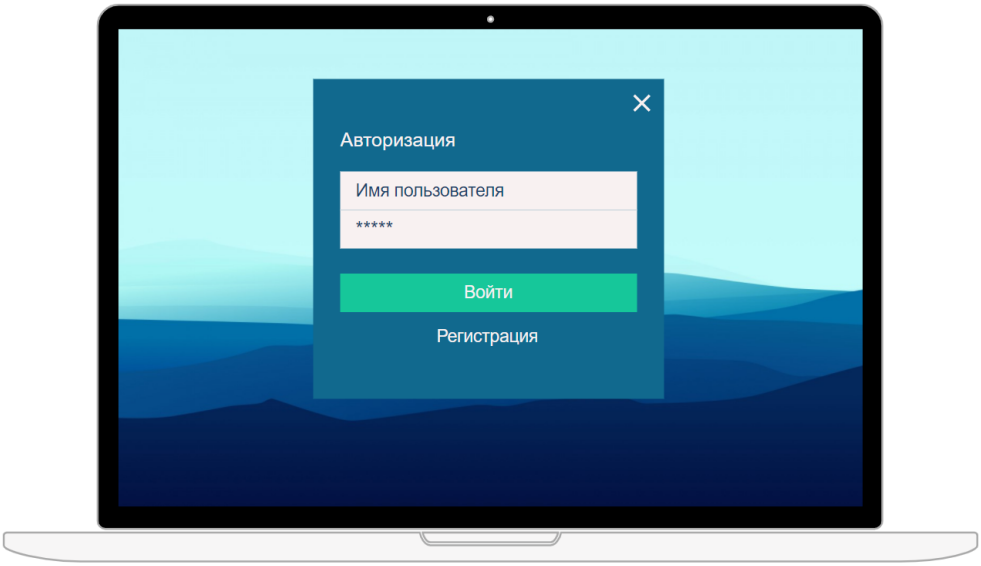

Рисунок 8 - Страница Авторизации для всех пользователей

На странице Регистрации (см. рисунок 9) пользователю предлагается ввести свои основные данные, которые необходимы для создания страницы пользователя в системе, после чего в БД появится новая запись. После регистрации пользователь возвращается на страницу авторизации для входа в систему.

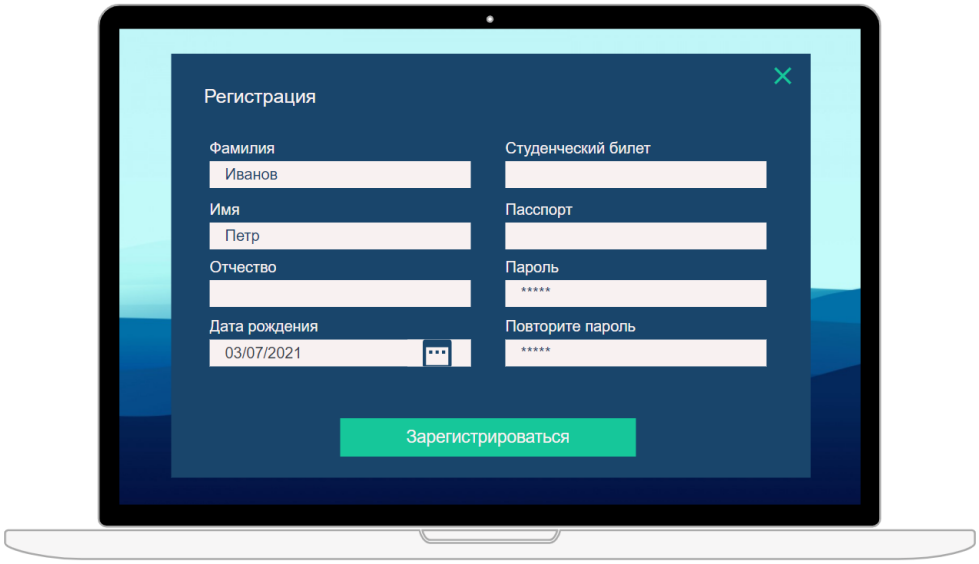

Рисунок 9 - Страница Регистрации для всех пользователей

8 Код

8.1 Клиентское приложение

StudentNavigation

```
using Goosle.Views;
using System.Windows;
using System.Windows.Controls;
using MySql.Data.MySqlClient;
using Goosle.Navigations;
using Goosle.Models;
using System.Data;

namespace Goosle
{
    /// <summary>
    /// Логика взаимодействия для StudentNavigation.xaml
    /// </summary>
    public partial class StudentNavigation : Page, IUserNavigation
    {
        MySqlConnection conn;
        private readonly UserModel user;

        public StudentNavigation(MySqlConnection connection, UserModel user)
        {
            InitializeComponent();
            conn = connection;
            this.user = user;
            if (RED_BUTTON(conn, user) == false)
                this.RED.Visibility = Visibility.Hidden;
            ChangeHeaderName($"{user.Surname} {user.Name[0]}. {user.Patronymic[0]}.");
            StudentFrame.Navigate(new Profile(conn, user));

        }

        private void Group(object sender, RoutedEventArgs e)
        {
            StudentFrame.Navigate(new Group(conn, user.Entry_Card));
        }

        public void ChangeHeaderName(string value)
        {
            NameField.Text = value;
        }

        public void Profile_Click(object sender, RoutedEventArgs e)
        {
            StudentFrame.Navigate(new Profile(conn, user));
        }

        public void Shedule_Click(object sender, RoutedEventArgs e)
        {
            StudentFrame.Navigate(new SheduleView(conn, user.id_person));
        }

        public void CovidStats_Click(object sender, RoutedEventArgs e)
        {
```

```
            StudentFrame.Navigate(new StatsView(conn, user));
        }

        public bool RED_BUTTON(MySqlConnection conn, UserModel user)
        {
            MySqlCommand cmd = new MySqlCommand("select ShouldCheck from COVID where id_person = " + user.id_person, conn);
            MySqlDataAdapter da = new MySqlDataAdapter(cmd);
            DataTable dt = new DataTable();
            da.Fill(dt);

            if (dt.Rows.Count == 0)
                return false;
            else if (dt.Rows[0].Field<bool>("ShouldCheck") != true)
                return false;
            else
                return true;
        }
        private void buttonClose_Click(object sender, RoutedEventArgs e)
        {
            Application.Current.MainWindow.Close();
        }

        private void buttonLogOut_Click(object sender, RoutedEventArgs e)
        {
            Application.Current.MainWindow.Hide();
            Application.Current.MainWindow = new MainWindow();
        }
    }
}
```

TeacherNavigation

```
using Goosle.Models;
using Goosle.Views;
using MySql.Data.MySqlClient;
using System.Windows;
using System.Windows.Controls;

namespace Goosle.Navigations
{
    /// <summary>
    /// Логика взаимодействия для TeacherNavigation.xaml
    /// </summary>
    public partial class TeacherNavigation : Page, IUserNavigation
    {
        MySqlConnection connection;
        private readonly UserModel user;

        public TeacherNavigation(MySqlConnection mySqlConnection, UserModel user)
        {
            connection = mySqlConnection;
            this.user = user;
            InitializeComponent();
            ChangeHeaderName($"{user.Surname} {user.Name[0]}. {user.Patronymic[0]}.");
            TeacherFrame.Navigate(new Profile(connection, user));
        }

        private void Lesson_Click(object sender, RoutedEventArgs e)
        {
            TeacherFrame.Navigate(new LessonView(connection));
        }

        public void ChangeHeaderName(string value)
        {
            NameField.Text = value;
        }

        public void Profile_Click(object sender, RoutedEventArgs e)
        {
            TeacherFrame.Navigate(new Profile(connection, user));
        }

        public void Shedule_Click(object sender, RoutedEventArgs e)
        {
            TeacherFrame.Navigate(new SheduleView(connection, user.id_person));
        }

        public void CovidStats_Click(object sender, RoutedEventArgs e)
        {
            TeacherFrame.Navigate(new StatsView(connection, user));
        }

        private void buttonClose_Click(object sender, RoutedEventArgs e)
        {
            Application.Current.MainWindow.Close();
        }

        private void buttonLogOut_Click(object sender, RoutedEventArgs e)
        {
```

```
            Application.Current.MainWindow.Hide();
            Application.Current.MainWindow = new MainWindow();
        }
    }
}
```

Group

```
using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using MySql.Data.MySqlClient;
using System.Data;



namespace Goosle.Views
{
   /// <summary>
   /// Логика взаимодействия для Group1.xaml
   /// </summary>
   public partial class Group : Page
   {
      MySqlConnection conn;
      string LoginName = "s123456";
      MySqlCommand cmd;


      public Group(MySqlConnection connection, string login)
      {
         InitializeComponent();
         LoginName = login;
         conn = connection;
         cmd = new MySqlCommand("SELECT Name, Surname, Patronymic, Birthday, Age from (SELECT Name, Surname, Patronymic,
Birthday, Age, id_group from Students S Inner JOIN People P on S.id_person = P.id_person) as T where(T.id_group IN(SELECT
T1.id_group from(SELECT id_group, Entry_Card from Students S Inner JOIN People P on S.id_person = P.id_person) as T1 where
T1.Entry_Card = '" +
            LoginName + "')); ", conn);
         MySqlDataAdapter da = new MySqlDataAdapter(cmd);
         DataTable dt = new DataTable();
         da.Fill(dt);


         List<MyTable> studentList = new List<MyTable>();
         for (int i = 0; i < dt.Rows.Count; i++)
         {
            MyTable student = new MyTable();
            //student.StudentId = Convert.ToInt32(dt.Rows[i]["StudentId"]);
            student.Name = dt.Rows[i]["Name"].ToString();
            student.Surname = dt.Rows[i]["Surname"].ToString();
            student.Patronymic = dt.Rows[i]["Patronymic"].ToString();
            student.Birthday = dt.Rows[i]["Birthday"].ToString();
            student.Age = dt.Rows[i]["Age"].ToString();
            studentList.Add(student);
         }
         //System.Console.WriteLine(studentList.);
         groupGrid.ItemsSource = studentList;



         //Поменяю на запросы гогда будут заполнены таблицы
         //cmd1 = new MySqlCommand("SELECT * FROM Students", conn);
```

```
            //MySqlDataAdapter da1 = new MySqlDataAdapter(cmd1);
            //da1 = new MySqlDataAdapter(cmd1);
            //DataTable dt1 = new DataTable();
            //da1.Fill(dt1);
            GroupNumber.Text = "Номер группы: МПИ-20-4-2";

            Quantity.Text = "Количество студентов: " + studentList.Count;
            Sensei.Text = "Куратор: Широков Андрей Иванович";
            ////Gop.Text = dt.Rows[0].Field<string>("Name");

        }
        //Получаем данные из таблицы
        private void grid_MouseUp(object sender, MouseButtonEventArgs e)
        {
            MyTable path = groupGrid.SelectedItem as MyTable;
            AdditionalInformation form = new AdditionalInformation("ФИО: " + path.Name + " " + path.Surname + " " + path.Patronymic +
                "\nТелефон: +79268735239" + "\nEmail: stdem@mail.ru");
            form.ShowDialog();
            //MessageBox.Show("Информация о выбранном студенте/преподавателе\n" + "ФИО: " + path.Name + " " + path.Surname + "
" + path.Patronymic +
            //    "\nТелефон: +79268735239" + "\nEmail: stdem@mail.ru");

        }

        private void dataGrid_AutoGeneratingColumn(object sender, DataGridAutoGeneratingColumnEventArgs e)
        {
            if (e.PropertyName.Contains("ID") || e.PropertyName.Contains("id_t") || e.PropertyName.Contains("id_kid") ||
e.PropertyName.Contains("id_bab") || e.PropertyName.Contains("id_pa") || e.PropertyName.Contains("id_adm") || e.PropertyName ==
"id_Schedule")
                e.Column.Visibility = Visibility.Hidden;

            if (e.PropertyType == typeof(DateTime))
                (e.Column as DataGridTextColumn).Binding.StringFormat = "dd.MM.yyyy";

            e.Column.Width = DataGridLength.Auto;
            //e.Column.Width = (dataGrid.Width - 19) / 8;

            if (e.PropertyName == "Name")
                e.Column.Header = "Имя";
            if (e.PropertyName.Contains("Surname"))
                e.Column.Header = "Отчество";
            if (e.PropertyName.Contains("Pytronamic"))
                e.Column.Header = "Отчество";
            if (e.PropertyName.Contains("Age"))
                e.Column.Header = "Возраст";
            if (e.PropertyName.Contains("Passport"))
                e.Column.Header = "Пасспорт";
            if (e.PropertyName.Contains("Birthday"))
                e.Column.Header = "Дата рождения";
            if (e.PropertyName.Contains("TelephoneNumber"))
                e.Column.Header = "Телефон";
            if (e.PropertyName.Contains("name_tutor"))
                e.Column.Header = "Имя";
            if (e.PropertyName.Contains("name_baby"))
                e.Column.Header = "Имя";
            if (e.PropertyName.Contains("name_kid"))
                e.Column.Header = "Имя";
            if (e.PropertyName == "name_administrator")
                e.Column.Header = "Имя";
```

```
            if (e.PropertyName.Contains("surname"))
                e.Column.Header = "Фамилия";
            if (e.PropertyName.Contains("group"))
                e.Column.Header = "Группа";
            if (e.PropertyName.Contains("work"))
                e.Column.Header = "Должность";
            e.Column.Width = DataGridLength.Auto;
        }

        private void DataGrid_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {

        }
        class MyTable
        {
            //public MyTable(string Name, string Surname,  string Patronymic, string Birthday, string Age)
            //{
            //    this.Name = Name;
            //    this.Surname = Surname;
            //    this.Patronymic = Patronymic;
            //    this.Birthday = Birthday;
            //    this.Age = Age;
            //}
            public string Name { get; set; }
            public string Surname { get; set; }
            public string Patronymic { get; set; }
            public string Birthday { get; set; }
            public string Age { get; set; }
        }
    }
}
```

