МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики
Кафедра информационных технологий

Допустить к защите
Заведующий кафедрой
канд. физ.-мат. наук, доц.
_____ В.В. Подколзин
(подпись)
_____2023г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(БАКАЛАВРСКАЯ РАБОТА)

ВЕБ-РЕСУРС УЧЕТА ПОСЕЩАЕМОСТИ ЗАНЯТИЙ В ВУЗЕ

Работу выполнил _____ К.С. Минин
(подпись)

Направление подготовки 02.03.02 Фундаментальная информатика и информационные технологии

Направленность (профиль) Математическое и программное обеспечение компьютерных технологий

Научный руководитель
канд. пед. наук, доц. _____Н.Ю. Добровольская
(подпись)

Нормоконтролер
канд. пед. наук, доц. _____А.В. Харченко
(подпись)

Краснодар
2023

РЕФЕРАТ

В выпускной квалификационной работе 50 страниц, 34 рисунка, 15 источников.

Ключевые слова: генерация QR кода, веб-ресурс, учет посещаемости занятий, цифровая среда вуза.

Цель работы – разработка веб-ресурса учета посещаемости занятий в вузе на основе применения QR кодов.

При подготовке выпускной квалификационной работы изучена структура QR кодов, алгоритмы кодирования информации, а также создан скрипт на языке Python, позволяющий генерировать QR код заданного текста в необходимом формате.

В работе выполнен обзор существующих систем учета посещаемости и успеваемости студентов с различными способами автоматизации процессов сбора и хранения необходимой информации, выделены их достоинства и недостатки.

В практической части выпускной квалификационной работы реализован веб-ресурс учета посещаемости занятий в вузе с использованием фрэймворка VueJS и библиотеки BootstrapVue для упрощения стилизации визуальных компонентов. Также создан сервер данных с использованием фреймворка Django и библиотек SimpleJWT и RestFramework для развертывания сервера авторизации и реализации API.

Предложенная технология учета посещаемости с использованием QR кодов может быть применима для фиксирования наличия различных объектов, например, участников конференции или заезжающих на парковку автомобилей.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Автоматизированный контроль знаний и умений обучающихся является неотъемлемой и обязательной частью образовательного процесса, современной цифровой среды вуза. При его отсутствии преподавателю сложно достоверно и за короткое время оценить текущий уровень подготовки студента. В то же время, качественно организованный процесс обучения предполагает своевременный контроль посещаемости занятий, отслеживание статистики пропусков и информирование об этом представителей деканата и кураторов групп. Естественно, такой процесс в рамках цифровой среды вуза целесообразно автоматизировать.

В качестве механизма цифрового учета посещаемости занятий целесообразно использовать технологию QR кодов. Сам код может быть достаточно информативным и способен хранить не только текстовую информацию, но и ссылки на различные сайты или документы, предзаполненные SMS-сообщения и электронные письма и т.д. При решении задачи цифровизации учета посещаемости сгенерированный код может содержать дату, номер пары, сведения о студенте, специальный уникальный идентификатор. Актуальность использования технологии QR кодов заключается в их повсеместном использовании для решения бытовых и производственных задач.

В первой главе рассмотрены теоретические аспекты генерации и декодирования QR кода, используемые методы кодирования информации и структура символов.

Вторая глава содержит сравнительный анализ существующих систем автоматизации контроля посещаемости мероприятий и выделение требований к разрабатываемому веб-ресурсу. При их изучении выявлена необходимость в специальном ресурсе, автоматизирующем процессы сбора и хранения информации об участниках различных мероприятий.

В третьей главе выпускной квалификационной работы реализован веб-ресурс учета посещаемости занятий в вузе. Модули генерации QR кода, его обработки и внесения необходимой информации о посещении студентом определенного учебного занятия в базу данных, авторизации пользователей, а также составления сводных отчетов для преподавателей и студентов представлены в разработанном веб-ресурсе.

1 Задача генерации и декодирования QR кода

QR код (Quick Response – быстрое реагирование) — тип двухмерных штриховых кодов, представляет собой черно-белый узор, состоящий из квадратных модулей фиксированного размера (рисунок 1) [1]. Данный тип кодов был разработан для использования в автомобильной промышленности, но вскоре благодаря более высокой информационной емкости по сравнению с UPC-кодами (Universal Product code), а также возможности быстрого считывания получил широкое распространение во многих других областях жизни людей по всему миру. В настоящее время QR коды используются повсеместно: с их помощью зашифрованы реквизиты управляющих компаний в счетах-квитанциях на оплату жилищно-коммунальных услуг для быстрой и удобной оплаты через банковское приложение; они используются в качестве маркировки молочной продукции, благодаря чему потребитель может отследить полный путь продукта от производителя до прилавка через специальную систему и убедиться в его свежести; они применяются для простого и быстрого доступа к официальному сертификату о вакцинации а также для решения многих других задач идентификации и учета.

Рисунок 1 – Пример изображения символа QR кода

В общем случае задача создания QR кода включает в себя следующие этапы: кодирование исходных данных; добавление необходимой технической

информации и заполнение неиспользуемого объема бит для данной версии символа; разделение закодированной информации на блоки данных; генерация байтов исправления ошибок; объединение информационных и корректирующих блоков; размещение полученных данных на QR коде.

1.1 Структура символов QR кодов

QR код является матричным символом, который состоит из квадратных модулей темного и светлого цветов, составляющих квадратную матрицу [2]. Все символы QR кодов включают в себя область кодирования и различные функциональные шаблоны, такие как шаблоны поиска, синхронизации, направляющие шаблоны, разделители. Расположение блоков, составляющих символ QR кода, схематично отражено на рисунке 2.

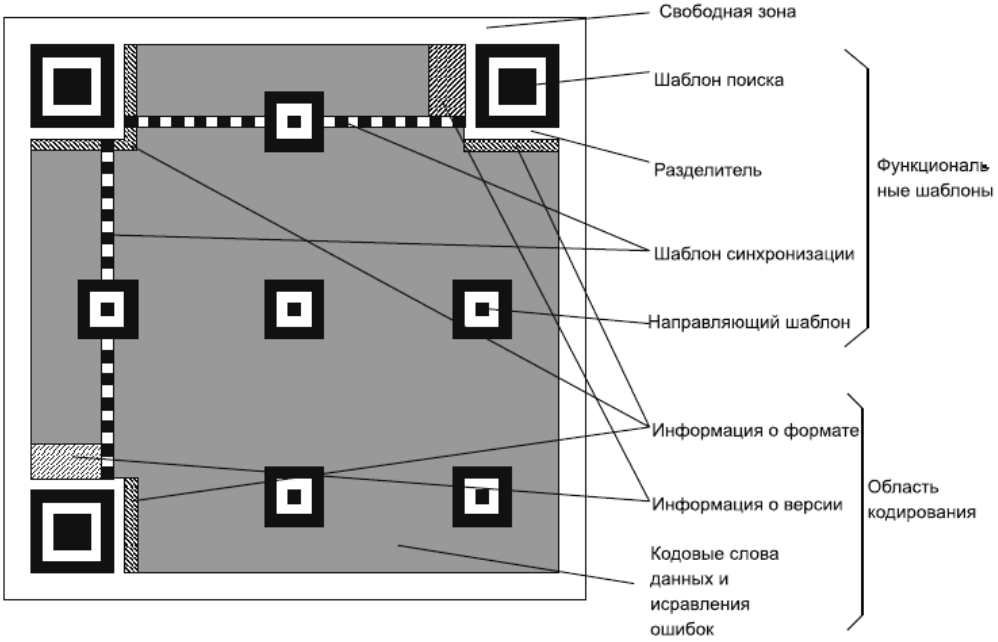

Рисунок 2 – Структура символа QR кода (на примере версии 7)

Функциональные шаблоны необходимы для упрощения процесса поиска точных координат границ символа на графическом изображении, его размера и угла наклона, а также для определения местонахождения и точных координат квадратных модулей темного цвета в символе, данных о способе

кодирования информации. Область кодирования содержит и отражает информацию о версии и формате символа, последовательности бит закодированных данных и блоков исправления ошибок.

Область с информацией о формате содержит индикатор, который указывает режим кодирования следующей за ним последовательности бит и данные о способе преобразования кодируемых символов в двоичный формат. Информация о формате включает в себя данные об уровне исправления ошибок, используемом в данном символе QR кода. Существуют следующие уровни: L, M, Q и H, которые помогают исправить до 7, 15, 25 и 30 процентов ошибок от общего числа кодовых бит символа. Для создания байтов коррекции, позволяющих исправлять ошибки при чтении потока данных из QR кода, используется алгоритм Рида-Соломона.

QR коды, представляющие большой объем информации, также должны содержать в себе область с информацией о версии символа – его размере, который определяется в соответствии с специальной таблицей. Символы QR кодов имеют размеры от 21x21 модулей (версия 1) до 177x177 модулей (версия 40), которые варьируются в зависимости от количества кодируемых символов.

Направляющие шаблоны необходимо располагать только в символах QR кодов, имеющих версии 2 и выше. Количество направляющих шаблонов и места их расположения зависят от версии символа QR кода и заранее определены в ГОСТ Р ИСО/МЭК 18004-2015, регламентирующем спецификацию символики QR кодов.

В символе любого QR кода должны содержаться три шаблона поиска, расположенных во всех углах символа, кроме правого нижнего. Такие шаблоны имеют идентичную структуру вложенных квадратов следующего вида: темный квадрат, размерами 7x7 модулей; светлый, размерами 5x5 модулей; темный, размерами 3x3 модуля (рисунок 3). Идентификация на символе QR кода именно этих шаблонов позволяет определить его местонахождение и угловую ориентацию, вследствие чего накладывается следующее ограничение на представление кодируемых данных: необходимо,

чтобы вероятность нахождения такого шаблона среди расположенных на символе QR кода модулей была очень мала. Это позволит быстро определить наличие символа QR кода в обозреваемом поле изображения.

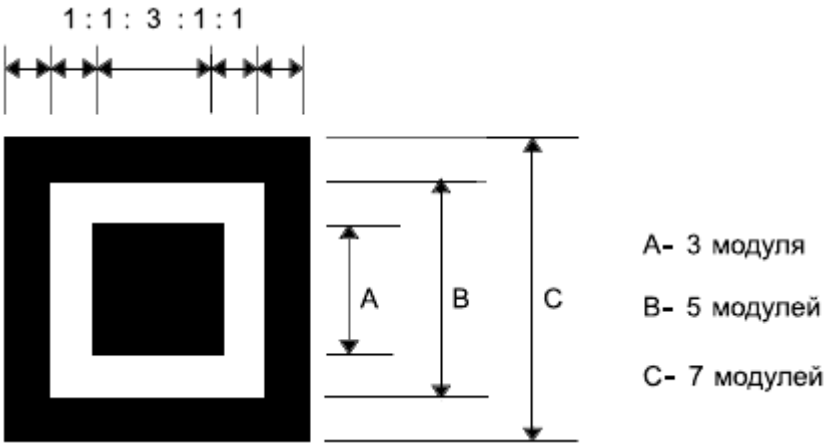

Рисунок 3 – Структура шаблона поиска

Шаблоны синхронизации являются чередующимися последовательностями темных и светлых модулей, которые используются для уточнения координат модулей, представляющих данные, в символе QR кода. Шаблоны разделители, в свою очередь, состоящие из светлых модулей и имеющие ширину один модуль, отделяют шаблоны поиска от остальных модулей и шаблонов символа. Шаблоны синхронизации и разделители наряду с шаблонами поиска являются обязательными, неизменными и присутствуют в символе QR кода любой версии.

1.2 Кодирование данных

Выбранный режим кодирования, который указан в области с информацией о формате, задает правила преобразования символов данных в двоичный поток [3]. При этом результирующая последовательность нулей и единиц должна быть преобразована в некоторое количество кодовых слов размером 8 бит. Также есть возможность замены режима в разных сегментах – для этого в начало каждого нового блока данных необходимо вставить

соответствующие им индикаторы режима кодирования, а в конце блока добавить особый ограничитель. В том случае, если длина результирующей последовательности меньше, чем того требует выбранная версия символа QR кода, после крайнего бита последовательности необходимо добавить соответствующее число знаков-заполнителей.

Стандарт QR кодов поддерживает следующие режимы кодирования данных.

Числовой режим используется для кодирования данных, состоящих из десятичных цифр (0-9). Исходная последовательность должна быть разбита на группы по 3 цифры, затем каждая группа (представляющая трехзначное число) преобразуется в двоичное число (10 бит). В том случае, если количество символов входной последовательности не кратно трем, рассматривается остаток от деления количества символов на три. Если он равен двум, то последние две цифры исходной последовательности данных кодируются семью битами. Если остаток равен одному, то крайний символ кодируется четырьмя битами.

