В качестве среды разработки кроссплатформенного приложения был выбран Python 3.6. Такой выбор был сделан в силу наличия отличного фреймворка Kivy [5]. Kivy – инструмент разработки кроссплатформенных мобильных приложений. Данный фреймворк имеет огромное количество поддерживаемых инструментов и способов построения приложений. Так же Kivy представляет возможность написания собственных файлов расширения. kv, где можно описывать структуру работу окон приложения и их функциональность с привязкой к разрабатываемым классам.

Список используемых методов в системе для написания прототипа приложения:

1. from kivy.uix.screenmanager – позволяет наследовать встроенный класс Screen, для сопоставления разрабатываемого класса и необходимого окна исполнителя. Следующее классы наследники так же будут наследовать базовый класс Screen, что позволит унаследовать структуру построения окна.
2. from kivy.app import App – класс наследник от класс App будет исполняющим классом. Обычно класс наследник должен возвращать Builder.load\_file("style.kv"), который будет запускать общий вид приложения.
3. from kivy.uix.gridlayout import GridLayout – способ формирования кнопок в форме матрицы, spacing задает размеры зазоров между кнопками, а cols задает количество элементов в матрице.
4. from kivy.uix.boxlayout import BoxLayout – способ формирования кнопок вдоль выбраной оси (либо вертикально, либо горизонтально).
5. from kivy.uix.label import Label – выводит передаваемый текст.
6. from kivy.uix.widget import Widget – метод использующий добавления новых виджетов в уже существующий.

Входе разработки решения с использованием роутера, было принято решение использовать фреймворк Selenium [6] позволяющий обращаться к web интерфейсам через их элементы. Для того что бы узнать необходимые элементы страницы достаточно открыть режим разработчика в браузере и посмотреть содержащиеся элементы и их принимаемые аргументы.

**3.2 Описание работа программы**

Main\_program.py

Class Screen1(Screen):

Класс в котором описаны методы для работы технологии Blockchain.

1. def \_\_init\_\_(self, \*\*kwargs): конструктор класса, реализующий появление кнопок, для взаимодействия с системой, а так же считывает бинарный файл, где храниться информация о текущей цепи у пользователя.
2. def lbl\_update(self): метод обновления отображаемого текста на экране.
3. def callback(self,value): метод передачи получаемого значения в имя пользователя (self.user\_name)
4. def clk(self,obj): функция вызова всплывающего окна для заполнения данных пользователя
5. def last\_block(self, instance): возвращает на экран последний вычисленный блок.
6. def full\_chain(self, instance): возвращает содержимое в цепочке на экран
7. def hash(self,block): находит хеш передаваемой информации
8. def add\_block(self,instance): добавляет новый вычисленный блок. Для добавления отметки необходимо получить proof от метода proof\_of\_work, который в свою очередь вернет информацию о физическом mac адресе устройства, с указанием имени пользователя.
9. def proof\_of\_work(self): Сначала метод получает логи роутера, при помощи метода Selenium\_req.wifi\_mac\_ip().lower(). Используется преобразование строки получаемой информации с маленьким шрифтом, для исключения возможных конфликтов поиска в списке всех mac адресов. Далее используется метод Selenium\_req.my\_mac(logi) для получения mac адреса текущего устройства. И так же используется метод self.valid\_proof(proof,logi), для проверки наличия полученного mac адреса устройства. В случае успеха сам метод возвращает proof, содержащий в себе mac адрес устройства.
10. def valid\_proof(self, mac,logi): использует Selenium\_req.valid\_mac (logi,mac) для определния наличия mac адреса в сети.
11. def valid\_chain(self, instance) :метод проверки правильности вычисленных хешей каждого блока цепи.
12. def finish\_add\_gen(self,instance): данная функция добавляет текущую цепочку в одну большую основную цепь, где будет храниться вся информация об этой цепи. Так же здесь реализован метод Merkl tree для вычисления основного хеша цепи, что в бедующем может пригодиться, с целью удаления устаревшей и ненужной информации.

class Screen3(Screen1): и class Screen4(Screen1):

Являются классами наследниками от Screen1 и реализуют тот же функционал что и базовый класс, только рассчитаны на использование для других предметов. Для этого у каждого класса имеются свои файлы для ведения записей по каждому из предметов.

class Screen5(Screen):

Класс для отображения общей цепи всех предметов, основной функцией которого является отображение последнего добавленного предмета или задаваемого по поиску.