AdditionalInformation

```
using System;
using System.Windows;
using System.Windows.Input;

namespace Goosle
{
    /// <summary>
    /// Логика взаимодействия для AdditionalInformation.xaml
    /// </summary>
    public partial class AdditionalInformation : Window
    {
        public AdditionalInformation(string text)
        {
            InitializeComponent();
            textBox.Text = text;
        }

        private void buttonClose_Click(object sender, RoutedEventArgs e)
        {
            Close();
        }

        // Позволяет перетаскивать окно программы по экрану зажатой левой кнопкой мыши
        private void grid_MouseDown(object sender, MouseButtonEventArgs e)
        {
            try
            {
                DragMove();
            }
            catch (InvalidOperationException) { }
        }
    }
}
```

Registration

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Windows;
using System.Windows.Input;

namespace Goosle
{
    /// <summary>
    /// Логика взаимодействия для RegistrationWindow.xaml
    /// </summary>
    public partial class RegistrationWindow : Window
    {

        public RegistrationWindow()
        {
            InitializeComponent();
        }

        // Вызывается при нажатии крестика. Закрывает окно регистрации.
        private void buttonClose_Click(object sender, RoutedEventArgs e)
        {
            Close();
        }

        // Позволяет перетаскивать окно программы по экрану зажатой левой кнопкой мыши
        private void grid_MouseDown(object sender, MouseButtonEventArgs e)
        {
            try
            {
                DragMove();
            }
            catch (InvalidOperationException) { }
        }



        //Проверка полей регистрации на выполнние требований:
        //ФИО только русские буквы, Пасспорт только цифры, Номер пропуска только из s,t,a и набора цифр, пароли совпадают
        public bool checkInputs(string Surname, string UserName, string Patronymic, string Passport, string EntryCard, string Pass1, string
Pass2)
        {
            if (Surname == "" || UserName == "" || Patronymic == "" || Passport == "" || EntryCard == "" || Pass1 == "" || Pass2 == "")
            {
                MessageBox.Show("Все поля обязательны к заполнению!");
                return false;
            }
            List<string> values = new List<string>();
            if (CheckRegProp.checkStringInputs(Surname))
                values.Add("Фамилия (возможны только рус. буквы)\n");
            if (CheckRegProp.checkStringInputs(UserName))
                values.Add("Имя (возможны только рус. буквы)\n");
            if (CheckRegProp.checkStringInputs(Patronymic))
                values.Add("Отчество (возможны только рус. буквы)\n");
            if (CheckRegProp.checkNumInputs(Passport))
```

```
            values.Add("Пасспорт (должны быть только цифры)\n");
        if (CheckRegProp.checkEntryCard(EntryCard))
            values.Add("Номер пропуска ( должен содержать в начале s,t,a и потом набор цифр)\n");
        if (Pass1 != Pass2)
            values.Add("Пароли не совпадают");

        if (values.Count != 0)
        {
            MessageBox.Show("Ошибки в данных: \n" + String.Join("", values));
            return false;
        }
        else
            return true;
    }

    //Функция при нажатии кнопки "Зарегистрироваться", проверяет корректность ввода и отправляет запрос на регистрацию в
БД
    private void buttonRegistration_Click(object sender, RoutedEventArgs e)
    {
        string Surname = textBoxSurname.Text;
        string UserName = textBoxName.Text;
        string Patronymic = textBoxPatronymic.Text;
        string BirthDate = dataSelector.Text;
        string Passport = textBoxPassport.Text;
        string EntryCard = textBoxEntryCard.Text;
        string Pass1 = passwordBox1.Password;
        string Pass2 = passwordBox1.Password;

        if (checkInputs(Surname, UserName, Patronymic, Passport, EntryCard, Pass1, Pass2))
        {
            string connString = "Server=" + "db-goosle.ck2ojbqqmm8s.eu-central-1.rds.amazonaws.com" +
            ";Database=" + "goosleDB"
            + ";port=" + "3306" + ";User Id=" + "a100001" + ";password=" + "12345";
            MySqlConnection conn = new MySqlConnection(connString);
            try
            {
                conn.Open();
                //Проверяем есть ли в БД строка с такими данными
                MySqlCommand command = new MySqlCommand($"SELECT Surname, Name,  Patronymic, Birthday, Passport " +
                    $"FROM People WHERE Entry_Card='{EntryCard}'", conn);
                MySqlDataAdapter userAdapter = new MySqlDataAdapter(command);
                DataTable CacheData = new DataTable();
                userAdapter.Fill(CacheData);
                //Если все совпало, то можно установить пароль
                if (CacheData.Rows[0][0].ToString() == Surname && CacheData.Rows[0][1].ToString() == UserName
                    && CacheData.Rows[0][2].ToString() == Patronymic && CacheData.Rows[0][4].ToString() == Passport)
                {
                    command = new MySqlCommand($"UPDATE People SET Pass='{Pass1}' WHERE Entry_Card='{EntryCard}'", conn);
                    command.ExecuteNonQuery();
                }
                conn.Close();
                MessageBox.Show("Регистрация прошла успешно!\nЛогин: "+EntryCard+"\nПароль: " +Pass1);
                this.Close();
            }
            catch
            {
                MessageBox.Show("Данные не совпадают с информацией в системе.");
            }
        }
```

```
    }

    // Исправляет ввод в полях фамилия, имя, отчество: первую букву делает заглавной, остальные строчными
    // Для составных фио также делает заглавной букву в каждой части
    private void FirstUpperCase(object sender, RoutedEventArgs e)
    {
        try
        {
            textBoxSurname.Text = textBoxSurname.Text.Replace(" ", "");
            string[] strParts = textBoxSurname.Text.Split('-');
            textBoxSurname.Text = "";
            for (int i = 0; i < strParts.Length; i++)
            {
                textBoxSurname.Text = textBoxSurname.Text + strParts[i].First().ToString().ToUpper() +
strParts[i].ToLower().Substring(1);
                if (i < strParts.Length - 1)
                    textBoxSurname.Text = textBoxSurname.Text + "-";
            }

            textBoxName.Text = textBoxName.Text.Replace(" ", "");
            strParts = textBoxName.Text.Split('-');
            textBoxName.Text = "";
            for (int i = 0; i < strParts.Length; i++)
            {
                textBoxName.Text = textBoxName.Text + strParts[i].First().ToString().ToUpper() + strParts[i].ToLower().Substring(1);
                if (i < strParts.Length - 1)
                    textBoxName.Text = textBoxName.Text + "-";
            }

            textBoxPatronymic.Text = textBoxPatronymic.Text.Replace(" ", "");
            strParts = textBoxPatronymic.Text.Split('-');
            textBoxPatronymic.Text = "";
            for (int i = 0; i < strParts.Length; i++)
            {
                textBoxPatronymic.Text = textBoxPatronymic.Text + strParts[i].First().ToString().ToUpper() +
strParts[i].ToLower().Substring(1);
                if (i < strParts.Length - 1)
                    textBoxPatronymic.Text = textBoxPatronymic.Text + "-";
            }
        }
        catch
        {

        }
    }

    private void DelSpaces(object sender, RoutedEventArgs e)
    {
        try
        {
            textBoxPassport.Text = textBoxPassport.Text.Replace(" ", "");
            textBoxEntryCard.Text = textBoxEntryCard.Text.Replace(" ", "");
            textBoxEntryCard.Text = textBoxEntryCard.Text.First().ToString().ToLower() + textBoxEntryCard.Text.Substring(1);
        }
        catch
        {

        }
```

```
        }
    }

}
```

Login

```
using System;
using System.Data;
using System.Windows;
using System.Windows.Input;
using Goosle.Models;
using MySql.Data.MySqlClient;

namespace Goosle
{
    /// <summary>
    /// Логика взаимодействия для LoginWindow.xaml
    /// В окне авторизации пользователь вводит свои логин и пароль в соответствующие текстовые поля. Текст пароля скрыт точками.
    /// Затем нажимается кнопка Войти. Если подлкючение к БД пройдет успешно, то окно закрывается и открывается главное окно.
    /// Если подключение не удалось, то появится уведомление.
    /// Если пользватель не имеет учетной записи, то он нажимает конпку Регистрация, открывающую соответсвующее окно.
    /// При нажатии на крестик программа завершает работву.
    /// </summary>
    public partial class LoginWindow : Window
    {
        bool conn_status = false;
        MySqlConnection conn;
        string Login_name;
        private UserModel user;

        public LoginWindow()
        {
            InitializeComponent();
        }

        internal UserModel GetUser()
        {
            return user;
        }

        // Функция при нажатия кнопки Закрыть (крестик)
        // При нажатии закрывает приложение
        private void buttonClose_Click(object sender, RoutedEventArgs e)
        {
            if (Application.Current.Windows.Count > 1)
                for (int i = 0; i < Application.Current.Windows.Count; i++)
                    Application.Current.Windows[i].Close();
            Close();
            //Application.Current.MainWindow.Close();
        }