Для кодирования входных данных, состоящих из набора 10 десятичных цифр, 26 латинских букв (A-Z) и 9 специальных символов (SP, $, %, *, +, -, ., /, :) целесообразно использовать алфавитно-цифровой режим кодирования. В первую очередь каждому символу исходных данных необходимо сопоставить значение от 0 до 44 в соответствии с таблицей 1, представленной далее. Затем исходная последовательность данных разбивается на группы по 2 символа и каждая группа кодируется 11 битами следующим образом: получают сумму значения из таблицы 1 для первого знака, умноженного на 45, и значения из таблицы 1 для второго знака. После чего этого производят преобразование данной суммы в 11-битовое двоичное число. В случае если количество символов входных данных нечетно, соответствующее значение последнего знака представляется двоичным числом размером 6 бит.

Таблица 1 – Соответствие символов для алфавитно-цифрового режима

Знак	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
Значение	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Знак	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
Значение	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
Знак	U	V	W	X	Y	Z	Пробел	$	%	*	+	-	.	/	:
Значение	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44

Например, входная строка имеет вид: «HIKUBSU». Она разделяется на группы: «HI», «KU», «BS» и «U». Из таблицы каждому символу в каждой группе находится соответствие: (17, 18), (20, 30), (11, 28) и (30). Далее находится одиннадцатибитное (или шестибитное) значение для каждой группы.

Первая группа: $17 * 45 + 18 = 783_{10} = 1100001111_2$.

Вторая группа: $20 * 45 + 30 = 930_{10} = 01110100010_2$.

Третья группа: $11 * 45 + 28 = 523_{10} = 01000001011_2$.

Четвертая группа: $30_{10} = 011110_2$.

Полученные двоичные последовательности объединяются в один поток данных: «11000011110111010001001000001011011110».

Режим кандзи предназначен для эффективного кодирования китайских и японских иероглифов в соответствии с системой Shift JIS. В данном режиме каждый знак (2 байта) кодируется в кодовое слово длиной 13 бит.

Универсальным способом кодирования является байтовый режим, однако при его использовании наблюдается довольно низкая плотность информации. Для кодирования в данном режиме исходная последовательность данных представляется в любой двоичной кодировке (для корректного отображения символов Кириллицы и универсальности кода рекомендована UTF-8) и полученный набор байтов переводится в единый двоичный поток без изменений. Все коды символов, длина двоичного

представления которых меньше 8, необходимо дополнить незначащими нулями до 8 бит.

Для решения поставленной задачи – разработки системы автоматизированного учета посещаемости занятий в вузе – будем использовать QR код как основу механизма учета. Первым этапом работы является разработка структуры и собственно генерация QR кода.

Реализованный скрипт генерации QR кода, который в дальнейшем используется в веб-ресурсе учета посещаемости занятий в вузе, включает в себя метод байтового кодирования исходных текстовых данных (рисунок 4). Входная строка преобразуется в поток объектов типа Bytes с помощью встроенного метода .encode(). Полученный поток байтов преобразуется в бинарное представление в методе binary_encoding() с помощью встроенного метода bin(), который преобразует объект типа Bytes в двоичное число. В методе binary_encoding() реализовано дополнение двоичного числа, длина которого меньше 8 бит, незначащими нулями.

```
encoding_string = "Я <3 КИТ"

# получаем байтовое представление UTF-8
bytes_encoded = encoding_string.encode(encoding='utf8', errors='replace')

# получаем бинарное представление UTF-8
binary_encoded = binary_encoding(bytes_encoded)
```

Рисунок 4 – Байтовое кодирование в реализованном приложении

Стандарт QR кодов позволяет использовать смешанный режим кодирования. В таком случае информация может быть представлена в виде нескольких последовательностей двоичных данных, каждая из которых может быть представлена в любом из режимов кодирования.

1.3 Этапы генерации QR кода

Этапы создания QR кода в общем виде представлены на рисунке 5.

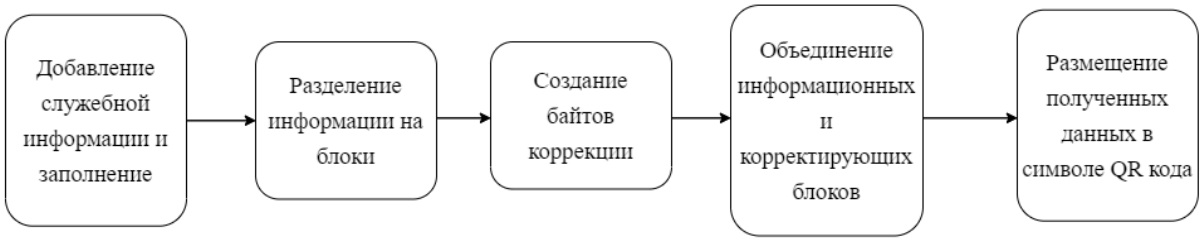

Рисунок 5 – Этапы генерации QR кода

Этап 1. Добавление служебной информации и заполнение.

Служебная информация содержит в себе индикаторы режима кодирования входных данных и числа знаков (число бит в этом индикаторе).

Для определения количества бит в индикаторе числа знаков на данном этапе необходимо определить версию QR кода и величины, определяющей какое количество ошибок представится возможным найти и исправить. Это связано с тем, что версия QR кода напрямую зависит от количества кодируемых данных. Для обеспечения минимального количества битов-заполнителей необходимо выбирать версию таким образом, чтобы ее емкость данных была максимально близка к объему закодированных данных.

После определения последовательностей бит, обозначающих соответствующий режим кодирования и число данных, необходимо добавить их в начало закодированного потока данных.

В случае если длина полученного потока бит не кратна 8, необходимо добавить незначащие нули в конец потока. Например, поток имеет вид: <последовательность бит (длина кратна 8)> 100110101001. Так как остаток от деления длины потока на 8 равен 4, необходимо добавить 4 незначащих нуля в конец этого потока. Результат будет следующим: <последовательность бит (длина кратна 8)> 1001101010010000.

Если полученная двоичная последовательность не заполняет полностью емкость символа, определяемую в соответствии с выбранными версией и уровнем исправления ошибок, то она должна быть расширена путем поочередной конкатенации к нему слов-заполнителей «11101100» и «11101100».

В реализованном скрипте на языке Python, позволяющем сгенерировать QR код, дополнение данных происходит при помощи метода fill_data(), код которого изображен на рисунке 6.

```
# кодовые слова-заполнители - 11101100 и 00010001 (поочередно)
def fill_data(data_str, c_bytes_ver):
    l_data_str = len(data_str)
    # если длина не делится на 8, то дополняем нулями в конец
    if l_data_str % 8 != 0:
        data_str += '0' * (len(data_str) % 8)
    t_count_bytes = len(data_str) / 8   # сколько байт у нас сейчас есть?
    fl = t_count_bytes < c_bytes_ver    # флаг проверки, меньше ли у нас байтов, чем нужно для версии
    fill_code_word = "11101100"
    while fl:
        if fill_code_word == "11101100":
            data_str += "11101100"
            fill_code_word = "00010001"
            t_count_bytes = len(data_str) / 8  # сколько байт у нас сейчас есть?
            fl = t_count_bytes < c_bytes_ver
        else:
            data_str += "00010001"
            fill_code_word = "11101100"
            t_count_bytes = len(data_str) / 8  # сколько байт у нас сейчас есть?
            fl = t_count_bytes < c_bytes_ver
    return data_str
```

Рисунок 6 – Код метода fill_data() в реализованном приложении

Этап 2. Разделение информации на блоки.

Для реализации алгоритма Рида-Соломона, позволяющего декодировать символ QR кода с исправлением ошибок, полученная на предыдущем этапе последовательность двоичных символов должна быть разделена на блоки. Разделение происходит в соответствии с специальной таблицей, определенной в ГОСТ Р ИСО/МЭК 18004-2015, часть которой представлена в таблице 2. В случае, когда должен быть получен только один блок, необходимость в разделении отсутствует и последовательность принимается в исходном виде.

Таблица 2 – Количество блоков для некоторых версий QR кодов

Версия / Уровень коррекции	1	2	3	4	5	6	7	8	9	10
L	1	1	1	1	1	2	2	2	2	4
M	1	1	1	2	2	4	4	4	5	5
Q	1	1	2	2	4	4	6	6	8	8
H	1	1	2	4	4	4	5	6	8	8

Для определения количества данных в каждом блоке необходимо разделить общее число байт данных на число блоков, определенное для конкретной версии символа QR кода и уровня исправления ошибок. Если при делении будет получен ненулевой остаток, то он покажет, сколько блоков данных, находящихся в конце, необходимо дополнить. После этого блоки должны быть заполнены полученными ранее байтами данных. Когда очередной блок заполнен, начинается заполнение следующего блока.

В разработанном скрипте генерации QR кода некоторой строки данных заранее определен список, содержащий информацию о том, на какое количество блоков необходимо разделить данные для каждой из версий QR кодов. Подсчет количества байт данных для каждого блока происходит с помощью простых арифметических операций (рисунок 7).

```
# кол-во блоков, на которые нужно поделить данные (для уровня коррекции H и версий 1, 2, 3, ...)
blocks_amount_corr_h = [1, 1, 2, 4, 4, 4, 5, 6, 8, 8,
                        11, 11, 16, 16, 18, 16, 19, 21, 25, 25,
                        25, 34, 30, 32, 35, 37, 40, 42, 45, 48,
                        51, 54, 57, 60, 63, 66, 70, 74, 77, 81]

# кол-во блоков для нашего случая:
amount_blocks = blocks_amount_corr_h[version - 1]

# сколько байт инфы в каждом блоке:
amount_data_in_block = int(byte_amount_data / amount_blocks)

# сколько байт осталось?
amount_data_in_block_remains = byte_amount_data % amount_blocks

# создаем массив блоков информации (сколько байт данных в каждом блоке)
blocks = get_blocks(amount_data_in_block, amount_blocks, amount_data_in_block_remains)
```

Рисунок 7 – Часть реализованного приложения, разделяющая поток данных на блоки

Этап 3. Создание байтов коррекции.

Вычисление кодовых слов, добавляемых в двоичный поток для последующего обнаружения и исправления ошибок (байтов коррекции) производится для каждого отдельного блока, полученного на предыдущем шаге. Затем они добавляются к кодовым словам данных по определенному правилу. Для генерации байтов коррекции в символах QR кодов используется алгоритм кодирования Рида-Соломона – частный случай кодов БЧХ [4].

Создание кодовых слов, позволяющих производить обнаружение и исправление ошибок, основывается на использовании поля Галуа 2^8, где простому минимальному полиному поля $x^8+x^4+x^3+x^2+1$ соответствует последовательность «100011101», а также соответствующих этому полю арифметических операций: побитового счета по модулю «2» и побайтового счета по модулю «100011101» [5].

Все кодовые слова, представляющие исходные данные, являют собой числовые коэффициенты при членах полинома. При этом первое слово является коэффициентом при члене с самой старшей степенью, а последнее слово данных – коэффициентом при члене с наиболее младшей степенью.

Для создания последовательности кодовых слов, позволяющих обнаруживать и исправлять ошибки в символах QR кодов, стандартом определены 36 различных порождающих многочленов, часть из которых приведена в таблице 3.

Таблица 3 – Порождающие многочлены для создания кодовых слов обнаружения и исправления ошибок по Риду-Соломону

Кол-во кодовых слов исправления ошибок	Порождающие многочлены
2	$x^2 + \alpha^{25}x + \alpha$
5	$x^5 + \alpha^{113}x^4 + \alpha^{164}x^3 + \alpha^{166}x^2 + \alpha^{119}x + \alpha^{10}$
6	$x^6 + \alpha^{116}x^5 + x^4 + \alpha^{134}x^3 + \alpha^5x^2 + \alpha^{178}x + \alpha^{15}$
7	$x^7 + \alpha^{87}x^6 + \alpha^{229}x^5 + \alpha^{146}x^4 + \alpha^{149}x^3 + \alpha^{238}x^2 + \alpha^{102}x + \alpha^{21}$
8	$x^8 + \alpha^{175}x^7 + \alpha^{238}x^6 + \alpha^{208}x^5 + \alpha^{249}x^4 + \alpha^{215}x^3 + \alpha^{252}x^2 + \alpha^{196}x + \alpha^{28}$

Результирующий набор кодовых слов, обеспечивающих обнаружение и исправление ошибок, может быть получен путем деления многочлена, представляющего исходные данные в кодовых словах (который составлен как было описано выше) на порождающий полином. При этом коэффициенты полинома, который является остатком от такого деления, будут значениями соответствующих кодовых слов, используемых для последующего обнаружения и исправления ошибок: коэффициент при старшей степени – первое слово, коэффициент при нулевой степени – последнее слово.