1. def lbl\_update(self): метод обновления текста на экране.
2. def \_\_init\_\_(self, \*\*kwargs): конструктор экрана и элементов на нем
3. def callback(self,value): функция сохранения вводимого названия предмета для поиска.
4. def clk(self,obj): функция построения всплываемого окна.
5. def find\_block(self,instance): функция поиска предмета среди всех
6. def last\_block(self, instance): функция отображения последнего завершенного предмета.

class Screen2(Screen):

Пустой класс для реализации меню взаимодействия между предметами.

class ScreenManagement(ScreenManager):

Пустой класс для реализации взаимодействия между разными экранами.

class MyApp(App):

Класс реализующий запуск приложения с конструктором возвращающим Builder.load\_file("style.kv"), для построения стиля и вида приложения. Он же и является классом, запускающим работу приложения.

Selenium\_req.py

Файл реализующий методы взаимодействия с роутером, используя фреймворк selenium.

1. def wifi\_mac\_ip(): Метод реализующий подключение к роутеру c целью выявления списка подключенных устройств на данный момент. Selenium предоставляется возможности взаимодействия с компонентами настроек роутера, посредством web инструментов.
2. def my\_mac(text): Метод позволяющий найти физический mac адрес устройства среди все возможных подключений устройства. Получив все подключения, проводиться проверка нахождения mac адреса среди списка передаваемых устройств в text. Найденный mac возвращается в качестве физического адреса текущего устройства.
3. def valid\_mac(text, mac): функция проверки нахождения передаваемого mac адреса среди всего текущего списка устройств. Возвращает либо правду, либо ложь.

Style.kv

Описывает архитектуру приложения и взаимодействие экранов. Описание работы видимой части для пользователя выполняется в kivy как на подобие CSS, однако разметку видимых элементов необходимо проводить на прямую в самом файле. Прописываемые атрибуты, а также виджеты подобны их описанию при описании классов приложения.

Далее будут приведены файлы в качестве примеров работы взаимодействия двух пользователей. Каждый пользователь должен иметь свой порт на роутере для своего устройства.

Server.py

Выполняет роль сервера на указанном порту роутера и прослушивает все передаваемые сообщения от клиента.

mySocket.bind( (hostName, PORT\_NUMBER) )

Выстраиваем работу сервера.

mySocket.recvfrom(SIZE)

Прослушиваем передаваемые сообщения.

Client.py

Выполняет роль клиента для взаимодействия с сервером. Отправляет сообщения на указанный сервер.

mySocket.sendto(myMessage.encode('utf-8'),(SERVER\_IP,PORT\_NUMBER))

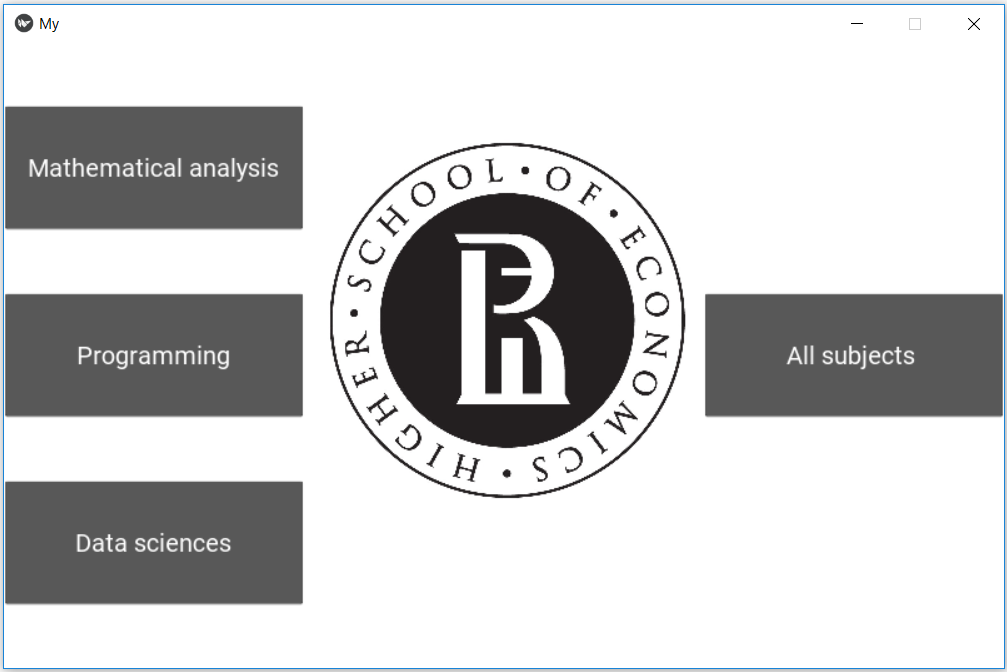
Основной метод передачи сообщения серверу.