        // Функция возращает значение переменноц conn_failed
        // Если подключение к БД не вышло - true, иначе - false
        // Используется в главном окне, чтобы закрыть приложение при неудачном подключении к БД
        internal bool GetStatus()
        {
            return conn_status;
        }
```

```
        // Функция возвращает соединение с БД
        // Нужна для передачи доступа к БД в другие окна приложения
        internal MySqlConnection GetConnection()
        {
            return conn;
        }

        internal string GetLogin()
        {
            return Login_name;
        }

        // Функция при нажатия кнопки Войти
        // При нажатии производится попытка подключения к БД
        // Если удается подключение, то форма закрывается и открывается главное окно
        private void buttonLogin_Click(object sender, RoutedEventArgs e)
        {
            String connString = "Server=" + "db-goosle.ck2ojbqqmm8s.eu-central-1.rds.amazonaws.com" +
                ";Database=" + "goosleDB"
                + ";port=" + "3306" + ";User Id=" + "a100001" + ";password=" + "12345";
            conn = new MySqlConnection(connString);
            Login_name = textBoxLogin.Text.Trim();

            try
            {
                conn.Open();
                user = getUser(Login_name, passwordBox.Password);
                conn_status = true;
            }
            catch (Exception ex) when (ex is MySqlException || ex is IndexOutOfRangeException)
            {
                MessageBox.Show("Неверный логин или пароль!", "Ошибка авторизации", MessageBoxButton.OK,
MessageBoxImage.Error);
            }
            conn.Close();
            if (conn_status)
            {
                Close();
            }
        }

        // Функция при нажатия кнопки Регистрация
        // При нажатии откроет форму регистрации
        private void ButtonReg_Click(object sender, RoutedEventArgs e)
        {
            RegistrationWindow form_reg = new RegistrationWindow();
            form_reg.ShowDialog();
        }

        // Позволяет перетаскивать окно программы по экрану зажатой левой кнопкой мыши
        private void grid_MouseDown(object sender, MouseButtonEventArgs e)
        {
            try
            {
                DragMove();
            }
            catch (InvalidOperationException) { }
        }
```

```
        // Функция срабатывает при получении фокуса на textBoxLogin (поле для логина)
        // удалает "Имя пользователя", освобождая место для ввода логина
        private void textBoxLogin_GotFocus(object sender, RoutedEventArgs e)
        {
            if (textBoxLogin.Text == "Имя пользователя")
                textBoxLogin.Text = "";
        }

        // Функция срабатывает при потере фокуса на textBoxLogin (поле для логина)
        // Если поле было пустое (т.е. пользователь получал фокус, но не вводил логин), то возвращается надпись "Имя пользователя"
        private void textBoxLogin_LostFocus(object sender, RoutedEventArgs e)
        {
            if (textBoxLogin.Text == "")
                textBoxLogin.Text = "Имя пользователя";
        }

        // Функция срабатывает при получении фокуса на passowrdBox (поле для пароля)
        // удалает дефолтные звездочки, освобождая место для ввода пароля
        private void passwordBox_GotFocus(object sender, RoutedEventArgs e)
        {
            if (passwordBox.Password == "Пароль")
                passwordBox.Password = "";
        }

        // Функция срабатывает при потере фокуса на passowrdBox (поле для пароля)
        // Если поле было пустое (т.е. пользователь получал фокус, но не вводил пароль), то возвращаются дефолтные звездочки
        private void passwordBox_LostFocus(object sender, RoutedEventArgs e)
        {
            if (passwordBox.Password == "")
                passwordBox.Password = "Пароль";
        }

        /// <summary>
        /// Возвращает UserModel с инициализированными полями из базы данных.
        /// Не вызывать при отсутствии пользователей.
        /// </summary>
        /// <param name="login">Логин пользователя.</param>
        /// <param name="pass">Пароль позователя.</param>
        /// <exception cref="IndexOutOfRangeException">Выбрасывается при отсутствии пользователя с таким логином и паролем.</exception>
        private UserModel getUser(string login, string pass)
        {
            var user = new MySqlCommand($"SELECT * FROM People WHERE Entry_Card='{login}' AND Pass='{pass}' LIMIT 1", conn);
            var userAdapter = new MySqlDataAdapter(user);
            var userTable = new DataTable();
            userAdapter.Fill(userTable);
            return new UserModel()
            {
                id_person = int.Parse(userTable.Rows[0]["id_person"].ToString()),
                id_department = int.Parse(userTable.Rows[0]["id_department"].ToString()),
                Name = $"{userTable.Rows[0]["Name"]}",
                Surname = $"{userTable.Rows[0]["Surname"]}",
                Patronymic = $"{userTable.Rows[0]["Patronymic"]}",
                Birthday = $"{userTable.Rows[0]["Birthday"]}",
                Age = $"{userTable.Rows[0]["Age"]}",
                Passport = $"{userTable.Rows[0]["Passport"]}",
                TelephoneNumber = $"{userTable.Rows[0]["TelephoneNumber"]}",
```

```
            Photo = $"{userTable.Rows[0]["Photo"]}",
            Entry_Card = $"{userTable.Rows[0]["Entry_Card"]}",
            Pass = $"{userTable.Rows[0]["Pass"]}",
            Role = $"{userTable.Rows[0]["Role"]}",
        };
    }
  }
}
```

CheckRegProp

```
using System.Linq;
using System.Text.RegularExpressions;

namespace Goosle
{
   public class CheckRegProp
   {
      //Проверка полей. Возвращает true, если будет найдены цифры или символы не русккого алфавита
      public static bool checkStringInputs(string userInfo)
      {
         int hasNonRussianLetters = Regex.Matches(Regex.Replace(userInfo, @"\W", ""), @"\P{IsCyrillic}").Count;
         if (userInfo.Any(char.IsDigit) ||
            userInfo.Replace("-", "").Any(char.IsPunctuation) ||
            userInfo.Replace("-", "").Any(char.IsControl) ||
            userInfo.Replace("-", "").Any(char.IsSymbol) ||
            userInfo.Replace(" ", "").Length == 0 ||
            hasNonRussianLetters > 0)
         {
            return true;
         }
         else
            return false;
      }
      //Проверка полей. Возвращает true, если будет найдены не числовые символы
      public static bool checkNumInputs(string userInfo)
      {
         if (!userInfo.Replace(" ", "").All(char.IsDigit) || userInfo.Replace(" ", "").Length == 0)
         {
            return true;
         }
         else
            return false;
      }
      //Проверка полей. Возвращает true, если номер пропуска подходит под шаблон s24556.., t4543..., a23455...
      public static bool checkEntryCard(string userInfo)
      {
         userInfo = userInfo.Replace(" ", "");
         if (userInfo.Length != 0)
            userInfo = userInfo.First().ToString().ToLower() + userInfo.Substring(1);
         if (!Regex.IsMatch(userInfo, @"^[a,s,t][0-9]+$") || userInfo.Length == 0)
         {
            return true;
         }
         else
            return false;
      }
   }
}
```

Profile

```
using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media.Imaging;
using MySql.Data.MySqlClient;
using Goosle.Models;
using System.Data;

namespace Goosle.Views
{
   /// <summary>
   /// Логика взаимодействия для Profile.xaml
   /// </summary>
   public partial class Profile : Page
   {
      MySqlCommand cmd;
      MySqlConnection conn;
      UserModel user;

      public Profile(MySqlConnection connection, UserModel user)
      {
         InitializeComponent();

         this.user = user;
         conn = connection;

         Photo.Source = new BitmapImage(new Uri("/Goosle;component/boy.png", UriKind.Relative));

         Questionnaire.Text = "\n" + user.Surname + " " + user.Name + " " + user.Patronymic + "\n" + "Группа:  БПМ-16-2\n" + "Дата
рождения:  " + user.Birthday.Substring(0, user.Birthday.Length - 8) + "\n" + "Номер телефона:  " + user.TelephoneNumber + "\n" +
"Студенческий билет:  " + user.Entry_Card;
         line1.Text = "Паспортные данные:";
         line2.Text = "Серия|Номер:  " + user.Passport;
         line3.Text = "Дата выдачи:  25.06.2018";

         Covid(conn, user);

      }

      public void Covid (MySqlConnection conn, UserModel user)
      {
         cmd = new MySqlCommand("select ShouldCheck, is_sick, DateStart, DateEnd from People P inner join COVID C on P.id_person
= C.id_person where Entry_Card = '" + user.Entry_Card + "'", conn);
         MySqlDataAdapter da = new MySqlDataAdapter(cmd);
         DataTable dt = new DataTable();
         da.Fill(dt);
         if (dt.Rows.Count != 0)
         {
            DataRow row = dt.Rows[0];
            if (row["DateStart"] != DBNull.Value)
            {
               line4.Visibility = Visibility.Visible;
               string DateStart = dt.Rows[0].Field<DateTime>("DateStart").ToString();
               line4.Text = "Дата заболевания: " + DateStart.Substring(0, DateStart.Length - 8);
               string DateEnd = " ";
               if (row["DateEnd"] != DBNull.Value)
               {
```

```
                DateEnd = dt.Rows[0].Field<DateTime>("DateEnd").ToString();
                line5.Text = "Дата выхода: " + DateEnd.Substring(0, DateEnd.Length - 8);
            }
            else
            {
                line5.Text = "На больничном";
            }

        }
        if (Convert.ToInt32(row["ShouldCheck"]) == 1)
        {
            line4.Visibility = Visibility.Collapsed;
            line5.Text = "COVID: необходима проверка";
        }
        else if (Convert.ToInt32(row["is_sick"]) == 0)
        {
            line4.Visibility = Visibility.Collapsed;
            line5.Text = "COVID: выздоровел";
        }

    }

    else
    {
        line4.Visibility = Visibility.Collapsed;
        line5.Text = "COVID: не болел";
    }

}


private void Update(object sender, RoutedEventArgs e)
{
    conn.Open();
    UpdateStatus form = new UpdateStatus(user.id_person, conn);
    form.ShowDialog();
    conn.Close();
    Covid(conn, user);
}
}
}
```

UpdateStatus

```
using System;
using System.Windows;
using System.Windows.Input;
using MySql.Data.MySqlClient;

namespace Goosle
{
    /// <summary>
    /// Логика взаимодействия для UpdateStatus.xaml
    /// </summary>
    public partial class UpdateStatus : Window
    {
        int ID;
        MySqlConnection conn;
        public UpdateStatus(int ID, MySqlConnection conn)
        {
            InitializeComponent();
            this.ID = ID;
            this.conn = conn;
        }

        private void buttonClose_Click(object sender, RoutedEventArgs e)
        {
            Close();
        }

        private void healthy_Button(object sender, RoutedEventArgs e)
        {
            this.Hide();
            int document_flag = 1;
            DownloadWindow form_dow = new DownloadWindow(document_flag, ID, conn);
            form_dow.ShowDialog();
            Close();
        }

        private void sick_Button(object sender, RoutedEventArgs e)
        {
            this.Hide();
            int document_flag = 2;
            DownloadWindow form_dow = new DownloadWindow(document_flag, ID, conn);
            form_dow.ShowDialog();