В ГОСТ Р ИСО/МЭК 18004-2015 представлена схема, с помощью которой может быть выполнена генерация байтов коррекции (рисунок 8).

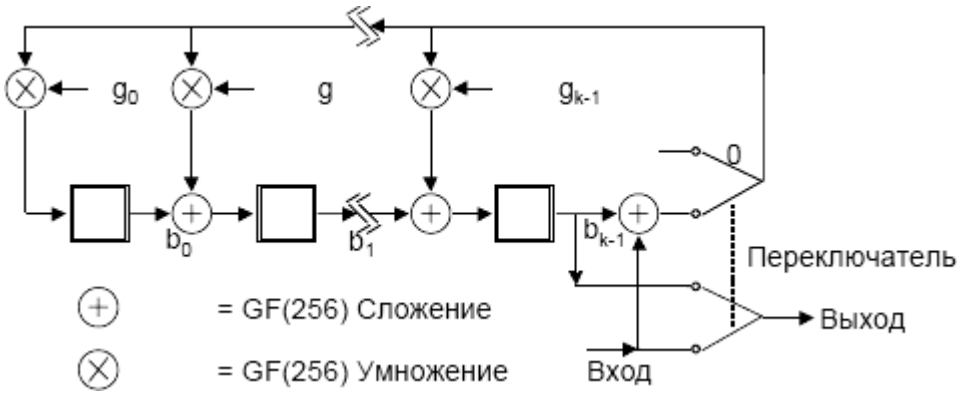

Рисунок 8 – Схема генерации байтов исправления ошибок

Представленный алгоритм создания байтов коррекции для каждого блока в разработанном скрипте выполнен в методе get_corr_bytes() (рисунок 9). В качестве аргументов он принимает поделенные на блоки данные, количество байтов коррекции для каждого блока (эта величина заранее известна и явно задана в виде списка), а также генерирующий многочлен (выбирается в зависимости от необходимого количества байтов коррекции). Также для упрощения нахождения байтов исправления ошибок в программе заранее явно заданы поля GF(256) и обратное ему в виде списков. Метод get_corr_bytes() использует значения полей для выполнения сложения, показанного в схеме на рисунке 8.

```
# кол-во байтов коррекции на один блок данных (для уровня коррекции H и версий 1, 2, 3, ...)
corr_bytes_by_block_h = [...]

# кол-во байтов коррекции для нашего случая:
amount_corr_bytes_by_block = corr_bytes_by_block_h[version - 1]

# генерирующие многочлены. Ключ - кол-во байтов коррекции, знач - коэффициенты полинома
generic_polys = {...}

# поля Галуа
inverse_galoi_256 = [...]
galoi_256 = [...]

# полином для нашего случая:
gen_poly = generic_polys[amount_corr_bytes_by_block]

# считаем корректирующие байты
correction_bytes = get_corr_bytes(data_block_divided, amount_corr_bytes_by_block, gen_poly)
```

Рисунок 9 – Часть реализованного приложения, генерирующая байты исправления ошибок

Этап 4. Объединение информационных и корректирующих блоков.

На следующем этапе происходит составление итоговой последовательности, которая затем будет записана с помощью темных и светлых модулей в символе QR кода. В общем виде процесс объединения кодовых слов, представляющих данные, а также использующиеся для исправления ошибок, можно представить следующим образом (рисунок 10). При этом если в выбранном блоке уже отсутствуют байты, то он пропускается.

	Кодовые слова данных						Кодовые слова исправления ошибок					
Блок 1	$D_{1,1}$	↓	$D_{2,1}$	↓	…		$E_{1,1}$	↓	$E_{2,1}$	↓	…	$E_{m,1}$ ↓
Блок 2	$D_{1,2}$	↓	$D_{2,2}$	↓	…		$E_{1,2}$	↓	$E_{2,2}$	↓	…	$E_{m,2}$ ↓
…	…	↓	…	↓	…	…	…	↓	…	↓	…	… ↓
Блок n-1	$D_{1,n-1}$	↓	$D_{2,n-1}$	↓	…	$D_{m,n-1}$ ↓	$E_{1,n-1}$	↓	$E_{2,n-1}$	↓	…	$E_{m,n-1}$ ↓
Блок n	$D_{1,n}$	↓	$D_{2,n}$	↓	…	$D_{m,n}$ ↓	$E_{1,n}$	↓	$E_{2,n}$	↓	…	$E_{m,n}$ ↓

Рисунок 10 – Создание итоговой последовательности кодовых слов

В большинстве случаев блоки данных и исправления ошибок полностью заполняют емкость символов QR кодов. Однако в некоторых случаях число модулей, составляющих область кодирования, не делится на 8 без остатка. В такой ситуации к завершенному двоичному потоку может быть добавлено 3, 4 или 7 нулевых битов, не представляющих данные, с целью полного заполнения области кодирования символа.

Объединение блоков данных и байтов исправления ошибок, реализованное в скрипте генерации QR кода, происходит с помощью метода combine_sys_data_corr_bytes() (рисунок 11). В качестве аргументов данный метод использует байты информации и байты исправления ошибок.

```
def combine_sys_data_corr_bytes(data, corr_data):
    ...
    ready_d_flow = []
    # сначала добавим данные
    num_B = 0  # номер текущего записываемого байта
    is_empty_d = False
    while not is_empty_d:
        is_empty_d = True
        for i in range(0, len(data)):
            if num_B <= len(data[i]) - 1:  # если еще есть в блоке данные с таким номером
                ready_d_flow.append(data[i][num_B])
                is_empty_d = False
        num_B += 1
    # теперь добавляем байты коррекции
    count_corr = len(corr_data[0])  # кол-во байтов коррекции в каждом блоке, оно одинаковое
    for i in range(0, count_corr):
        for j in range(0, len(corr_data)):
            ready_d_flow.append(corr_data[j][i])
    return ready_d_flow
```

Рисунок 11 – Метод combine_sys_data_corr_bytes() реализованного программного продукта

Этап 5. Размещение полученных данных в символе QR кода.

Изначально создается пустая квадратная матрица, состоящая из соответствующего используемой версии числа модулей по горизонтали и вертикали. Необходимо учесть, что рамка вокруг символа QR кода шириной в 4 модуля должна быть заполнена белыми модулями. Размер символа (в

модулях) без учета рамки будет равен сумме последней координаты для соответствующей версии из таблицы 4 и числа 7. Производится сумма именно с числом семь вследствие того, что в стандарте QR кода однозначно определено количество модулей, которые могут быть расположены далее, чем последний правый нижний выравнивающий узор.

Таблица 4 – Координаты (столбец/строка) центрального модуля направляющих шаблонов

Версия	Число направляющих шаблонов	Координаты (столбец/строка) центрального модуля		
1	0	-		
2	1	6	18	
3	1	6	22	
4	1	6	26	
5	1	6	30	
6	1	6	34	
7	6	6	22	38

Примечание – в данной таблице представлены координаты не для всех версий символов.

В первую очередь на символе QR кода располагаются поисковые шаблоны и полосы синхронизации. Их расположение и вид всегда одинаковы. Определив координаты центральных модулей направляющих узоров из таблицы 4, необходимо расположить их в соответствующих местах символа.

Если версия QR кода выше 7, то на символе необходимо расположить код версии. Часть кодов версий отображена в таблице 5.

Таблица 5 – Кодовые последовательности версий для QR кодов

Версия	Код версии
7	000010 011110 100110
8	010001 011100 111000
9	110111 011000 000100
10	101001 111110 000000

Далее необходимо определить информацию о формате. Это последовательность из 15 бит, 5 из которых – данные, 10 – биты исправления ошибок, вычисляемые по коду БЧХ (15, 5) [6].

В первых двух битах данных содержится обозначенный особым образом идентификатор, определяющий уровень возможного обнаружения и исправления ошибок при чтении символа.

В битах с третьего по пятый содержится указатель, соответствующий определенному шаблону маски данных (таблица 6). Его формирование происходит путем применения ко всем модулям в области кодирования (в обязательном порядке необходимо исключить область, содержащую информацию о формате кодирования и версии QR кода) маски, при совпадении условий истинности которой рассматриваемый темный модуль определяется как темный, а при несовпадении – как светлый. Заранее определено, что координаты каждого модуля определяются как позиция в строке (i) и позиция в столбце (j), причем координата (i, j) = (0, 0) задает верхний левый модуль символа (рисунок 12).

Таблица 6 – Условия, накладываемые масками данных

Идентификатор шаблона маски данных	Условие
000	$(i+j) \bmod 2=0$
001	$i \bmod 2=0$
010	$j \bmod 3=0$
011	$(i+j) \bmod 3=0$
100	$((i \operatorname{div} 2)+(j \operatorname{div} 3)) \bmod 2=0$
101	$(i\,j) \bmod 2+((i\,j) \bmod 3=0$
110	$((i\,j) \bmod 2+(i\,j) \bmod 3) \bmod 2=0$
111	$(i+j) \bmod 2+(i\,j) \bmod 3) \bmod 2=0$

Процесс маскирования необходим для установления баланса между количеством темных и светлых модулей, а также для сведения к минимуму вероятности присутствия на символе комбинаций модулей, обнаружение которых может помешать быстрой обработке и декодированию символа QR кода.

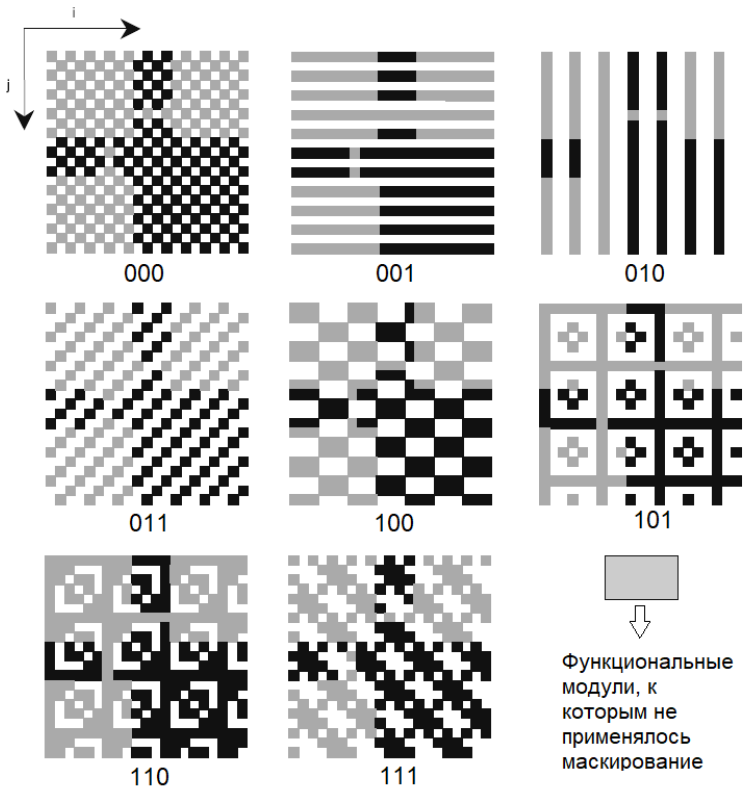

Рисунок 12 – Изображение масок данных для символа версии 1

Последовательность бит, содержащую информацию о формате, необходимо размещать в соответствующей области символа QR кода. Особое внимание нужно обратить на то, что данные о формате содержатся в двух местах символа, что обеспечивает избыточность. Использование такого способа размещения обеспечивает правильное декодирование информации о формате, необходимой для корректной интерпретации завершенного символа.

На последнем этапе происходит добавление полученной ранее последовательности байтов данных и исправления ошибок, а также маскирование этих данных.

Последовательность кодовых слов рассматривается как единый двоичный поток начиная со старшего бита. Размещение происходит в столбцах, имеющих ширину два модуля. В каждом из столбцов данные размещаются поочередно сперва в правых, а затем левых модулях. Перемещение происходит вверх или вниз согласно текущему направлению (начальное направление – вверх и далее поочередно изменяется вверху или

внизу столбца). Области символа, в которых находятся функциональные шаблоны, пропускаются. Схематично описанная схема расположения данных на символе QR кода показана на рисунке 13.