**3.3 Тесты работы приложения**

Первым делом необходимо сформировать все необходимые файлы, записав в них изначальные структуры. Для записи будут использоваться бинарные файлы, где каждый файл будет соответствовать каждому предмету, в свою очередь файл, хранящий информацию о всех предметах будет сформирован отдельно. Для создания, чтения и работы с бинарными файлами будет использоваться библиотека pickle в силу удобства и простоты применения.

В качестве первого блока, каждой цепи, будет выступать блок genesis, хранящий в себе изначальную структуру каждого блока.

Начальное окно работы с приложением будет выглядеть следующим образом (Рис. 3):

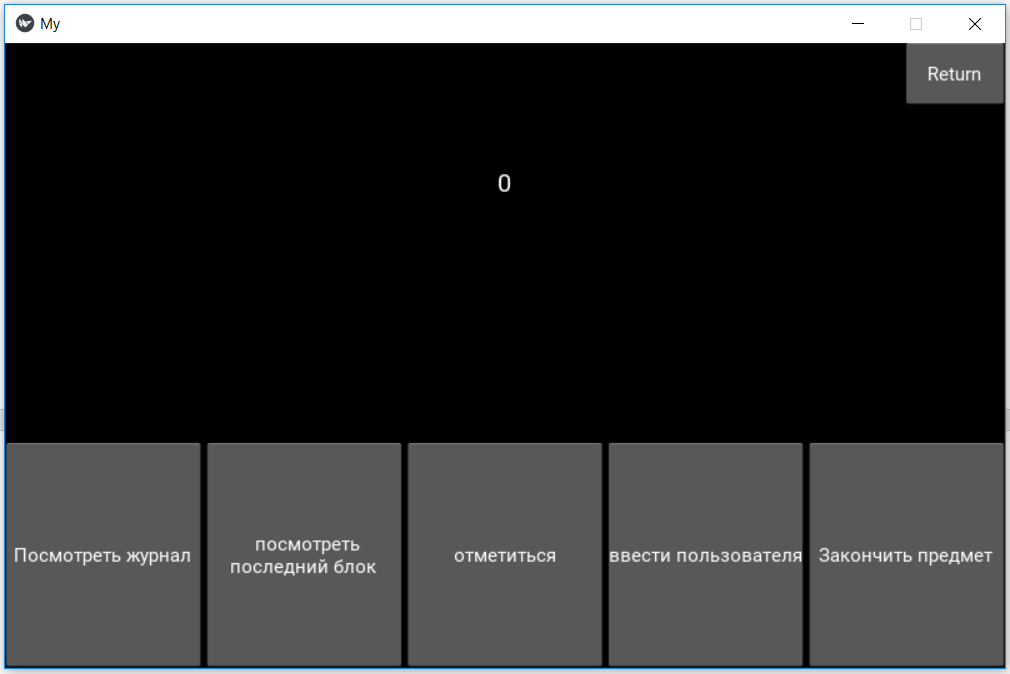


(Рис. 3) Меню приложения

Слева приведен список предметов. При необходимости добавления новых предметов, необходимо создать классы наследники и указать в файле style.kv расположение новых окон и объектов.