            //this.Show();
        }

        // Позволяет перетаскивать окно программы по экрану зажатой левой кнопкой мыши
        private void grid_MouseDown(object sender, MouseButtonEventArgs e)
        {
            try
            {
                DragMove();
            }
            catch (InvalidOperationException) { }
        }
    }
}
```

DownloadWindow

```
using System;
using System.Windows;
using System.Windows.Input;
using Microsoft.Win32;
using MySql.Data.MySqlClient;
using System.Data;
using Goosle.Models;
using Goosle.Views;

namespace Goosle
{
    /// <summary>
    /// Логика взаимодействия для DownloadWindow.xaml
    /// </summary>
    public partial class DownloadWindow : Window
    {
        public
        int flag;
        int ID;
        MySqlConnection conn;
        MySqlCommand cmd;
        MySqlCommand cmd1;
        UserModel user;


        public DownloadWindow(int flag, int ID, MySqlConnection conn)
        {
            InitializeComponent();
            this.flag = flag;
            this.ID = ID;
            this.conn = conn;

            line1.Text = "Вы загружаете следующие документы:";
            if (flag == 1)
                line2.Text = "- Справка о здравии";
            else
                line2.Text = "- Справка о болезни";
            line3.Text = "- Дополнительные документы";
        }

        private void buttonClose_Click(object sender, RoutedEventArgs e)
        {
            Close();
        }

        private void choose_Button(object sender, RoutedEventArgs e)
        {
            //this.Hide();
            OpenFileDialog open = new OpenFileDialog();
            open.Filter = "Image Files(*.jpg; *.jpeg; *.gif; *.bmp)|*.jpg; *.jpeg; *.gif; *.bmp";
            if (open.ShowDialog() == true)
            {
                // display image in picture box
                string filename = open.FileName;
            }
            //DownloadWindow form_dow = new DownloadWindow(flag);
            //form_dow.ShowDialog();
            //Close();
```

```
            //this.Show();
        }

        private void send_Button(object sender, RoutedEventArgs e)
        {
            if(flag != 1)
            {
                DateTime end = DateTime.Now;
                DateTime begin = end.Subtract(TimeSpan.FromDays(13));
                string sqlend = end.ToString("yyyy-MM-dd");
                string sqlbegin = begin.ToString("yyyy-MM-dd");
                cmd = new MySqlCommand("INSERT INTO COVID (id_person, is_sick, ShouldCheck, DateStart, DateEnd)" +
    "VALUES(" + ID.ToString() + ", 1, 0, '" + sqlbegin + "', '" + sqlend + "')", conn);
                //conn.Open();

                cmd.ExecuteNonQuery();

                cmd1 = new MySqlCommand("SELECT id_person FROM Attendance A INNER JOIN Students S2 on A.id_student =
S2.id_student WHERE (A.id_student IN ( SELECT id_student FROM People P inner join Students S ON P.id_person = S.id_person
WHERE S.id_group IN (SELECT id_group FROM Students WHERE Students.id_person = " + ID.ToString() + ")) AND (id_date >= '" +
sqlbegin + "') AND (id_date<= '" + sqlend + "')) GROUP BY id_person", conn);
                MySqlDataAdapter da = new MySqlDataAdapter(cmd1);
                DataTable dt = new DataTable();
                da.Fill(dt);
                for (int i = 0; i < dt.Rows.Count; i++)
                {
                    cmd = new MySqlCommand("INSERT INTO COVID (id_person, ShouldCheck) " +
    "VALUES(" + dt.Rows[i].Field<int>("id_person").ToString() + ", 1)", conn);
                    //conn.Open();

                    cmd.ExecuteNonQuery();
                }
            }
            else
            {
                cmd = new MySqlCommand("UPDATE COVID SET is_sick=0, ShouldCheck=0 WHERE id_person =" + ID.ToString(),
conn);

                cmd.ExecuteNonQuery();
            }
            Close();
        }

        // Позволяет перетаскивать окно программы по экрану зажатой левой кнопкой мыши
        private void grid_MouseDown(object sender, MouseButtonEventArgs e)
        {
            try
            {
                DragMove();
            }
            catch (InvalidOperationException) { }
        }
    }
}
```

LessonView

```
using Goosle.Models;
using MySql.Data.MySqlClient;
using System;
using System.Data;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;

namespace Goosle.Views
{
    internal class SelectionObject
    {
        public string Name { get; private set; }
        public int Id { get; private set; }
        public SelectionObject(int id, string name)
        {
            Name = name;
            Id = id;
        }
    }


    /// <summary>
    /// Логика взаимодействия для LessonView.xaml
    /// </summary>
    public partial class LessonView : Page
    {
        private readonly MySqlConnection connection;
        /// <summary>
        /// Селекторы.
        /// </summary>
        private SelectionObject[] groupNames;
        private SelectionObject[] subjects;

        private Student[] students;

        private DataTable AttendsTable;

        public LessonView(MySqlConnection connection)
        {
            this.connection = connection;
            InitializeComponent();
            DateBox.SelectedDate = DateTime.Now;
            FillGroupList();
            FillSubjectsList();
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            if (GroupList.SelectedItem == null || SubjectsBox.SelectedIndex == -1)
            {
                MessageBox.Show($"Требуется выбрать предмет и группу.");
                return;
            }
            var changedStudents = students.Where(s => s.isChanged).ToArray();
            MessageBox.Show($"Изменений: {changedStudents.Length}");
```

```
        if (connection.State != ConnectionState.Open)
        {
            connection.Open();
        }
        for (int i = 0; i < changedStudents.Length; i++)
        {
            var stud = changedStudents[i];
            if (stud.isFromData)
            {
                var deleteCmd = new MySqlCommand(
                    $"UPDATE Attendance SET " +
                    $"Check_={stud.isAttend}, " +
                    $"points={stud.Points} " +
                    $"WHERE id_date='{DateBox.SelectedDate.Value.ToString("yyyy-MM-dd")}' " +
                    $"AND id_subject={subjects[SubjectsBox.SelectedIndex].Id} " +
                    $"AND id_student={stud.Id} " +
                    $"AND Check_={stud._isAttendFromData} " +
                    $"AND points={stud._pointsFromData} " +
                    $"LIMIT 1",
                    connection
                );
                deleteCmd.ExecuteNonQuery();
            } else
            {
                var subjectsCmd = new MySqlCommand(
                    $"INSERT INTO " +
                    $"Attendance (id_date, id_lesson, id_subject, id_student, id_teacher, Check_, points) " +
                    $"VALUES ('{DateBox.SelectedDate.Value.ToString("yyyy-MM-dd")}', 1, {subjects[SubjectsBox.SelectedIndex].Id},
{stud.Id}, 1, {(stud.isAttend ? 1 : 0)}, {stud.Points})",
                    connection
                );
                subjectsCmd.ExecuteNonQuery();
            }
        }
        changedStudents.Select(s => s.isChanged = false);
    }

    private void FillSubjectsList()
    {
        var subjectsCmd = new MySqlCommand($"SELECT * FROM Subjects", connection);
        var adapter = new MySqlDataAdapter(subjectsCmd);
        var subjectsTable = new DataTable();
        adapter.Fill(subjectsTable);
        subjects = new SelectionObject[subjectsTable.Rows.Count];
        for (int i = 0; i < subjectsTable.Rows.Count; i++)
        {
            subjects[i] = new SelectionObject(int.Parse($"{subjectsTable.Rows[i]["id_subject"]}"),
$"{subjectsTable.Rows[i]["Sub_Name"]}");
        }
        SubjectsBox.ItemsSource = subjects;
    }

    private void FillGroupList()
    {
        var groupCmd = new MySqlCommand($"SELECT * FROM `Group`", connection);
        var adapter = new MySqlDataAdapter(groupCmd);
        var groupTable = new DataTable();
        adapter.Fill(groupTable);
        groupNames = new SelectionObject[groupTable.Rows.Count];
```

```
        for (int i = 0; i < groupTable.Rows.Count; i++)
        {
            groupNames[i] = new SelectionObject(int.Parse($"{groupTable.Rows[i]["id_group"]}"),
$"{groupTable.Rows[i]["name_group"]}");
        }
        GroupList.ItemsSource = groupNames;
    }

    private void GroupList_SelectionChanged(object sender, SelectionChangedEventArgs e)
    {
        var groupName = groupNames[GroupList.SelectedIndex];

        var groupCmd = new MySqlCommand(
            $"SELECT * FROM Students " +
            $"JOIN People ON Students.id_person = People.id_person " +
            $"JOIN `Group` ON Students.id_group = `Group`.id_group " +
            $"LEFT JOIN COVID ON COVID.id_person = People.id_person " +
            $"WHERE `Group`.name_group = '{groupName.Name}'",
            connection
        );
        var adapter = new MySqlDataAdapter(groupCmd);
        var lessonsTable = new DataTable();
        adapter.Fill(lessonsTable);
        students = new Student[lessonsTable.Rows.Count];
        for (int i = 0; i < lessonsTable.Rows.Count; i++)
        {
            var name = $"{lessonsTable.Rows[i]["Surname"]} {lessonsTable.Rows[i]["Name"]} {lessonsTable.Rows[i]["Patronymic"]}";
            students[i] = new Student(int.Parse($"{lessonsTable.Rows[i]["id_student"]}"), name);
            if (bool.TryParse($"{lessonsTable.Rows[i]["is_sick"]}", out bool value))
            {
                students[i].Color = (Brush) new BrushConverter().ConvertFrom("#FF3939");
            }
        }
        StudentsListView.ItemsSource = students;
        if (SubjectsBox.SelectedIndex != -1 && DateBox.SelectedDate.HasValue)
        {
            LessonList_SelectionChanged(sender, e);
        }
    }

    private void LessonList_SelectionChanged(object sender, SelectionChangedEventArgs e)
    {
        if (students == null)
        {
            return;
        }
        if (students.Length != 0)
        {
            foreach (var s in students)
            {
                s.Points = 0;
                s.isAttend = false;
            }
        }
        if (GroupList.SelectedIndex == -1)
        {
            return;
        }
        var groupName = groupNames[GroupList.SelectedIndex];
```

```
        var dateDBformat = DateBox.SelectedDate.Value.ToString("yyyy-MM-dd");
        var selectedSub = subjects[SubjectsBox.SelectedIndex];
        var lessonsCmd = new MySqlCommand(
            $"SELECT * FROM Attendance " +
            $"JOIN Students ON Students.id_student=Attendance.id_student " +
            $"WHERE Attendance.id_date='{dateDBformat}' AND Attendance.id_subject={selectedSub.Id}",
            connection
        );
        var adapter = new MySqlDataAdapter(lessonsCmd);
        AttendsTable = new DataTable();
        adapter.Fill(AttendsTable);
        for (int i = 0; i < AttendsTable.Rows.Count; i++)
        {
            var studId = int.Parse($"{AttendsTable.Rows[i]["id_student"]}");
            var isCheck = bool.Parse($"{AttendsTable.Rows[i]["Check_"]}");
            var points = int.Parse($"{AttendsTable.Rows[i]["points"]}");
            var id = int.Parse($"{AttendsTable.Rows[i]["points"]}");
            if (students.Any(s => s.Id == studId))
            {
                var student = students.First(s => s.Id == studId);
                student.SetFromData(isCheck, points);
            }
        }
        StudentsListView.ItemsSource = students;
    }
  }
}
```

ScheduleView