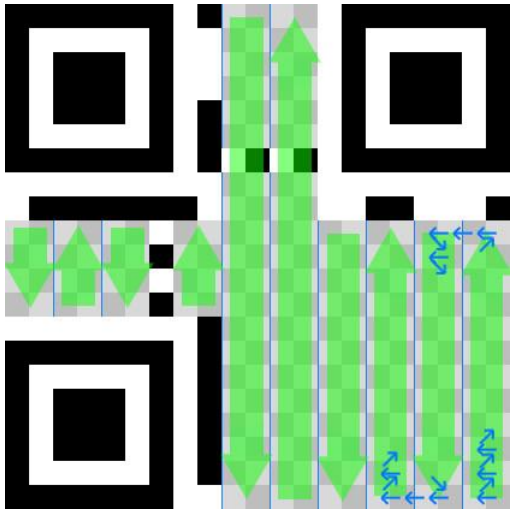

Рисунок 13 – Расположение кодовых слов в символе QR кода

Конечным шагом в размещении информации на символе QR кода является применение к модулям, не являющимся функциональными шаблонами, выбранной ранее маски и инвертировать модули, которые совпадают с ее темными областями [7].

Добавление данных в реализованном скрипте, используемом для генерации QR кода, происходит при вызове метода draw_data() (рисунок 14).

```
# записываем данные в qr код
draw_data(used_c_1, data_flow_bit, count_modules)

# записываем файл + рисуем в картинку
print_matr_file(used_c_1)
draw_qr(used_c_1, count_modules)
```

Рисунок 14 – Часть реализованного приложения, заполняющая символ данными и производящая отрисовку символа

На вход в качестве аргумента данный метод принимает единый двоичный поток и заполняет символ QR кода по алгоритму, описанному выше. Одновременно с добавлением очередного бита информации происходит проверка условия маскирования.

Метод draw_qr() создает файл изображения символа QR кода, определяет размер холста и единичного квадратного модуля а также переносит информацию из полученной матрицы на плоскость изображения (рисунок 15).

```
def draw_qr(matrix, c_modules):
    im_wid_heig = c_modules * 10
    module = im_wid_heig / c_modules  # pixels - ширина длина блока модуля
    image = Image.new("RGB", (im_wid_heig, im_wid_heig))
    draw = ImageDraw.Draw(image)
    # заливаем белый фон
    draw.rectangle((0, 0, im_wid_heig, im_wid_heig), fill=ImageColor.getrgb("white"))
    # рисуем блоки
    for i in range(len(matrix)):
        for j in range(len(matrix)):
            if matrix[i][j] == 1:
                draw.rectangle((j * module, i * module, j * module + module, i * module + module),
                               fill=ImageColor.getrgb("black"))
    image.save("my_qr.png", "PNG")
```

Рисунок 15 – Метод отрисовки QR кода в реализованном приложении

1.4 Алгоритм декодирования QR кода

В ГОСТ Р ИСО/МЭК 18004-2015 определен рекомендованный алгоритм, с помощью которого происходит обнаружение символов QR кодов и их декодирование.

В первую очередь происходит определение положения шаблонов поиска на изображении (рисунок 16). Каждый такой шаблон определяется последовательностью модулей «т-с-т-с-т» (т – темный, с – светлый). Относительные значения ширины элементов, обозначенных выше, находятся в соотношении 1:1:3:1:1.

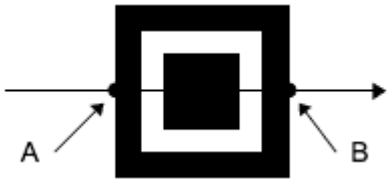

Рисунок 16 – Линия поиска для поискового шаблона

В случае если не было найдено ни одной линии, которая удовлетворяла бы такому условию, происходит изменение пикселей на инвертированные светлые и темные.

После нахождения всех трёх шаблонов поиска в первую очередь определяется угловая ориентация символа посредством анализа координат центральных модулей найденных шаблонов [8]. Угол поворота символа QR кода определяется после уточнения информации о шаблоне поиска, находящемся в левом верхнем углу символа.

Далее происходит определение версии символа QR кода. Для этого определяются величины D, W_{UL} и W_{UR}: расстояние между центрами верхних шаблонов поиска, ширины левого и правого верхних шаблонов поиска соответственно (рисунок 17).

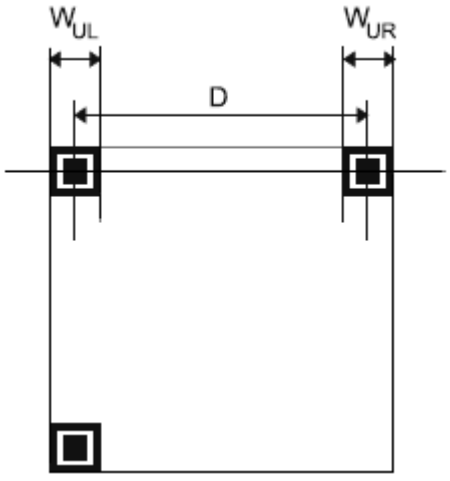

Рисунок 17 – Графическая интерпретация расстояний D, W_{UL} и W_{UR}

Номинальный размер символа вычисляется по следующей формуле:

$$X = (W_{UL} + W_{UR})/14 \quad (1)$$

Версию символа (приблизительную) можно вычислить по формуле:

$$V = [(D/X)-10]/4 \quad (2)$$

Если значение, полученное после подсчета по этой формуле, 6 или менее, то оно принимается в качестве окончательной версии. Если приблизительное значение получилось равным 7 или более, необходимо декодировать информацию о версии, анализируя модули, находящиеся в соответствующей области символа QR кода.

После определения версии символа QR кода происходит определение координат центров всех направляющих шаблонов, а затем происходит формирование сетки выборки пикселей для сканирования, которые расположены равноудаленно от этих точек (рисунки 18-19).

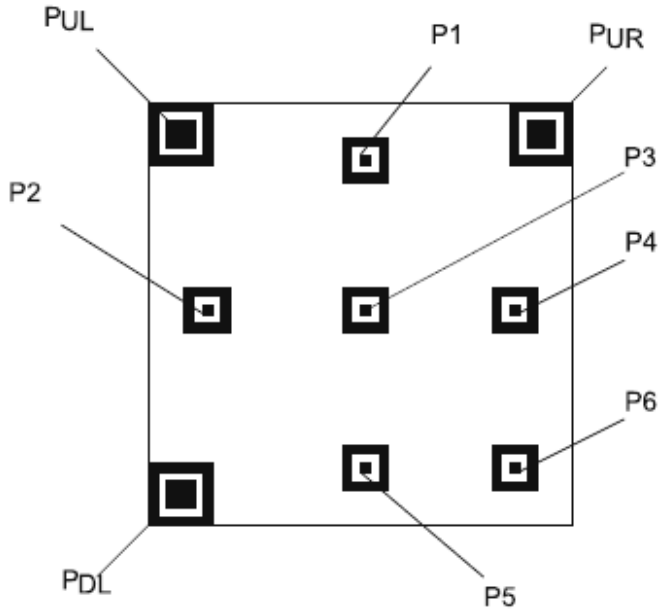

Рисунок 18 – Координаты центров шаблонов поиска и направляющих шаблонов

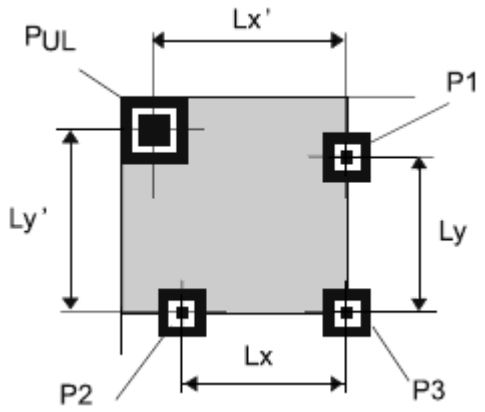

Рисунок 19 – Верхний левый квадрант символа

Аналогичным образом определяются области сканирования вокруг других двух шаблонов поиска, а также во всех неохваченных областях символа.

Затем формируется битовая матрица таким образом, что темным модулям присваивается двоичная 1, а светлым модулям – двоичный 0. Также из соответствующей области рядом с верхним левым направляющим шаблоном декодируется информация о формате, использующаяся для получения уровня исправления ошибок, используемого в данном символе QR кода, а также указателя, определяющего шаблон маски, примененной к области кодирования символа. Если при попытке получения информации о формате обнаружено количество ошибок, превышающее способность их исправления, происходит повторение описанной выше процедуры для области, находящейся рядом с правым верхним и нижним левым шаблонами поиска.

Если получение из считанной двоичной строки корректной информации о формате невозможно, предпринимаются попытки определения правильности последовательности битов, записанной в обратном порядке. В случае успешного декодирования «инверсированной» строки декодирование изображения продолжается уже для зеркально отображенного символа QR кода с заменой координат строк и столбцов.

На заключительном этапе декодирования происходит выполнение процедуры демаскирования: операция «XOR» применяется к шаблону маски и области кодирования символа, в результате чего обеспечивается восстановление двоичного потока, представляющего данные и кодовые слова исправления в этих данных ошибок. Этот процесс – обратен процедуре маскирования области кодирования, которая применяется при размещении данных на символе QR кода. Также происходит определение кодовых слов символа согласно правилам их размещения.

После этого осуществляется перегруппировка кодовых слов по блокам таким образом, как этого требует определенная версия символа и уровень исправления ошибок. На этом же этапе проводится декодирующая процедура обнаружения и исправления ошибок. Затем происходит восстановление первоначального двоичного потока сообщения, посредством сбора блоков данных в последовательность и декодирования каждого сегмента из битового формата в соответствии с указанным режимом.

В результате изучения закрепленных государственным стандартом структуры символа QR кода и особенностей его построения и декодирования получены все необходимые знания, результатом применения которых является программа генерации различных типов QR кодов произвольного текста (номер телефона, гиперссылка, электронное письмо), которая в свою очередь является частью веб-ресурса учета посещаемости занятий в вузе.

2 Автоматизация учета посещаемости студентов

Контроль знаний и умений обучающихся является неотъемлемой и обязательной частью образовательного процесса [9]. При его отсутствии у преподавателя не будет возможности достоверно и точно оценить текущий уровень подготовки студента. В то же время, качественно организованный процесс контроля может предоставить конкретные данные, необходимые для управления учебным процессом и внесения изменений для повышения его эффективности. Для организации наиболее эффективного процесса проверки знаний и умений студентов необходимо соблюдение следующих условий:

1) контроль должен осуществляться планомерно и систематически;

2) быть объективным и позволяющим всесторонне оценить фактический уровень усвоения студентом знаний а также успехи и недостатки текущего способа организации учебной деятельности;

3) быть как можно более экономичным по затратам времени преподавателя и студента.

Качество и количество усваиваемых обучающимся знаний и умений напрямую зависит не только от способностей студента, но и от количества посещаемых им занятий. Также широко применяется модульно-рейтинговая система оценки качества подготовки, которая учитывает как оценки, получаемые студентом за выполненные задания, так и посещаемость занятий. Стоит отметить, что наличие информации о посещаемости студентами учебных занятий позволяет сделать выводы о влиянии пропусков на успеваемость, а также при необходимости принять какие-либо меры в отношении пропускающих занятия без уважительной причины студентов.

В ходе анализа работы системы учета посещаемости занятий в нашем ВУЗе выявлено, что для контроля используются данные, предоставляемые старостами учебных групп в бумажном виде. Также стоит заметить, что человеческий фактор, устаревшая форма отчетности, трудоемкость обработки и аккумуляции информации, а также отсутствие возможности оперативно

контролировать посещаемость не позволяет оптимизировать процесс учета посещаемости занятий студентами.

Таким образом, сделан вывод о необходимости создания программного продукта для решения задач представления и сбора информации о посещаемости студентов в электронном виде. Выполним анализ преимуществ существующих разработок контроля посещаемости.

2.1 Система «Электронный университет» МТУСИ

Гуриков С.В. в своей статье описывает процесс создания электронного модуля учета посещаемости [10]. К основным особенностям данного программного продукта относятся:

1) реализация проверки доступа к внесению данных и процесса авторизации;

2) возможность добавления студентов в список группы, просмотра статистики каждого студента;

3) функция отображения данных по текущему семестру;

4) возможность создания отчетности по каждой учебной группе с последующей выгрузкой данных в Microsoft Excel;

5) функция поиска пользователей и просмотра информации о них;

Программный модуль содержит страницы групп, содержащие информацию о количестве студентов, старосте, количество часов учебных занятий, пропущенных студентами. Также у старост есть возможность внесения данных о пропусках занятий в систему, а у остальных студентов – возможность просмотра своей собственной статистики и статистики группы. Для увеличения вовлеченности студентов в модуле формируется список из пяти лучших студентов и групп, которые имеют наименьшее количество пропусков занятий.

Модуль имеет адаптивный дизайн, предоставляющий пользователям доступ к сервису с любого устройства – компьютера или смартфона – что в

свою очередь позволяет старостам групп оперативно вносить данные о присутствующих на занятиях студентах, находясь непосредственно в аудитории.

Автоматизация подразумевает сведение к минимуму участие человека в сборе и обработке информации, однако при использовании данного модуля старостам учебных групп необходимо самостоятельно собирать данные об отсутствующих на каждом занятии и затем вносить их в базу данных сервиса. Поэтому главным недостатком электронного модуля учета посещаемости, предложенного Гуриковым С.В., является отсутствие автоматического учета посещаемости.

2.2 Программа «Учет текущей успеваемости»

В статье «Автоматизация учета текущей успеваемости студентов» Ведерниковой Т.А., Родионова А.В. и др. представлена программа «Учет текущей успеваемости», предназначенная для оперативного получения информационных отчетов о текущей успеваемости и посещаемости студентов, а также позволяющая на их основе принимать решения по корректировке учебного процесса [11].

Данная программа может быть использована для автоматизации учета успеваемости студентов, формирования оценки промежуточной аттестации, контроля посещаемости и выполнения заданий. К основным функциям программы относятся: отображение учебных групп и предметов, преподавателей; добавление заданий и их настройка для предмета или группы; оценивание студентов по каждому заданию; учет посещаемости студентов; хранение информации в базе данных; экспорт данных о посещениях, успеваемости в Microsoft Office Excel.

Программа «Учет текущей успеваемости» имеет двухзвенную архитектуру (рисунок 20). Базовые компоненты – уровни представления,

бизнес логики и доступа к данным – в этом случае распределены между двумя узлами – клиентом и сервером.

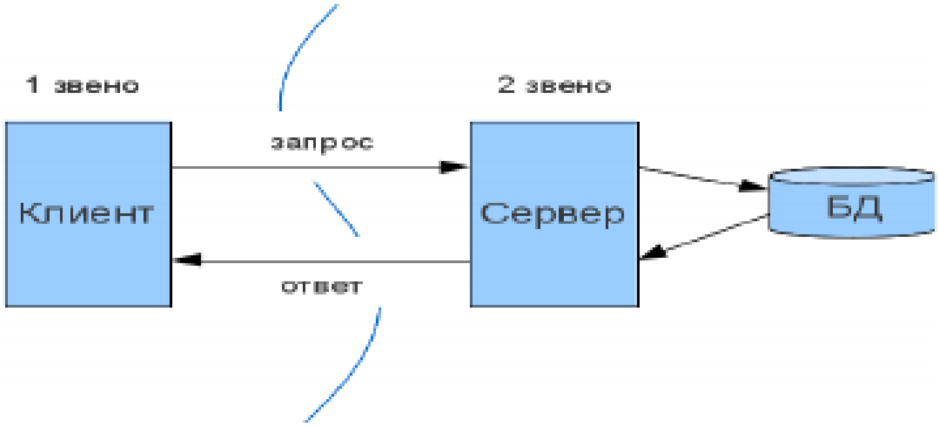

Рисунок 20 – Двухзвенная архитектура

Серверная часть программы – база данных – реализована с помощью Microsoft SQL Server 2008. Все правила и логики работы программы содержатся в хранимых процедурах SQL-сервера. Пользователь имеет возможность производить манипуляции с данными только посредством вызова этих процедур, хранящихся на сервере.

Клиентская часть программы выполнена на языке C# с использованием компонентов Windows Forms. Примеры интерфейсов окон программы представлены на рисунках 21 и 22. Также в программе предусмотрен подсчет успеваемости студента с использованием формулы балльно-рейтинговой системы, целью которой является оценка качества работы студента при освоении им образовательной программы.

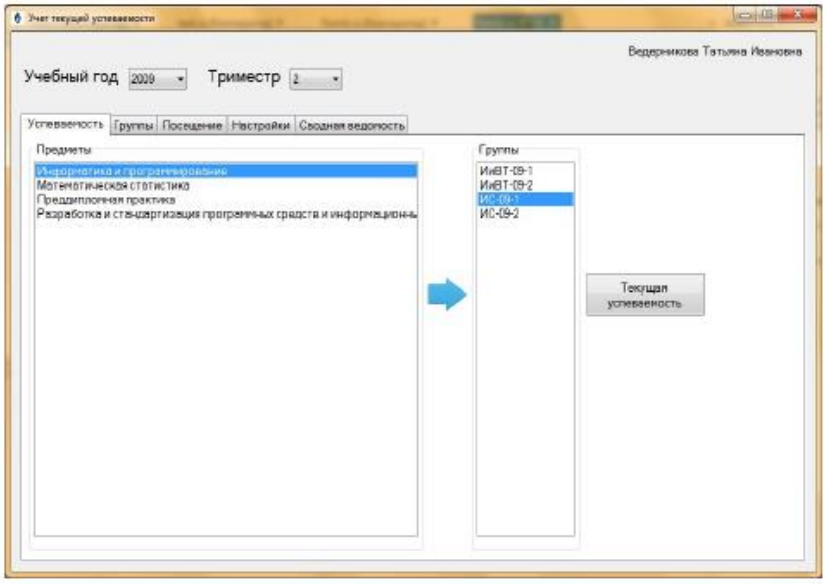

Рисунок 21 – Главное окно программы «Учет текущей успеваемости»

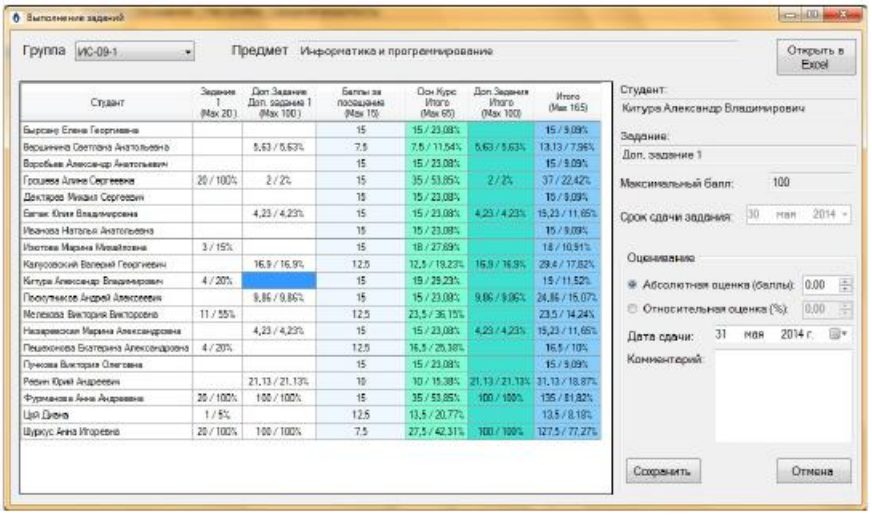

Рисунок 22 – Форма «Выполнение заданий» программы «УТУ»

Несомненным плюсом данной программы является автоматическое выполнение расчета успеваемости студента по специальной формуле, что представляет точную аккумуляцию всей информации о посещаемости и успеваемости каждого студента. Однако возможность использования программы без использования дополнительного ПО (программ-эмуляторов, компиляторов) только на персональном компьютере с установленной операционной системой Windows является большим недостатком данного программного решения. Отсутствие автоматизации сбора информации о

посещаемости студентов также является очевидным недостатком программы «Учет текущей успеваемости».

2.3 Электронная система учета посещений СибФУ

Студенткой Сибирского федерального университета была предложена и реализована система учета посещений, содержащая RFID-метки [12]. В СибФУ, как и в Кубанском государственном университете, внедрена система контроля входа в здания учебных корпусов, имеющая в своем составе турникеты и банковские карты со специально настроенными RFID-метками. Преподаватели и студенты допускаются ко входу в университет только после сканирования метки считывающим устройством. Работа Салтынюк О.В. основывается на использовании данной системы для учета посещаемости студентов.

На рисунке 23 представлена структурная схема системы, предложенной Салтынюк О.В. Помимо считывающих устройств, используемых для чтения RFID-меток с карт студентов, в системе содержится серверная часть, отвечающая за сохранение и обработку получаемой информации о посещениях. Также на схеме в общих чертах представлены потоки передачи информации между ее структурными элементами.

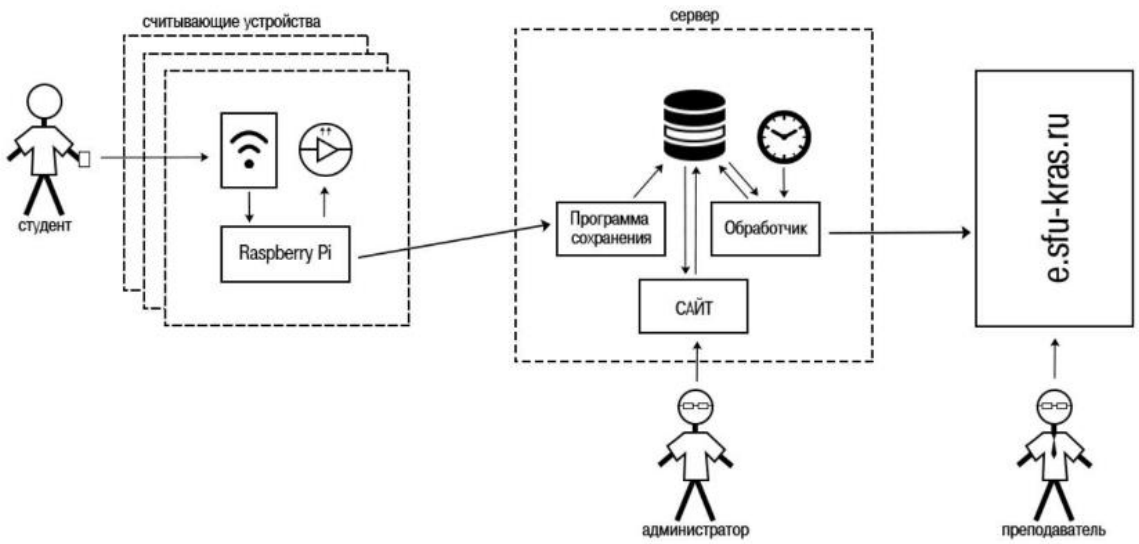

Рисунок 23 – Структурная схема электронной системы учета посещаемости СибФУ

В случае Сибирского федерального университета использование считывающих устройств для учета посещаемости оправдано, так как для входа в каждый кабинет используется электронный ключ и задача установки микрокомпьютеров и считывающих устройств для разрабатываемой системы частично решена. Однако, для обработки и отправки данных на сервер необходимо наличие специального модуля, который и был разработан в первой части работы Салтынюк О.В. Несомненно, для реализации таких модулей необходимы особые технические компоненты, а при условии масштабирования системы учета на все аудитории университета в огромном количестве, на получение которых требуется выделение денежных средств. Также стоит сказать и о расходах денежных и человеческих ресурсов на сборку и интеграцию модулей в имеющуюся пропускную систему. Все вышеперечисленное без сомнения является недостатком данной системы.

В результате изучения и анализа имеющихся проектов и разработок в сфере учета успеваемости и посещаемости студентов выявлена информация о достоинствах и недостатках каждой из систем. Задаче минимизации использования сложных технических средств (специальные RFID-модули)

необходимо поставить высокий приоритет. Также стоит учитывать необходимость автоматизации процесса учета посещаемости для сведения к минимуму действий со стороны студента и преподавателя для ведения и хранения информации о посещениях. Учесть все вышеперечисленные требования может система, использующая для учета посещаемости QR коды и специальный веб-интерфейс.

3 Проектирование и реализация веб-ресурса учета посещаемости занятий

В предлагаемой выпускной квалификационной работе представлен веб-ресурс, предоставляющий функционал для реализации учета посещаемости учебных занятий в вузе. Он служит заменой уже существующим журналам, находящимся у старост учебных групп – автоматизация процесса сбора информации о посещениях занятий и упрощение хранения исторических данных при использовании только бумажных носителей в качестве основных источников абсолютно невозможна. Все перечисленные проблемы призван решать разработанный веб-ресурс.