```
using Goosle.Models;
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Media;

namespace Goosle.Views
{
    /// <summary>
    /// Логика взаимодействия для SheduleView.xaml
    /// </summary>
    public partial class SheduleView : Page
    {
        private readonly MySqlConnection connection;
        private readonly static string[] WeekNames = new string[] { "Понедельник", "Вторник", "Среда", "Четверг", "Пятница",
"Суббота" };
        private readonly List<LessonModel> lessonList;
        private BrushConverter colorConverter = new BrushConverter();
        private bool isTeacher = false;
        private DataTable CacheData;

        public SheduleView(MySqlConnection connection, int userId)
        {
            this.connection = connection;
            InitializeComponent();
            var user = new MySqlCommand($"SELECT * FROM People WHERE id_person = {userId} LIMIT 1", connection);
            var userAdapter = new MySqlDataAdapter(user);
            var userTable = new DataTable();
            userAdapter.Fill(userTable);
            MySqlCommand cmd;
            if ($"{userTable.Rows[0]["Role"]}" == "Студент")
            {
                cmd = GetStudentCommand(Convert.ToInt32($"{userTable.Rows[0]["id_person"]}"));
            } else
            {
                isTeacher = true;
                cmd = GetTeacherCommand(Convert.ToInt32($"{userTable.Rows[0]["id_person"]}"));
            }
            var da = new MySqlDataAdapter(cmd);
            CacheData = new DataTable();
            lessonList = new List<LessonModel>();
            da.Fill(CacheData);
            foreach (DataRow row in CacheData.Rows)
            {
                lessonList.Add(new LessonModel()
                {
                    Group = $"{row["name_group"]}",
                    Name = $"{row["Sub_Name"]}",
                    Teacher = $"{row["Name"]} {row["Surname"]} {row["Patronymic"]}",
                    Office = $"{row["Lesson_office"]}",
                    WeekType = $"{row["day_of_week"]}",
```

```
            StartedAt = $"{row["Start"]}"
        });
    }
    FillViewWithSheduleBlocks();
}

public void FillViewWithSheduleBlocks()
{
    for (int indX = 0, dayInd = 0; indX < 2; indX++)
    {
        for (var indY = 0; indY < 3; indY++, dayInd++)
        {
            var weekField = new TextBlock();
            weekField.Margin = new Thickness(5, 5, 0, 0);
            weekField.FontSize = 14;
            weekField.Text = WeekNames[dayInd];
            var grid = CreateSheduleGrid();
            var list = lessonList.Where(l => l.WeekType == WeekNames[dayInd]);
            Grid.SetRow(weekField, indY);
            Grid.SetColumn(weekField, indX);
            grid.ItemsSource = list;
            Grid.SetRow(grid, indY);
            Grid.SetColumn(grid, indX);
            SheduleGrid.Children.Add(weekField);
            SheduleGrid.Children.Add(grid);
        }
    }
}

private GridViewColumn CreateColumn(string bindingName, string columnName, int width)
{
    return new GridViewColumn()
    {
        Header = columnName,
        DisplayMemberBinding = new Binding(bindingName),
        Width = width
    };
}

private ListView CreateSheduleGrid()
{
    var grid = new ListView()
    {
        Padding = new Thickness(-1),
        Margin = new Thickness(5, 25, 5, 5),
        BorderThickness = new Thickness(2),
        Background = Brushes.White,
        BorderBrush = (Brush)colorConverter.ConvertFrom("#3F9EA0"),
    };
    grid.View = AddColumns();
    grid.ItemContainerStyle = Application.Current.FindResource("ListViewItemStyle") as Style;
    return grid;
}

private GridView AddColumns()
{
    var grid = new GridView();
    grid.ColumnHeaderContainerStyle = Application.Current.FindResource("ListViewHeaderStyle") as Style;
    grid.Columns.Add(CreateColumn("Name", "Предмет", 120));
```

```
    if (isTeacher)
    {
        grid.Columns.Add(CreateColumn("Group", "Группа", 100));
    } else
    {
        grid.Columns.Add(CreateColumn("Teacher", "Препод.", 100));
    }
    grid.Columns.Add(CreateColumn("StartedAt", "Время", 100));
    grid.Columns.Add(CreateColumn("Office", "Кабинет", 100));
    return grid;
}

private MySqlCommand GetTeacherCommand(int userId)
{
    return new MySqlCommand(
    "SELECT " +
        "Schedule.day_of_week, " +
        "`Group`.name_group, " +
        "People.Name, " +
        "People.Surname, " +
        "People.Patronymic, " +
        "Offices.Lesson_office, " +
        "Subjects.Sub_Name, " +
        "Lesson_duration.Start, " +
        "Lesson_duration.End " +
    "FROM Schedule " +
        "JOIN `Group` ON Schedule.id_group = `Group`.id_group " +
        "JOIN Teachers ON Teachers.id_teacher = Schedule.id_teacher " +
        "JOIN People ON Teachers.id_teacher = People.id_person " +
        "JOIN Lesson_duration ON Lesson_duration.id_lesson = Schedule.id_lesson " +
        "JOIN Subjects ON Schedule.id_subject = Subjects.id_subject " +
        "JOIN Offices ON Schedule.id_office = Offices.id_office " +
    $"WHERE People.id_person = {userId}",
    connection);
}

private MySqlCommand GetStudentCommand(int userId)
{
    var studentCmd = new MySqlCommand(
        $"SELECT * FROM Students JOIN `Group` ON `Group`.id_group = Students.id_group " +
        $"WHERE Students.id_person = {userId}",
        connection
    );
    var adapter = new MySqlDataAdapter(studentCmd);
    var studentTable = new DataTable();
    adapter.Fill(studentTable);
    var groupId = $"{studentTable.Rows[0]["id_group"]}";
    return new MySqlCommand(
    "SELECT " +
        "Schedule.day_of_week, " +
        "`Group`.name_group, " +
        "People.Name, " +
        "People.Surname, " +
        "People.Patronymic, " +
        "Offices.Lesson_office, " +
        "Subjects.Sub_Name, " +
        "Lesson_duration.Start, " +
        "Lesson_duration.End " +
    "FROM Schedule " +
```

```
            "JOIN `Group` ON Schedule.id_group = `Group`.id_group " +
            "JOIN Teachers ON Teachers.id_teacher = Schedule.id_teacher " +
            "JOIN People ON Teachers.id_teacher = People.id_person " +
            "JOIN Lesson_duration ON Lesson_duration.id_lesson = Schedule.id_lesson " +
            "JOIN Subjects ON Schedule.id_subject = Subjects.id_subject " +
            "JOIN Offices ON Schedule.id_office = Offices.id_office " +
        $"WHERE Schedule.id_group = {groupId}",
        connection);
    }
  }
}
```

StatsView

```
using Goosle.Models;
using MySql.Data.MySqlClient;
using System;
using System.Data;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using System.Windows.Shapes;

namespace Goosle.Views
{
    internal class DepSelection : SelectionObject
    {
        public int id_university { get; set; }
        public int On_Quarantine { get; set; }
        public DepSelection(int id, string name, int on_quarantine, int id_univer) : base(id, name)
        {
            On_Quarantine = on_quarantine;
            id_university = id_univer;
        }
    }
    struct DbDate
    {
        public DbDate(string date, string count)
        {
            Date = date;
            Count = count;
        }
        public string Date;
        public string Count;
    }
    /// <summary>
    /// Логика взаимодействия для StatsView.xaml
    /// </summary>
    public partial class StatsView : Page
    {
        private const int COLUMNS = 12;
        private DepSelection[] Departments;
        private DbDate[] ExistedDates;
        private bool isAdministator;
        private MySqlConnection conn;
        private UserModel user;

        public StatsView(MySqlConnection conn, UserModel user)
        {
            InitializeComponent();
            this.conn = conn;
            this.user = user;
```

```
        Departments = GetSelections();
        DepSelection.ItemsSource = Departments;
        DepSelection.SelectedIndex = 0;
        isAdministator = IsAdministrator(user.id_person);
        if (isAdministator)
        {
            Button_GoQuarantine.Visibility = Visibility.Visible;
            TB_Recomendation.Visibility = Visibility.Visible;
            TB_RecomendationTitle.Visibility = Visibility.Visible;
        }
    }

    private bool IsAdministrator(int id_person)
    {
        var cmd = new MySqlCommand($"SELECT * FROM Administration WHERE id_person={id_person} LIMIT
1", conn);
        var adapter = new MySqlDataAdapter(cmd);
        var table = new DataTable();
        adapter.Fill(table);
        if (table.Rows.Count == 0)
        {
            return false;
        } else
        {
            return true;
        }
    }

    private void Button_GoQuarantine_Click(object sender, RoutedEventArgs e)
    {
        if (conn.State != ConnectionState.Open)
        {
            conn.Open();
        }
        var selectedDep = Departments[DepSelection.SelectedIndex];
        if (selectedDep.Id == -1)
        {
            var cmd = new MySqlCommand(
                $"UPDATE Departments " +
                $"JOIN University ON Departments.id_university=University.id_university " +
                $"SET Departments.on_quarantine=0", conn);
            cmd.ExecuteNonQuery();
            var deps = Departments.Where(d => d.id_university == selectedDep.id_university).ToArray();
            for (var i = 0; i < deps.Count(); i++)
            {
                deps[i].On_Quarantine = 1;
            }
            MessageBox.Show($"Весь университет отправлен на карантин!");
            return;
        }
        if (selectedDep.On_Quarantine == 1)
```

```
        {
            selectedDep.On_Quarantine = 0;
            var cmd = new MySqlCommand(
                $"UPDATE Departments SET " +
                $"on_quarantine=0 " +
                $"WHERE id_department={selectedDep.Id};", conn);
            cmd.ExecuteNonQuery();
        }
        else if (selectedDep.On_Quarantine == 0)
        {
            selectedDep.On_Quarantine = 1;
            var cmd = new MySqlCommand(
                $"UPDATE Departments SET " +
                $"on_quarantine=1 " +
                $"WHERE id_department={selectedDep.Id};", conn);
            cmd.ExecuteNonQuery();
        }
        ChangeCovidRecomendationStatus(selectedDep);
        MessageBox.Show("Сохранено!");
        return;
    }

    private void SetTodayStats(SelectionObject depObject)
    {
        MySqlCommand todayCommand;
        if (depObject.Id == -1)
        {
            todayCommand = new MySqlCommand(
                "SELECT Count(is_sick) as is_sick_count FROM COVID " +
                "JOIN People ON People.id_person=COVID.id_person " +
                $"WHERE COVID.is_sick=1 AND COVID.DateStart=@date",
                conn
            );
            todayCommand.Parameters.AddWithValue("@date", DateTime.Now.ToString("yyyy-MM-dd"));
        } else
        {
            todayCommand = new MySqlCommand(
                "SELECT Count(is_sick) as is_sick_count FROM COVID " +
                "JOIN People ON People.id_person=COVID.id_person " +
                $"WHERE People.id_department=@id_department AND COVID.is_sick=1 AND
COVID.DateStart=@date",
                conn
            );
            todayCommand.Parameters.AddWithValue("@id_department", depObject.Id);
            todayCommand.Parameters.AddWithValue("@date", DateTime.Now.ToString("yyyy-MM-dd"));
        }
        var adapter = new MySqlDataAdapter(todayCommand);
        var table = new DataTable();
        adapter.Fill(table);
        Today.Text = $"Сегодня: {table.Rows[0]["is_sick_count"]}";
    }
```