3.1 Структура веб-ресурса и используемый стек технологий

Для обеспечения реализации грамотного архитектурного подхода к построению и дальнейшей разработке веб-ресурса принято решение отделить хранилище данных от клиента и разместить его на отдельном сервере [13]. Структура системы представлена на диаграмме компонентов (рисунок 24).

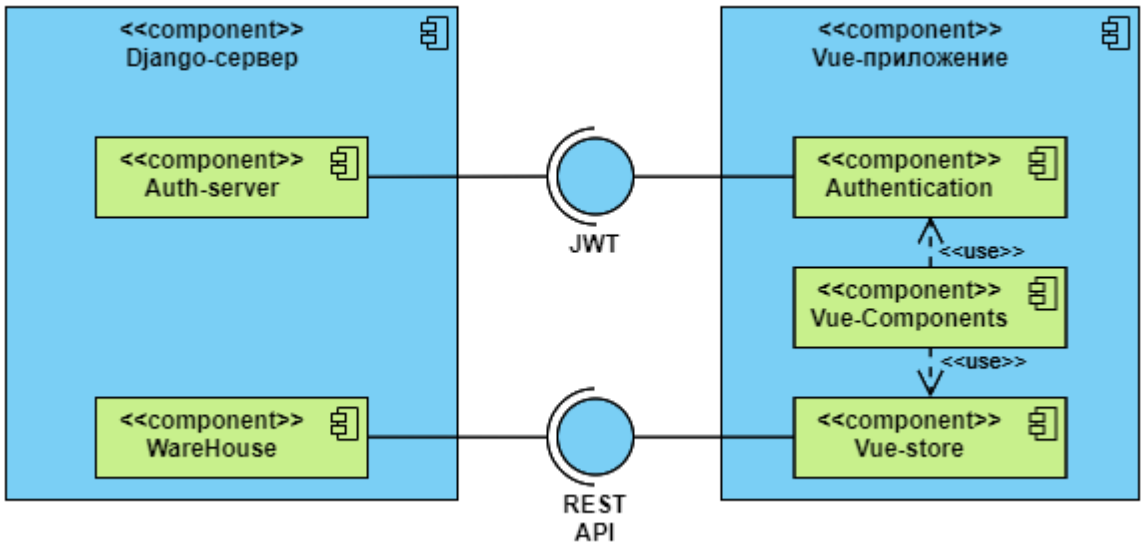

Рисунок 24 – Диаграмма компонентов разработанной системы

Реализованный с использованием фреймворка Django сервер включает в себя две компоненты: Auth-server, предоставляющий сервер для аутентификации и авторизации пользователей в системе, и WareHouse, использующий базу данных SQLite для реализации хранилища данных о пользователях, посещениях, учебных дисциплинах и т.д. Также сервер предоставляет интерфейсы JWT (JSON Web Token) и REST API для использования функций Auth-сервера и WareHouse со стороны клиентского приложения [14].

Компонент «Vue-приложение», разработанный с использованием современного фреймворка VueJS, при помощи vue-компонент предоставляет пользователям веб-интерфейс для взаимодействия с системой. Хранение необходимых данных и логика авторизации происходит с использованием специальных компонент «Vue-store» и «Authentication» внутри клиентского приложения. Необходимо отметить, что для упрощения разработки и стилизации vue-компонент используется библиотека «Bootstrap-Vue», предоставляющая простые инструменты для реализации всемирно известного пакета CSS-стилей и grid-системы Bootstrap.

Все перечисленные выше технологии и библиотеки являются современными, динамично развивающимися и постоянно обновляемыми и дополняемыми, что позволяет не только использовать всевозможные новые особенности и специальные возможности, но и улучшать быстродействие и безопасность веб-ресурса в целом. Также стоит отметить наличие понятной документации и большого количества справочных материалов и статей по использованию данных инструментов разработки.

3.2 Основные функции веб-ресурса

При проектировании и разработке веб-ресурса учитывались варианты использования, представленные на рисунке 25 [15].

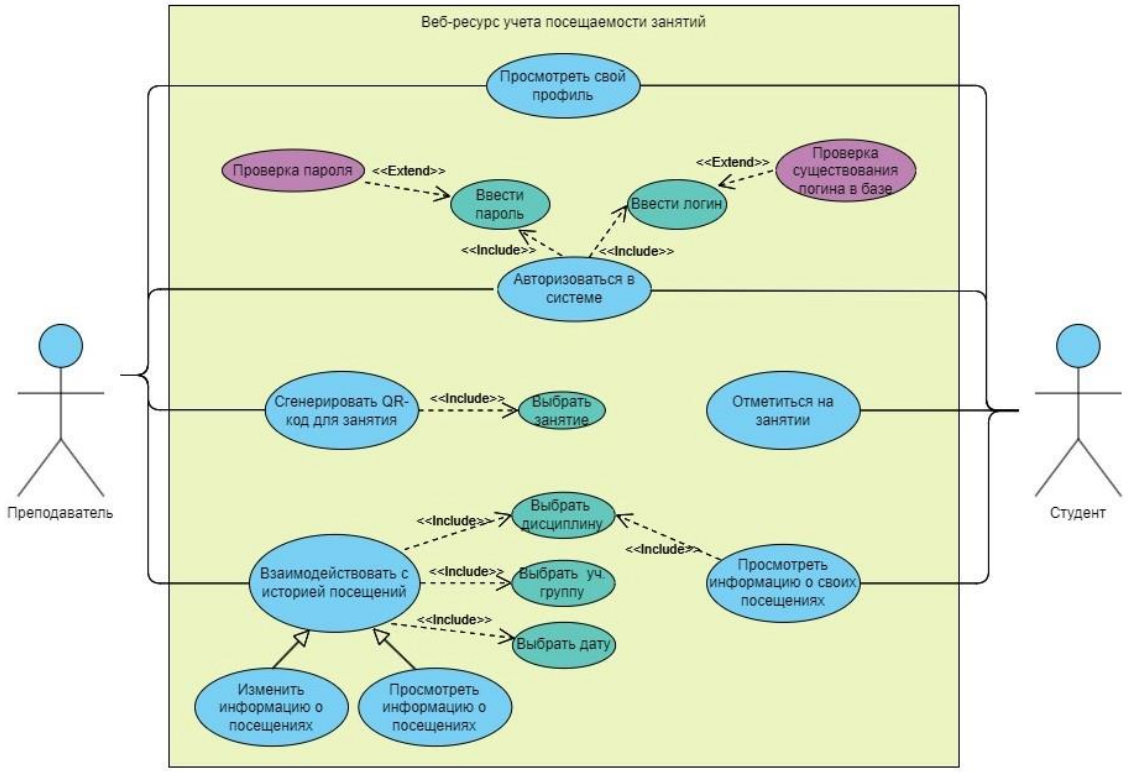

Рисунок 25 – Диаграмма вариантов использования разработанной системы

Описание используемой базы данных, в которой хранится информация о пользователях веб-ресурса, истории посещений учебных занятий, преподаваемых дисциплинах, студентах и преподавателях представлено в виде ER-диаграммы (рисунок 26) [15].

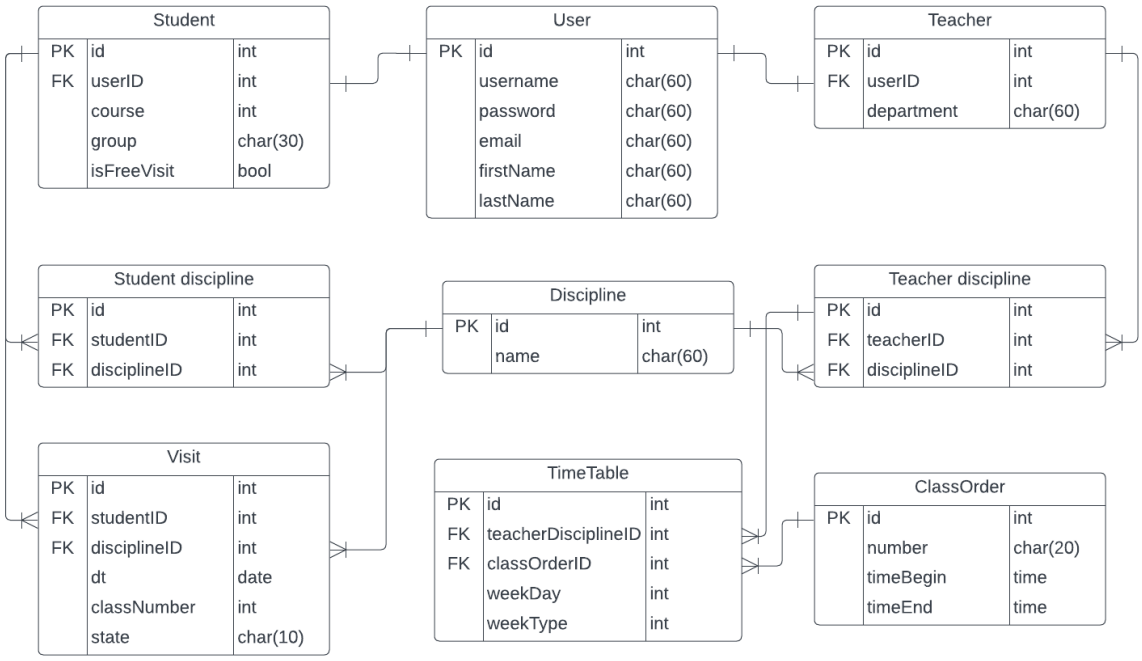

Рисунок 26 – ER-диаграмма базы данных

В разработанном веб-ресурсе учета посещаемости занятий в вузе пользователи разделены на две группы, каждая из которых обладает своими правами доступа: «студенты» и «преподаватели».

Для обеих групп пользователей формы авторизации в системе, просмотра личного профиля, а также шапка сайта являются идентичными (рисунки 27-30). Исключение составляют лишь данные, отображаемые после авторизации на странице «Профиль» и в шапке сайта. Они варьируются в зависимости от информации, хранящейся по авторизованному пользователю в базе данных.

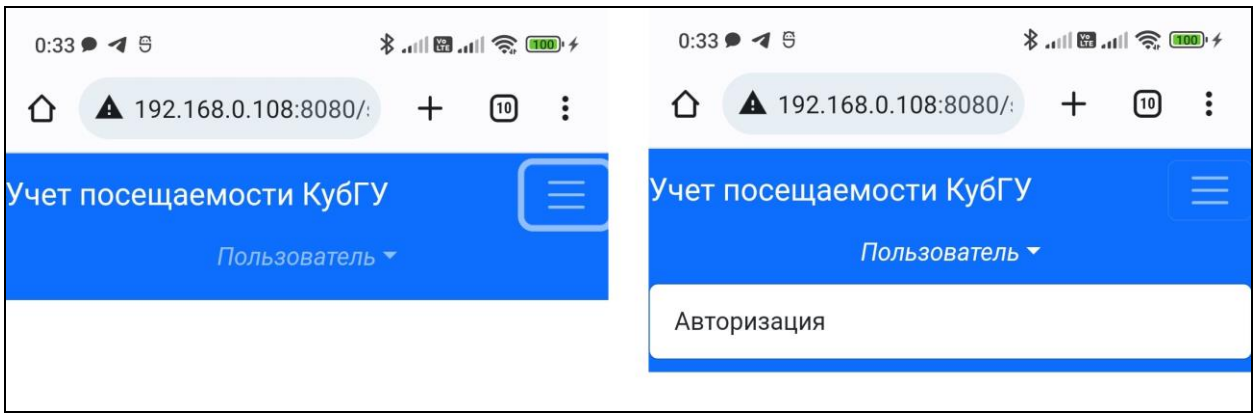

Рисунок 27 – Шапка сайта (неавторизованный пользователь)