```
        private DepSelection[] GetSelections()
        {
            var cmd = new MySqlCommand("SELECT * FROM Departments", conn);
            var adapter = new MySqlDataAdapter(cmd);
            var table = new DataTable();
            adapter.Fill(table);
            var namesArr = new DepSelection[table.Rows.Count + 1];

            namesArr[0] = new DepSelection(-1, "Университет", 0, 1);
            for (int ind = 1, tableInd = 0; ind <= table.Rows.Count; ind++, tableInd++)
            {
                int on_quarantine = bool.Parse($"{table.Rows[tableInd]["on_quarantine"]}") ? 1 : 0;
                namesArr[ind] = new DepSelection(
                    int.Parse($"{table.Rows[tableInd]["id_department"]}"),
                    $"{table.Rows[tableInd]["dep_name"]}",
                    on_quarantine,
                    int.Parse($"{table.Rows[tableInd]["id_university"]}")
                );
            }
            return namesArr;
        }

        private void ChangeCovidRecomendationStatus(DepSelection department)
        {
            if (department.Id == -1)
            {
                Button_GoQuarantine.Content = "Отправить на карантин!";
            }
            else
            {
                Button_GoQuarantine.Content = department.On_Quarantine == 1 ? "Отменить карантин!" : "Отправить на
карантин!";
            }
            double peopleCount = GetPeoplesDepartmentCount(department.Id);
            int sickCount = GetSickedPeoplesDepartmentCount(department.Id);
            TB_Recomendation.Text = sickCount > peopleCount * 0.7 ? "Требуется карантин" : "Карантин не
требуется";
            All_count.Text = $"Количество: {peopleCount}";
            Sicked_count.Text = $"Заболевших: {sickCount}";
        }

        private int GetSickedPeoplesDepartmentCount(int dep_id)
        {
            var cmd = new MySqlCommand(
                $"SELECT COUNT(People.id_person) AS people_count FROM People " +
                $"JOIN COVID ON People.id_person=COVID.id_person " +
                $"WHERE People.id_department={dep_id} AND COVID.is_sick=1 " +
                $"GROUP BY People.id_department", conn);
            if (dep_id == -1)
            {
```

```
        cmd = new MySqlCommand(
            $"SELECT COUNT(People.id_person) AS people_count FROM People " +
            $"JOIN COVID ON People.id_person=COVID.id_person " +
            $"JOIN Departments ON Departments.id_department=People.id_department " +
            $"JOIN University ON Departments.id_university=University.id_university " +
            $"WHERE COVID.is_sick=1 AND University.id_university=1 " +
            $"GROUP BY University.id_university", conn);
    }
    var adapter = new MySqlDataAdapter(cmd);
    var table = new DataTable();
    adapter.Fill(table);
    return table.Rows.Count > 0 ? int.Parse($"{table.Rows[0]["people_count"]}") : 0;
}

private int GetPeoplesDepartmentCount(int dep_id)
{
    var cmd = new MySqlCommand(
        $"SELECT COUNT(id_person) AS people_count FROM People " +
        $"WHERE id_department={dep_id} " +
        $"GROUP BY id_department", conn);
    if (dep_id == -1)
    {
        cmd = new MySqlCommand(
            $"SELECT COUNT(People.id_person) AS people_count FROM People " +
            $"JOIN Departments ON Departments.id_department=People.id_department " +
            $"JOIN University ON Departments.id_university=University.id_university " +
            $"GROUP BY University.id_university", conn);
    }
    var adapter = new MySqlDataAdapter(cmd);
    var table = new DataTable();
    adapter.Fill(table);
    return table.Rows.Count > 0 ? int.Parse($"{table.Rows[0]["people_count"]}") : 0;
}

public void Dep_FunSelector(object sender, SelectionChangedEventArgs e)
{
    StatsGrid.Children.Clear();
    var selectedDep = Departments[DepSelection.SelectedIndex];
    SetTodayStats(selectedDep);
    SetGraphHeighAndValue(GetGraphValues(DateTime.Now, selectedDep));
    SetDatesLine(DateTime.Now);
    ChangeCovidRecomendationStatus(selectedDep);
}

private void SetDatesLine(DateTime startTime)
{
    for (int i = 0; i < COLUMNS; i++)
    {
        var text = new TextBlock() { HorizontalAlignment = System.Windows.HorizontalAlignment.Center };
        text.Text = (startTime - new TimeSpan(24*i, 0, 0)).ToString("dd");
        Grid.SetRow(text, 1);
```

```
            Grid.SetColumn(text, COLUMNS-1-i);
            StatsGrid.Children.Add(text);
        }
    }

    private int[] GetGraphValues(DateTime date, SelectionObject depObject)
    {
        int[] resultTableInts = new int[COLUMNS];
        var needUpperDateStr = (date - new TimeSpan(24 * COLUMNS, 0, 0)).ToString("yyyy-MM-dd");
        MySqlCommand cmd;
        if (depObject.Id == -1)
        {
            cmd = new MySqlCommand(
            "SELECT DateStart, SUM(is_sick) as sick_sum FROM COVID " +
            $"WHERE is_sick=1 AND DateStart > '{needUpperDateStr}' " +
            "GROUP BY DateStart " +
            "ORDER BY DateStart", conn);
        } else
        {
            cmd = new MySqlCommand(
                "SELECT COVID.DateStart, SUM(COVID.is_sick) as sick_sum FROM COVID " +
                "JOIN People ON People.id_person = COVID.id_person " +
                $"WHERE COVID.is_sick=1 AND COVID.DateStart > '{needUpperDateStr}' AND
People.id_department={depObject.Id} " +
                "GROUP BY COVID.DateStart",
                conn
            );
        }
        var adapter = new MySqlDataAdapter(cmd);
        var table = new DataTable();
        adapter.Fill(table);
        ExistedDates = new DbDate[table.Rows.Count];
        for (int i = 0; i < ExistedDates.Length; i++)
        {
            ExistedDates[i] = new DbDate(
            $"{DateTime.Parse($"{table.Rows[i]["DateStart"]}").ToString("yyyy-MM-dd")}",
            $"{table.Rows[i]["sick_sum"]}"
            );
        }
        for (int i = 0; i < COLUMNS; i++)
        {
            var neededDate = (date - new TimeSpan(24*i, 0, 0)).ToString("yyyy-MM-dd");
            if (ExistedDates.Any(d => d.Date == neededDate))
            {
                resultTableInts[COLUMNS - 1 - i] = Convert.ToInt32(ExistedDates.First(d => d.Date ==
neededDate).Count);
            } else
            {
                resultTableInts[COLUMNS - 1 - i] = 0;
            }
        }
```

```
            return resultTableInts;
        }

        private void SetGraphHeighAndValue(int[] values)
        {
            int MAX_HEIGHT = 320;
            double maxValue = values.Max();
            double percent = maxValue / MAX_HEIGHT;
            for (int i = 0; i < values.Length; i++)
            {
                var text = new TextBlock() {
                    VerticalAlignment = System.Windows.VerticalAlignment.Bottom,
                    HorizontalAlignment = System.Windows.HorizontalAlignment.Center,
                    Margin = new System.Windows.Thickness(0,0,0,10)
                };
                if (values[i] != 0)
                {
                    text.Text = $"{values[i]}";
                }
                var rect = new Rectangle() {
                    Fill = (Brush)new BrushConverter().ConvertFrom("#045A8D"),
                    VerticalAlignment = System.Windows.VerticalAlignment.Bottom,
                    Height = (values[i] / percent)
                };
                Grid.SetColumn(text, i);
                Grid.SetColumn(rect, i);
                StatsGrid.Children.Add(rect);
                StatsGrid.Children.Add(text);
            }
        }
    }
}
```

8.2 База данных

Создание базы данных

```
use
goosleDB;
            CREATE TABLE goosleDB.University
            (
               id_university int AUTO_INCREMENT PRIMARY KEY,
               on_quarantine bool
            );
            CREATE TABLE goosleDB.Departments
            (
               id_department int AUTO_INCREMENT PRIMARY KEY,
               id_university int,
               dep_name varchar(10),
               on_quarantine bool,
               FOREIGN KEY (id_university) REFERENCES goosleDB.University (id_university) ON
            DELETE CASCADE
            );
            CREATE TABLE goosleDB.People
            (
               id_person      int AUTO_INCREMENT PRIMARY KEY,
               Name           tinytext,
               Surname        tinytext,
               Patronymic     tinytext,
               Birthday       date,
               Age            int,
               Passport       varchar(20),
               TelephoneNumber text,
               id_department   int,
               Photo          blob,
               Entry_Card     varchar(7),
               Pass           tinytext,
               Role           tinytext,
               FOREIGN KEY (id_department) REFERENCES goosleDB.University (id_university) ON
            DELETE SET NULL
            );
            CREATE TABLE goosleDB.COVID
            (
               id_person              int,
               is_sick                bool,
               ShouldCheck            bool,
               Scan_Of_CoronaTestResults blob,
```

```
   DateStart               date,
   DateEnd                 date,
   FOREIGN KEY (id_person) REFERENCES goosleDB.People (id_person) ON DELETE
CASCADE
);
CREATE TABLE goosleDB.Teachers
(
   id_teacher      int AUTO_INCREMENT PRIMARY KEY,
   id_person       int,
   Specializations text,
   FOREIGN KEY (id_person) REFERENCES goosleDB.People (id_person) ON DELETE
CASCADE
);
CREATE TABLE goosleDB.Offices
(
   id_office       int PRIMARY KEY,
   Lesson_office    varchar(9),
   Amount_of_places int
);
CREATE TABLE goosleDB.Administration
(
   id_administration int AUTO_INCREMENT PRIMARY KEY,
   id_person  int,
   Post       varchar(55),
   id_office  int,
   FOREIGN KEY (id_person) REFERENCES goosleDB.People (id_person) ON DELETE
CASCADE,
   FOREIGN KEY (id_office) REFERENCES goosleDB.Offices (id_office) ON DELETE SET
NULL

);
CREATE TABLE goosleDB.Group
(
   id_group   int AUTO_INCREMENT PRIMARY KEY,
   id_teacher int,
   name_group varchar(12),
   FOREIGN KEY (id_teacher) REFERENCES goosleDB.Teachers (id_teacher) ON DELETE
SET NULL
);
CREATE TABLE goosleDB.Students
(
   id_student                  int AUTO_INCREMENT PRIMARY KEY,
   id_person                   int,
```

```
   ScanCertificate_Of_PreviousEducation blob,
   id_group                    int,
   FOREIGN KEY (id_person) REFERENCES goosleDB.People (id_person) ON DELETE
CASCADE,
   FOREIGN KEY (id_group) REFERENCES goosleDB.Group (id_group) ON DELETE SET
NULL
);
CREATE TABLE goosleDB.Subjects
(
   id_subject int AUTO_INCREMENT PRIMARY KEY,
   Sub_Name   text,
   Curriculum text
);
CREATE TABLE goosleDB.Lesson_duration
(
   id_lesson int PRIMARY KEY,
   Start     time,
   End       time
);
CREATE TABLE goosleDB.Attendance
(
   id_date    date,
   id_lesson  int,
   id_subject int,
   id_student int,
   id_teacher int,
   Check_     boolean,
   points     int,
   FOREIGN KEY (id_subject) REFERENCES goosleDB.Subjects (id_subject) ON DELETE
SET NULL,
   FOREIGN KEY (id_student) REFERENCES goosleDB.Students (id_student) ON DELETE
CASCADE,
   FOREIGN KEY (id_lesson) REFERENCES goosleDB.Lesson_duration (id_lesson) ON
DELETE SET NULL,
   FOREIGN KEY (id_teacher) REFERENCES goosleDB.Teachers (id_teacher) ON DELETE
SET NULL

);
CREATE TABLE goosleDB.Schedule
(
   id_schedule int AUTO_INCREMENT PRIMARY KEY,
   id_group    int,
   id_teacher  int,
```

```
    id_subject  int,
    id_lesson   int,
    day_of_week varchar(12),
    WeekType    int,
    Link        text,
    id_office   int,
    FOREIGN KEY (id_group) REFERENCES goosleDB.Group (id_group) ON DELETE NO
ACTION,
    FOREIGN KEY (id_lesson) REFERENCES goosleDB.Lesson_duration (id_lesson) ON
DELETE NO ACTION,
    FOREIGN KEY (id_subject) REFERENCES goosleDB.Subjects (id_subject) ON DELETE
NO ACTION,
    FOREIGN KEY (id_teacher) REFERENCES goosleDB.Teachers (id_teacher) ON DELETE
NO ACTION,
    FOREIGN KEY (id_office) REFERENCES goosleDB.Offices (id_office) ON DELETE NO
ACTION
);
```

Создание тестовых данных

```
use
goosleDB;
        INSERT INTO goosleDB.University (on_quarantine ) VALUES (0);
        INSERT INTO goosleDB.Departments (id_university, on_quarantine, dep_name) VALUES (1,
        0, 'ИТКН');
        INSERT INTO goosleDB.Departments (id_university, on_quarantine, dep_name) VALUES (1,
        0, 'ЭКОТЕХ');
        INSERT INTO goosleDB.Departments (id_university, on_quarantine, dep_name) VALUES (1,
        0, 'ИНМИН');
        INSERT INTO goosleDB.People (Name, Surname, Patronymic, Birthday, Age, Passport,
        TelephoneNumber, id_department, Photo, Entry_Card, Pass, Role)
        VALUES ('Дмитрий', 'Иванов', 'Геннадьевич', '1997-04-10', 24, '1224354643',
        '89705112343', 1, null, 's123456', '4354646', 'Студент');
        INSERT INTO goosleDB.People (Name, Surname, Patronymic, Birthday, Age, Passport,
        TelephoneNumber, id_department, Photo, Entry_Card, Pass, Role)
        VALUES ('Олег', 'Сидиров', 'Евгеньевич', '1998-09-02', 23, '2343234654', '06112345902', 1,
        null, 's132454', '2342444', 'Студент');
        INSERT INTO goosleDB.People (Name, Surname, Patronymic, Birthday, Age, Passport,
        TelephoneNumber, id_department, Photo, Entry_Card, Pass, Role)
        VALUES ('Татьяна', 'Альмухаметова', 'Руслановна', '1998-12-24', 23, '1243686744',
        '23464353635', 1, null, 's243523', '3535422', 'Студент');
        INSERT INTO goosleDB.People (Name, Surname, Patronymic, Birthday, Age, Passport,
        TelephoneNumber, id_department, Photo, Entry_Card, Pass, Role)
        VALUES ('Инсур', 'Кунафин', 'Айдарович', '1999-03-29', 22, '2436467475', '14536464532', 1,
        null, 's453422', '3563411', 'Студент');
        INSERT INTO goosleDB.People (Name, Surname, Patronymic, Birthday, Age, Passport,
        TelephoneNumber, id_department, Photo, Entry_Card, Pass, Role)
        VALUES ('Киркоров', 'Киркоров', 'Бедросович', '2000-10-11', 21, '4335463422',
        '85685433431', 1, null, 't124547', '1245554', 'Преподаватель');
        INSERT INTO goosleDB.Teachers (id_person, Specializations) VALUES (5,'Старший
        преподаватель, к.т.н.');
        INSERT INTO goosleDB.Group (id_teacher, name_group) VALUES (1,'МПИ-20-7-2');
        INSERT INTO goosleDB.Students (id_person, ScanCertificate_Of_PreviousEducation,
        id_group)
        VALUES (1,null,1);
        INSERT INTO goosleDB.Students (id_person, ScanCertificate_Of_PreviousEducation,
        id_group)
        VALUES (2,null,1);
        INSERT INTO goosleDB.Students (id_person, ScanCertificate_Of_PreviousEducation,
        id_group)
        VALUES (3,null,1);
```

INSERT INTO goosleDB.Students (id_person, ScanCertificate_Of_PreviousEducation, id_group)
VALUES (4,null,1);
INSERT INTO goosleDB.People (Name, Surname, Patronymic, Birthday, Age, Passport, TelephoneNumber, id_department, Photo, Entry_Card, Pass, Role) VALUES ('Инна', 'Соколова', 'Арнольдовна', '1980-04-21', 41, '2914593023', '89071112332', 1, null, 't123456', '1224356', 'Преподаватель');
INSERT INTO goosleDB.People (Name, Surname, Patronymic, Birthday, Age, Passport, TelephoneNumber, id_department, Photo, Entry_Card, Pass, Role) VALUES ('Алина', 'Ксенофонтова', 'Андреевна', '1998-06-20', 23, '2909464837', '89034636272', 1, null, 's124433', '3454356', 'Студент');
INSERT INTO goosleDB.People (Name, Surname, Patronymic, Birthday, Age, Passport, TelephoneNumber, id_department, Photo, Entry_Card, Pass, Role) VALUES ('Денис', 'Фролов', 'Тимофеевич', '1998-03-23', 23, '2914664837', '89034636270', 1, null, 's224433', '2354634', 'Студент');
INSERT INTO goosleDB.People (Name, Surname, Patronymic, Birthday, Age, Passport, TelephoneNumber, id_department, Photo, Entry_Card, Pass, Role) VALUES ('Егор', 'Антипов', 'Максимович', '1970-09-22', 51, '2909461111', '89034636271', 2, null, 't120066', '2345432', 'Преподаватель');
INSERT INTO goosleDB.People (Name, Surname, Patronymic, Birthday, Age, Passport, TelephoneNumber, id_department, Photo, Entry_Card, Pass, Role) VALUES ('Светлана', 'Кучерова', 'Николаевна', '1998-06-11', 23, '2909464830', '89034636274', 2, null, 's125554', '3566453', 'Студент');
INSERT INTO goosleDB.People (Name, Surname, Patronymic, Birthday, Age, Passport, TelephoneNumber, id_department, Photo, Entry_Card, Pass, Role) VALUES ('Дарья', 'Кошель', 'Сергеевна', '1998-10-10', 23, '2909464835', '89034636273', 2, null, 's100433', '3787900', 'Студент');
INSERT INTO goosleDB.People (Name, Surname, Patronymic, Birthday, Age, Passport, TelephoneNumber, id_department, Photo, Entry_Card, Pass, Role) VALUES ('Виктория', 'Копылова', 'Юрьевна', '1976-12-14', 45, '2909464836', '89034636275', 3, null, 't123421', '6649000', 'Преподаватель');
INSERT INTO goosleDB.People (Name, Surname, Patronymic, Birthday, Age, Passport, TelephoneNumber, id_department, Photo, Entry_Card, Pass, Role) VALUES ('Эльдар', 'Рязанов', 'Тимурович', '1999-06-20', 22, '2909464831', '89034636276', 3, null, 's198763', '4457321', 'Студент');
INSERT INTO goosleDB.Teachers (id_person, Specializations) VALUES (11, 'Профессор математических наук');
INSERT INTO goosleDB.Teachers (id_person, Specializations) VALUES (17, 'Младший преподаватель');
INSERT INTO goosleDB.Departments (id_department, id_university, on_quarantine, dep_name) VALUES (666, 1, 0, 'ADMIN');
INSERT INTO goosleDB.People (Name, Surname, Patronymic, Birthday, Age, Passport, TelephoneNumber, id_department, Photo, Entry_Card, Pass, Role) VALUES ('Маргарита',

'Суханкина', 'Андреевна', '1977-12-14', 44, '2909464837', '89034636277', 666, null, 'a100001', '3257763', 'Администрация');
INSERT INTO goosleDB.People (Name, Surname, Patronymic, Birthday, Age, Passport, TelephoneNumber, id_department, Photo, Entry_Card, Pass, Role) VALUES ('Сергей', 'Рублев', 'Дмитриевич', '1976-01-14', 45, '2909464838', '89034636278', 666, null, 'a100002', '3564711', 'Администрация');
INSERT INTO goosleDB.Offices (id_office, Lesson_office, Amount_of_places) VALUES (1, 'Б-601', 3);
INSERT INTO goosleDB.Offices (id_office, Lesson_office, Amount_of_places) VALUES (2, 'Б-602', 3);
INSERT INTO goosleDB.Offices (id_office, Lesson_office, Amount_of_places) VALUES (3, 'Б-907', 50);
INSERT INTO goosleDB.Offices (id_office, Lesson_office, Amount_of_places) VALUES (4, 'Б-904а', 25);
INSERT INTO goosleDB.Offices (id_office, Lesson_office, Amount_of_places) VALUES (5, 'К-311', 100);
INSERT INTO goosleDB.Administration (id_person, Post, id_office) VALUES (19, 'Проректор по безопасности', 1);
INSERT INTO goosleDB.Administration (id_person, Post, id_office) VALUES (20, 'Проректор по учебной части', 2);
INSERT INTO goosleDB.Teachers (id_person, Specializations) VALUES (14, 'Старший преподаватель');
INSERT INTO goosleDB.`Group` (id_teacher, name_group) VALUES (4, 'БМН-19-2');
INSERT INTO goosleDB.`Group` (id_teacher, name_group) VALUES (3, 'БЭН-19-1-4');
INSERT INTO goosleDB.Students (id_person, ScanCertificate_Of_PreviousEducation, id_group) VALUES (11, null, 1);
INSERT INTO goosleDB.Students (id_person, ScanCertificate_Of_PreviousEducation, id_group) VALUES (12, null, 1);
INSERT INTO goosleDB.Students (id_person, ScanCertificate_Of_PreviousEducation, id_group) VALUES (13, null, 1);
INSERT INTO goosleDB.Students (id_person, ScanCertificate_Of_PreviousEducation, id_group) VALUES (15, null, 3);
INSERT INTO goosleDB.Students (id_person, ScanCertificate_Of_PreviousEducation, id_group) VALUES (16, null, 3);
INSERT INTO goosleDB.Students (id_person, ScanCertificate_Of_PreviousEducation, id_group) VALUES (18, null, 4);
INSERT INTO goosleDB.Subjects (Sub_Name) VALUES ('Математический анализ 1 часть');
INSERT INTO goosleDB.Subjects (Sub_Name) VALUES ('Математический анализ 2 часть');
INSERT INTO goosleDB.Subjects (Sub_Name) VALUES ('Ряды и ТФКП');
INSERT INTO goosleDB.Subjects (Sub_Name) VALUES ('Дифференциальные уравнения');
INSERT INTO goosleDB.Subjects (Sub_Name) VALUES ('Уравнения в частных

производных');
INSERT INTO goosleDB.Subjects (Sub_Name) VALUES ('Физика 1 часть');
INSERT INTO goosleDB.Subjects (Sub_Name) VALUES ('Физика 2 часть');
INSERT INTO goosleDB.Subjects (Sub_Name) VALUES ('История');
INSERT INTO goosleDB.Subjects (Sub_Name) VALUES ('Философия');
INSERT INTO goosleDB.Lesson_duration (id_lesson, Start, End) VALUES (1, '09:00:00', '10:35:00');
INSERT INTO goosleDB.Lesson_duration (id_lesson, Start, End) VALUES (2, '10:50:00', '12:25:00');
INSERT INTO goosleDB.Lesson_duration (id_lesson, Start, End) VALUES (3, '12:40:00', '14:15:00');
INSERT INTO goosleDB.Lesson_duration (id_lesson, Start, End) VALUES (4, '14:30:00', '16:05:00');
INSERT INTO goosleDB.Lesson_duration (id_lesson, Start, End) VALUES (5, '16:20:00', '17:55:00');
INSERT INTO goosleDB.Lesson_duration (id_lesson, Start, End) VALUES (6, '18:00:00', '19:25:00');
INSERT INTO goosleDB.Lesson_duration (id_lesson, Start, End) VALUES (7, '19:35:00', '20:55:00');
INSERT INTO goosleDB.Schedule (id_group, id_teacher, id_subject, id_lesson, day_of_week, WeekType, Link, id_office) VALUES (1, 1, 1, 1, 'Понедельник', 1, null, 3);
INSERT INTO goosleDB.Schedule (id_group, id_teacher, id_subject, id_lesson, day_of_week, WeekType, Link, id_office) VALUES (1, 1, 1, 2, 'Понедельник', 1, null, 4);
INSERT INTO goosleDB.Schedule (id_group, id_teacher, id_subject, id_lesson, day_of_week, WeekType, Link, id_office) VALUES (1, 4, 9, 3, 'Понедельник', 1, null, 3);
INSERT INTO goosleDB.Schedule (id_group, id_teacher, id_subject, id_lesson, day_of_week, WeekType, Link, id_office) VALUES (1, 4, 8, 2, 'Вторник', 1, null, 4);
INSERT INTO goosleDB.Schedule (id_group, id_teacher, id_subject, id_lesson, day_of_week, WeekType, Link, id_office) VALUES (1, 3, 6, 3, 'Вторник', 1, null, 3);
INSERT INTO goosleDB.Schedule (id_group, id_teacher, id_subject, id_lesson, day_of_week, WeekType, Link, id_office) VALUES (1, 3, 6, 4, 'Вторник', 1, null, 4);
INSERT INTO goosleDB.Schedule (id_group, id_teacher, id_subject, id_lesson, day_of_week, WeekType, Link, id_office) VALUES (1, 4, 9, 1, 'Четверг', 1, null, 3);
INSERT INTO goosleDB.Schedule (id_group, id_teacher, id_subject, id_lesson, day_of_week, WeekType, Link, id_office) VALUES (1, 4, 8, 3, 'Четверг', 1, null, 4);
INSERT INTO goosleDB.Schedule (id_group, id_teacher, id_subject, id_lesson, day_of_week, WeekType, Link, id_office) VALUES (1, 1, 1, 4, 'Четверг', 1, null, 3);
INSERT INTO goosleDB.Schedule (id_group, id_teacher, id_subject, id_lesson, day_of_week, WeekType, Link, id_office) VALUES (1, 1, 1, 1, 'Понедельник', 2, null, 3);
INSERT INTO goosleDB.Schedule (id_group, id_teacher, id_subject, id_lesson, day_of_week, WeekType, Link, id_office) VALUES (1, 1, 1, 2, 'Понедельник', 2, null, 4);
INSERT INTO goosleDB.Schedule (id_group, id_teacher, id_subject, id_lesson, day_of_week, WeekType, Link, id_office) VALUES (1, 4, 9, 3, 'Понедельник', 2, null, 3);

INSERT INTO goosleDB.Schedule (id_group, id_teacher, id_subject, id_lesson, day_of_week, WeekType, Link, id_office) VALUES (1, 4, 8, 2, 'Вторник', 2, null, 4);
INSERT INTO goosleDB.Schedule (id_group, id_teacher, id_subject, id_lesson, day_of_week, WeekType, Link, id_office) VALUES (1, 3, 6, 3, 'Вторник', 2, null, 3);
INSERT INTO goosleDB.Schedule (id_group, id_teacher, id_subject, id_lesson, day_of_week, WeekType, Link, id_office) VALUES (1, 4, 9, 1, 'Четверг', 2, null, 3);
INSERT INTO goosleDB.Schedule (id_group, id_teacher, id_subject, id_lesson, day_of_week, WeekType, Link, id_office) VALUES (1, 4, 8, 3, 'Четверг', 2, null, 4);
INSERT INTO goosleDB.COVID (id_person, is_sick, ShouldCheck, Scan_Of_CoronaTestResults, DateStart, DateEnd) VALUES (3, 1, 1, null, '2021-04-01', null);
INSERT INTO goosleDB.Attendance (id_date, id_lesson, id_subject, id_student, id_teacher, Check_, points) VALUES ('2021-04-05', 1, 9, 1, 4, 1, 5);
INSERT INTO goosleDB.Attendance (id_date, id_lesson, id_subject, id_student, id_teacher, Check_, points) VALUES ('2021-04-05', 1, 9, 2, 4, 0, 0);
INSERT INTO goosleDB.Attendance (id_date, id_lesson, id_subject, id_student, id_teacher, Check_, points) VALUES ('2021-04-05', 1, 9, 3, 4, 0, 0);
INSERT INTO goosleDB.Attendance (id_date, id_lesson, id_subject, id_student, id_teacher, Check_, points) VALUES ('2021-04-05', 1, 9, 4, 4, 1, 0);
INSERT INTO goosleDB.Attendance (id_date, id_lesson, id_subject, id_student, id_teacher, Check_, points) VALUES ('2021-04-05', 1, 9, 5, 4, 1, 0);
INSERT INTO goosleDB.Attendance (id_date, id_lesson, id_subject, id_student, id_teacher, Check_, points) VALUES ('2021-04-05', 1, 9, 6, 4, 1, 4);
INSERT INTO goosleDB.Attendance (id_date, id_lesson, id_subject, id_student, id_teacher, Check_, points) VALUES ('2021-04-05', 1, 9, 7, 4, 1, 1);
INSERT INTO goosleDB.Attendance (id_date, id_lesson, id_subject, id_student, id_teacher, Check_, points) VALUES ('2021-04-05', 3, 8, 1, 4, 1, 5);
INSERT INTO goosleDB.Attendance (id_date, id_lesson, id_subject, id_student, id_teacher, Check_, points) VALUES ('2021-04-05', 3, 8, 2, 4, 0, 0);
INSERT INTO goosleDB.Attendance (id_date, id_lesson, id_subject, id_student, id_teacher, Check_, points) VALUES ('2021-04-05', 3, 8, 3, 4, 0, 0);
INSERT INTO goosleDB.Attendance (id_date, id_lesson, id_subject, id_student, id_teacher, Check_, points) VALUES ('2021-04-05', 3, 8, 4, 4, 1, 5);
INSERT INTO goosleDB.Attendance (id_date, id_lesson, id_subject, id_student, id_teacher, Check_, points) VALUES ('2021-04-05', 3, 8, 5, 4, 1, 0);
INSERT INTO goosleDB.Attendance (id_date, id_lesson, id_subject, id_student, id_teacher, Check_, points) VALUES ('2021-04-05', 3, 8, 6, 4, 1, 4);
INSERT INTO goosleDB.Attendance (id_date, id_lesson, id_subject, id_student, id_teacher, Check_, points) VALUES ('2021-04-05', 3, 8, 7, 4, 1, 1);
INSERT INTO goosleDB.Attendance (id_date, id_lesson, id_subject, id_student, id_teacher, Check_, points) VALUES ('2021-04-05', 4, 1, 1, 1, 1, 5);
INSERT INTO goosleDB.Attendance (id_date, id_lesson, id_subject, id_student, id_teacher, Check_, points) VALUES ('2021-04-05', 4, 1, 2, 1, 0, 0);
INSERT INTO goosleDB.Attendance (id_date, id_lesson, id_subject, id_student, id_teacher,