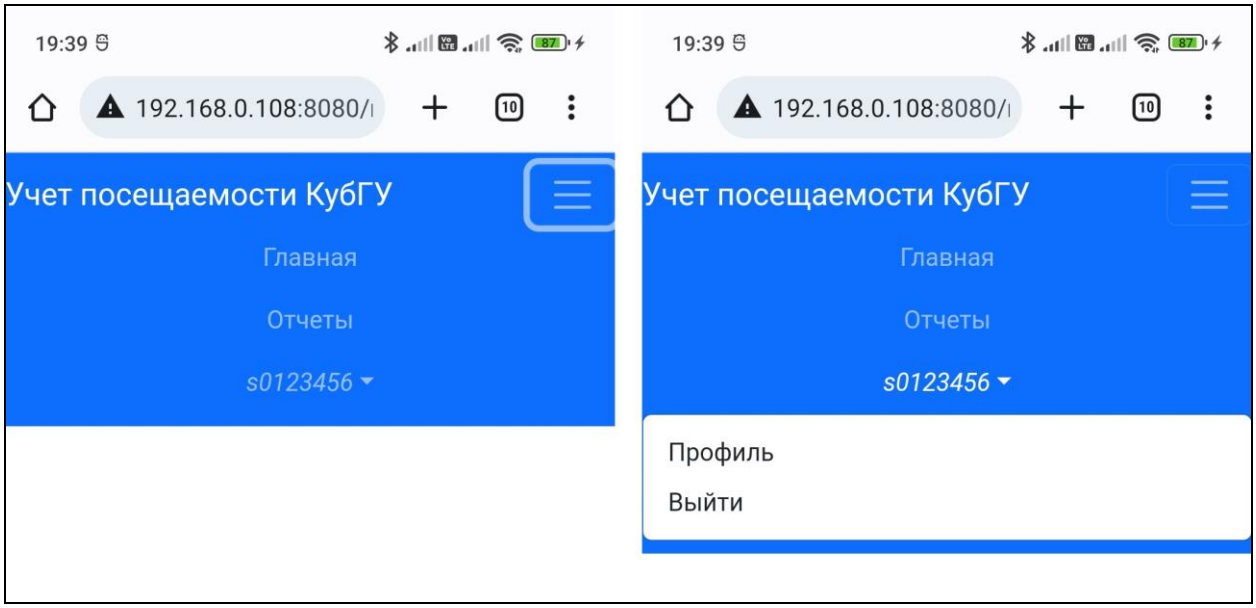

Рисунок 28 – Шапка сайта (авторизованный пользователь)

При помощи ссылок, расположенных в шапке сайта, пользователь может перемещаться по всем доступным ему на текущий момент разделам сайта, а также совершить выход из своей учетной записи, если он был ранее авторизован в системе.

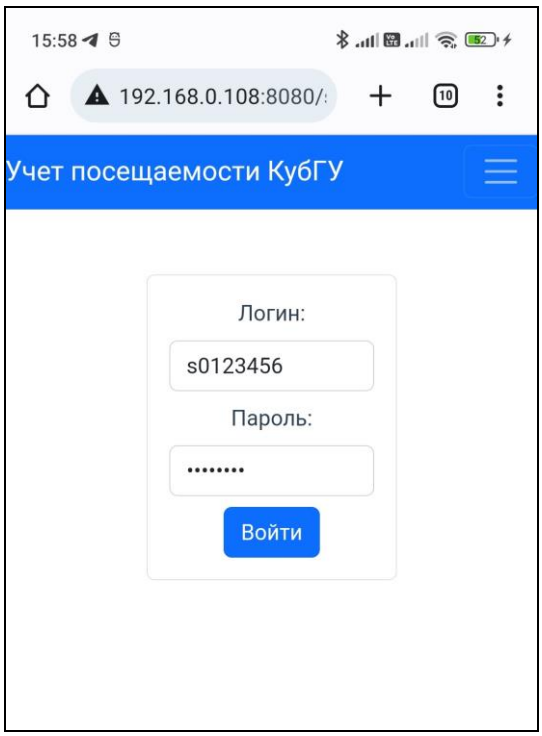

Рисунок 29 – Страница авторизации пользователя в системе

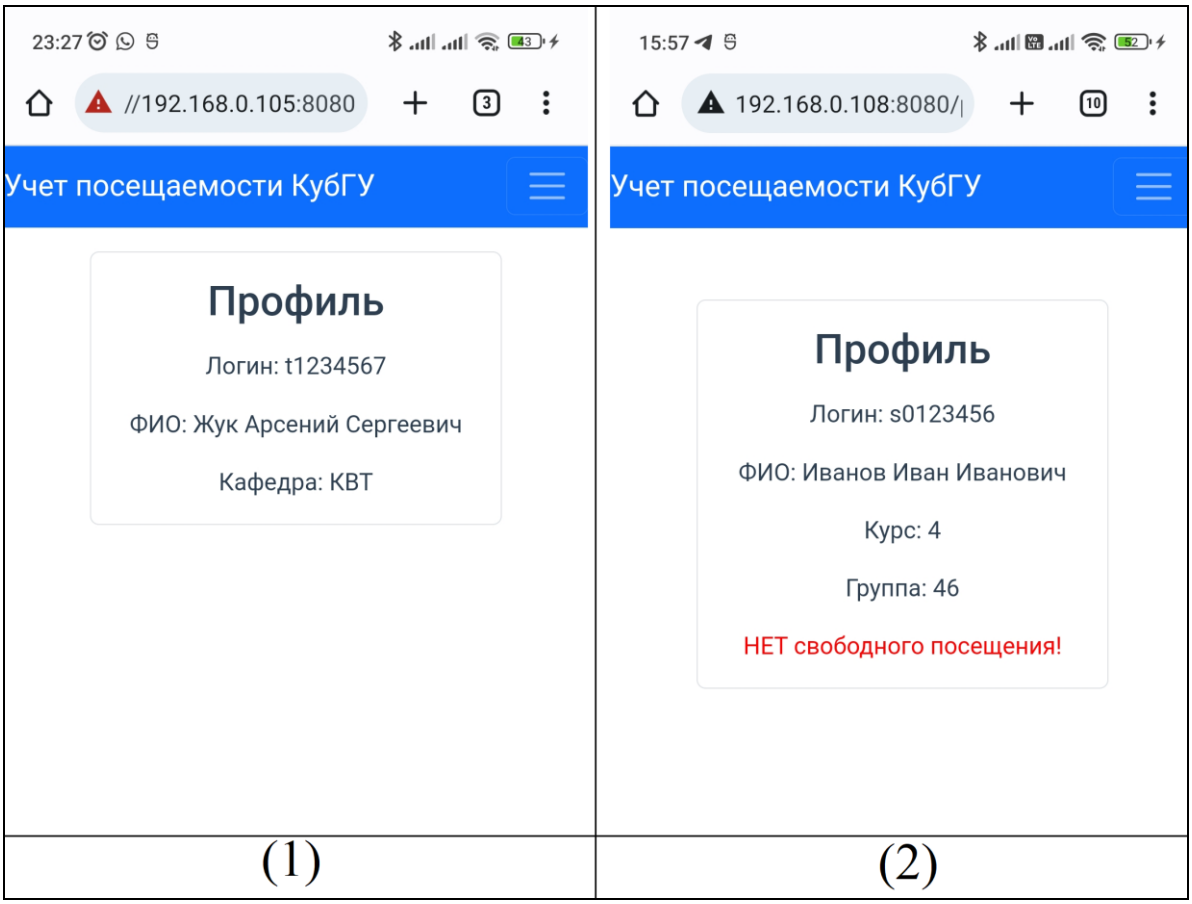

Рисунок 30 – Страница «Профиль» для авторизованного преподавателя (1) и студента (2)

После прохождения авторизации пользователи перенаправляются на главную страницу сайта (рисунок 31)

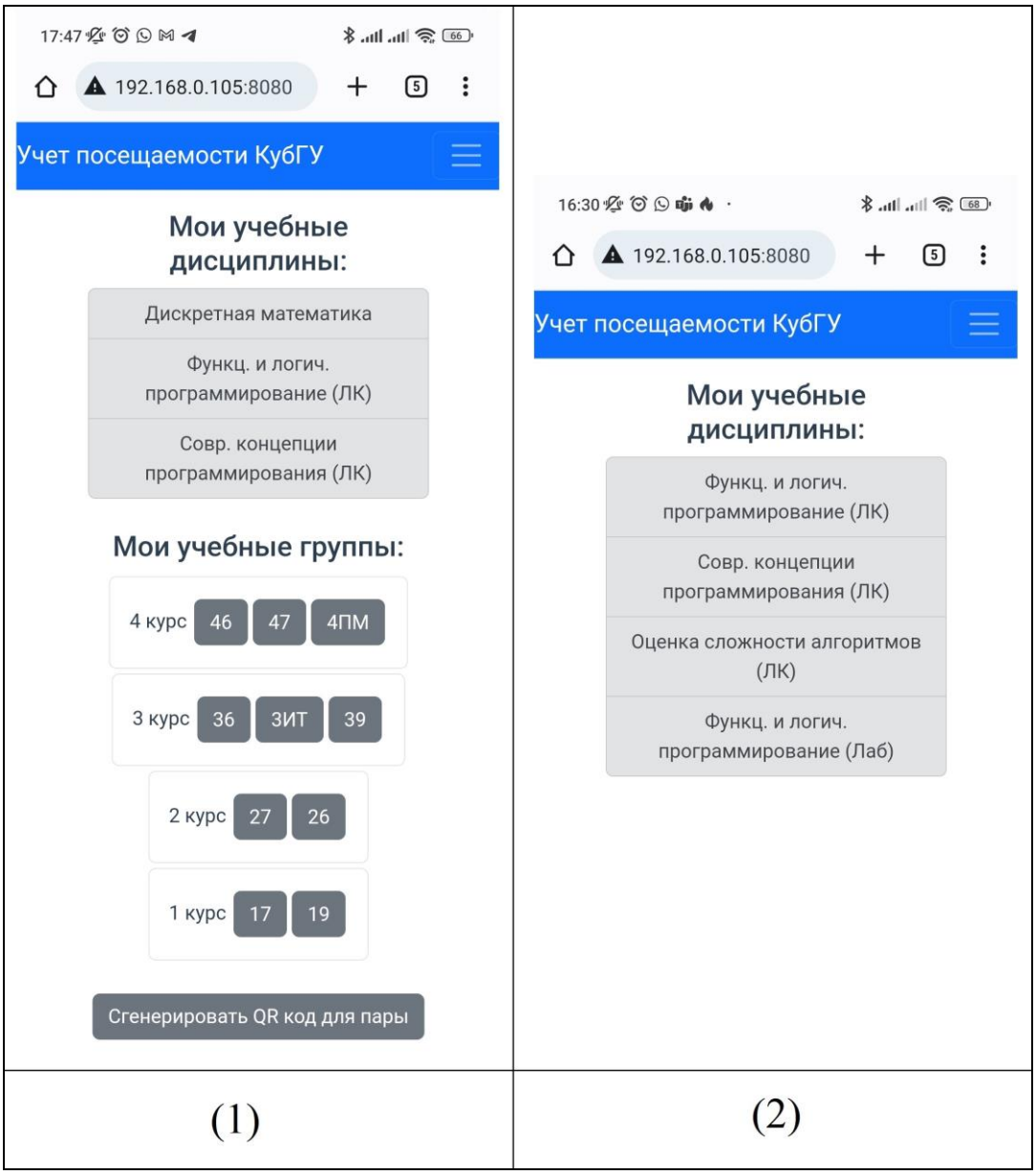

Рисунок 31 – Главная страница для преподавателя (1) и студента (2)

Пользователи, составляющие группу «студенты», имеют возможность просмотра только своих отчетов о посещаемости занятий по различным учебным дисциплинам. Переход на страницу с отчетами осуществляется посредством меню «Мои учебные дисциплины» на главной странице веб-ресурса и кнопки «Отчеты» в шапке. Предусмотрена возможность выбора дисциплины при помощи раскрывающегося списка (рисунок 32).