```
Check_, points) VALUES ('2021-04-05', 4, 1, 3, 1, 0, 0);
INSERT INTO goosleDB.Attendance (id_date, id_lesson, id_subject, id_student, id_teacher,
Check_, points) VALUES ('2021-04-05', 4, 1, 4, 1, 1, 3);
INSERT INTO goosleDB.Attendance (id_date, id_lesson, id_subject, id_student, id_teacher,
Check_, points) VALUES ('2021-04-05', 4, 1, 5, 1, 1, 0);
INSERT INTO goosleDB.Attendance (id_date, id_lesson, id_subject, id_student, id_teacher,
Check_, points) VALUES ('2021-04-05', 4, 1, 6, 1, 1, 4);
INSERT INTO goosleDB.Attendance (id_date, id_lesson, id_subject, id_student, id_teacher,
Check_, points) VALUES ('2021-04-05', 4, 1, 7, 1, 1, 1);
```

9 Список литературы

I. Законодательные и нормативные акты

1. ГОСТ 19.201-78. «Единая система программной документации. Техническое задание. Требования к содержанию и оформлению»;

2. ГОСТ 34.602-89. Техническое задание на создание автоматизированной системы.

II. Монографии, учебники, учебные пособия

3. Бабич А.В UML: Первое знакомство/ - М.: Национальный Открытый Университет "ИНТУИТ", 2016.-209 с

4. Храмцов П.Б., Брик С.А., Русак А.М., Сурин А.И Основы web-технологий /. - М.: Национальный Открытый Университет "ИНТУИТ", 2016.- 166

1) Медведев А.В. Программа производственной (в том числе и преддипломной) практики для студентов, обучающихся по направлению 09.03.03 «Прикладная информатика» (программа подготовки бакалавра). – М.: Финансовый университет, кафедра «Информационные технологии», 2015. – 31 с.;

III. Статьи из периодической печати

5. UML-Моделирование предметной области при проектировании информационной системы салона красоты / Журба А.К., Продан Е.А., Сушкова М.С. //Инновационная наука. -2015. -№ 7-8. -С. 52-55

IV. Иностранная литература

6. Ларман К. Применение UML и шаблонов проектирования/пер. с англ. Шелестова А. Ю.. - М.: Вильямс, 2017. - 727 с

V. Интернет-ресурсы

7. Информационный портал «Научная электронная библиотека», 2014-2018, Режим доступа - http://www.book.ru.(Дата обращения: 19.05.2019).

8. Информационный портал «Научная электронная библиотека», 2014-2018 Режим доступа -http://www.elibrary.ru. (Дата обращения: 19.05.2019).
9. Информационный портал «Научная электронная библиотека», 2014-2018, Режим доступа - http://www.znanium.ru. (Дата обращения: 19.05.2019).