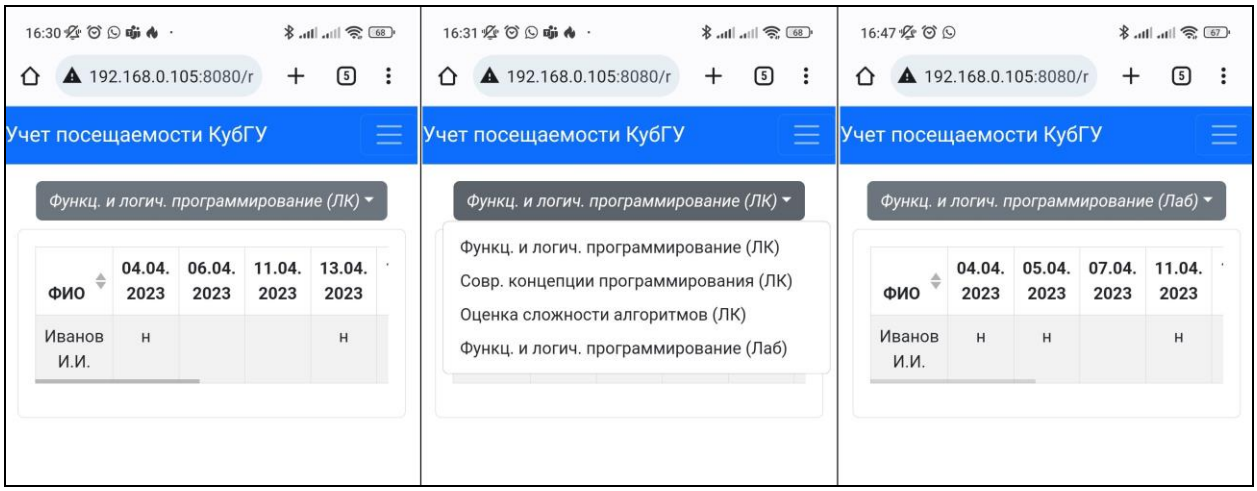

Рисунок 32 – Страница «Отчеты» для авторизованного студента

Главной отличительной особенностью данной группы пользователей является механизм, с помощью которого студенты самостоятельно производят отметку об их присутствии на занятии (рисунок 33). Для его реализации необходимо отсканировать сгенерированный преподавателем QR-код и перейти по временной ссылке на форму подтверждения присутствия на учебном занятии, после чего предоставить доступ к местоположению устройства для проверки нахождения геометки студента допустимом радиусе от метки преподавателя. Если пользователь не был авторизован в системе ранее, необходимо ввести логин и пароль.

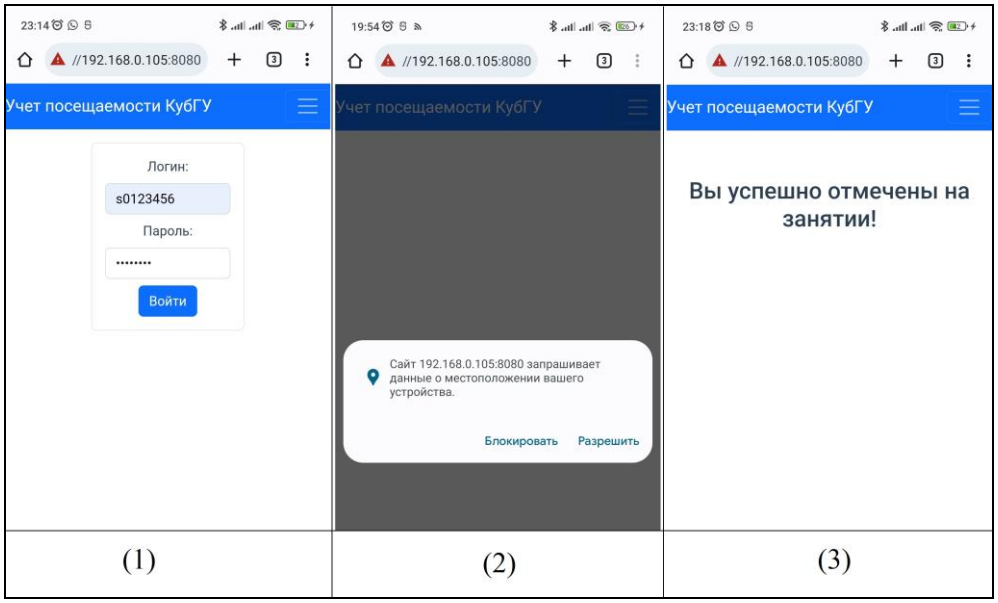

Рисунок 33 – Процесс получения отметки о присутствии на занятии

Пользователи, составляющие группу «преподаватели», имеют возможность просмотра отчетов о посещаемости занятий по назначенным им дисциплинам и учебным группам. Отличительными особенностями данной группы пользователей является возможность генерации QR-кода, содержащего ссылку на специальную форму подтверждения присутствия студента на занятии (рисунок 34), а также функционал, предоставляющий право изменения уже существующих данных о посещаемости занятий. Очевидно, что его использование необходимо при возникновении нестандартных ситуаций и ошибок в работе системы.

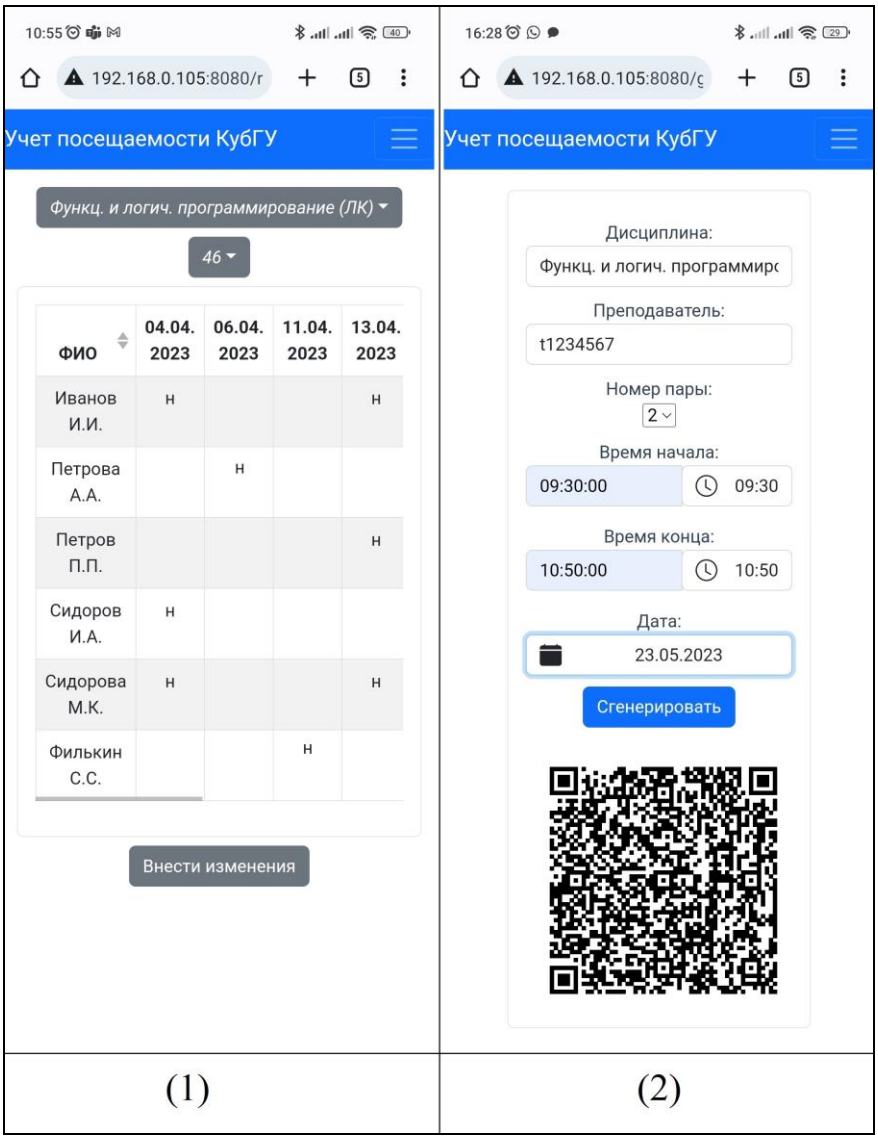

Рисунок 34 – Страницы «Отчеты» (1) и «Генерация QR кода занятия» (2) для авторизованного преподавателя

Преподаватель также имеет возможность выбора дисциплины и учебной группы при помощи выпадающих списков в верхней части страницы «Отчеты».

Все поля формы, расположенной на странице генерации QR кода для занятия по возможности заполняются автоматически. Для этого используются данные об авторизованном преподавателе, ближайшем занятии из расписания (таблицы «TimeTable» и «ClassOrder») и текущая дата. При необходимости преподаватель может внести изменения в данные формы.

ЗАКЛЮЧЕНИЕ

Цель выпускной квалификационной работы – разработать веб-ресурс учета посещаемости занятий в вузе на основе применения QR кодов – достигнута.

В результате выполнения выпускной квалификационной работы изучены режимы кодирования, использующиеся при подготовке информации для размещения на символе QR кода. Создан скрипт на языке Python, позволяющий генерировать QR код заданного текста в необходимом формате с использованием универсального байтового режима кодирования. Для отрисовки изображения символа QR кода также изучена его структура и правила размещения функциональных шаблонов.

Посредством сравнительного анализа существующих систем учета посещаемости и успеваемости студентов выявлены их достоинства и недостатки, а также составлены требования к разрабатываемому веб-ресурсу. Рассмотрены различные способы автоматизации процессов сбора и хранения информации, необходимой для контроля наличия объектов на мероприятии.

В практической части выпускной квалификационной работы реализован веб-ресурс учета посещаемости занятий в вузе, который при внесении небольших изменений может быть использован для других задач идентификации. Использование предложенной системы позволит упростить процессы сбора и обработки информации об участниках различных мероприятий.

Результаты работы представлены на V всероссийской научно-практической конференции молодых ученых "Прикладная математика: современные проблемы математики, информатики и моделирования". Опубликована статья «Технология генерации QR-кода как основа учета посещаемости занятий в вузе»

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Захожая, И.О. QR-код и его использование в современном мире // Научные междисциплинарные исследования / И.О. Захожая. 2021. №2. С.30-33.

2 ГОСТ Р ИСО/МЭК 18004-2015. Информационные технологии. Технологии автоматической идентификации и сбора данных. Спецификация символики штрихового кода QR Code : национальный стандарт Российской Федерации : издание официальное : утвержден и введен в действие Приказом федерального агентства по техническому регулированию и метрологии от 3 июня 2015 года №544-ст : введен впервые : Дата введения 2016-02-01 / разработан Ассоциацией автоматической идентификации «Юнискан/ГС1 РУС» совместно с ООО НПЦ «Интелком». – Москва :Стандартинформ, 2015. – II, 107 с. – (Система межгосударственных стандартов).

3 Конопелько, В.К. Кодирование информации в инфокоммуникациях // Доклады БГУИР / В.К. Конопелько, М.Н. Бобров, В.Ю. Цветков. 2014. №2 (80).

4 Кузьмин, О.В. Коды Боуза–Чоудхури–Хоквингема в системах обнаружения и исправления ошибок при передаче данных // Современные технологии. Системный анализ. Моделирование / О.В. Кузьмин, В.И. Дружинин. 2013. №3 (39).

5 Постников, М.М. Теория Галуа / М.М. Постников. – М.: «Факториал Пресс», 2003. – 304 с. – ISBN 5-88688-063-1.

6 Гошин, Е.В. Теория информации и кодирования : учебное пособие / Е.В. Гошин; Министерство образования и науки Российской Федерации, Самарский университет. — Самара: СамГУ, 2018. — 124 с. — ISBN 978-5-7883-1260-6.

7 Крашенинников, В.Р. Алгоритм оценивания сдвига и поворота изображений на основе метода неподвижной точки // Известия Самарского

научного центра РАН / В.Р. Крашенинников, А.Д. Кадеев. 2013. №4-4. С.931-935

8 Подколзина, Л.А. Аналитический обзор методов распознавания двумерных штрихкодов // Молодой исследователь Дона / Л.А. Подколзина. 2018. №1 (10). С.53-56.

9 Николаенко, Г.А. Перспективы использования QR-кодировки в академической сфере // Социология науки и технологий / Г.А. Николаенко, Е.В. Евсикова. 2015. №2. С.109-118.

10 Гуриков, С.Р. Автоматизация учета процесса посещаемости студентов // Педагогика и психология: актуальные вопросы теории и практики / С.Р. Гуриков, О.А. Борисова. 2017. С.56-59.

11 Ведерникова, Т.И. Автоматизация учета текущей успеваемости студентов // Baikal Research Journal / Т.И. Ведерникова, А.В. Родионов, В.В. Блудов, Д.А. Пичкур. 2019. №4.

12 Болотнова, Е. А. Анализ современных методов влияния QR-кодов на жизнь человека в современном мире // Естественно-гуманитарные исследования / Е. А. Болотнова, Б. И. Павлишин, В. К. Барейша. 2020. №29 (3). С.64-67.

13 Сукиасян, В.М. Современные принципы и подходы к Frontend архитектуре веб-приложений // Наука, техника и образование / В.М. Сукиасян, Е.С. Придиус. 2019. №10 (63).

14 Чеглаков, А.Л. Композиция web-сервисов на основе архитектуры REST // Инновационная наука / А.Л. Чеглаков. 2016. №12-2. С.118-120.

15 Фаулер, М. UML. Основы, 3-е издание. / М. Фаулер [пер. с англ.]. — Санкт-Петербург : Символ-Плюс, 2004. — 192 с., ил. — ISBN 5-93286-060-X.