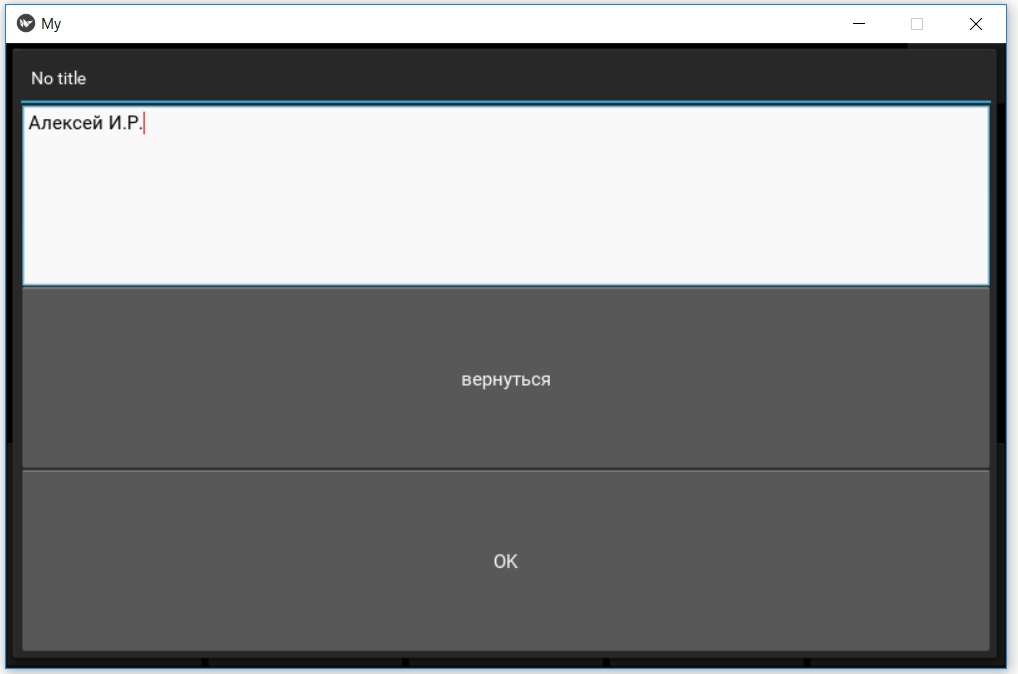
Справа кнопка All subjects переводит на окно для просмотра цепи всех завершенных предметов.

Выберем один из предметов, чтобы ознакомиться с интерфейсом и возможностями прописанных операций. Например, выберем предмет Mathematical analysis.



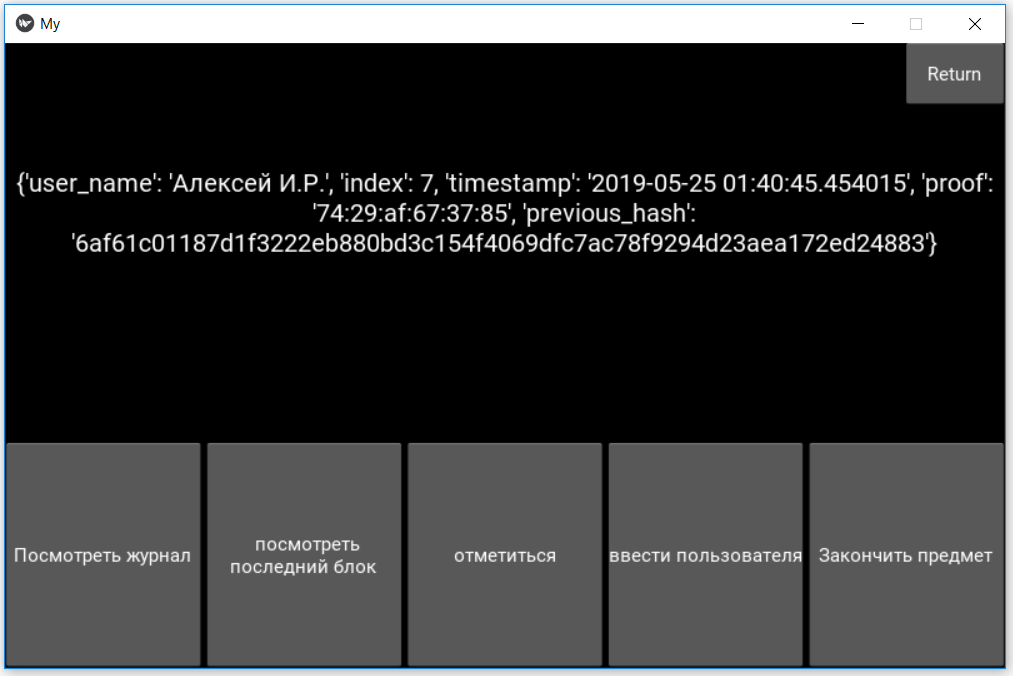
(Рис. 4) Экран предмета Mathematical analysis

Здесь присутствуют 6 кнопок (Рис. 4), где одна из них Return которая вернет пользователя на главный экран. Для того, чтобы отметиться на занятии, необходимо сначала нажать кнопку ввести пользователя. Далее появиться всплывающее окно (Рис. 5), где необходимо указать в свободном формате инициалы студента. После ввода данных необходимо нажать кнопку «ОК» и нажать кнопку вернуться, которое вернет пользователя обратно к предыдущему экрану. Далее необходимо нажать кнопку отметиться.



(Рис. 5) Всплывающее окно, для ввода имени пользователя

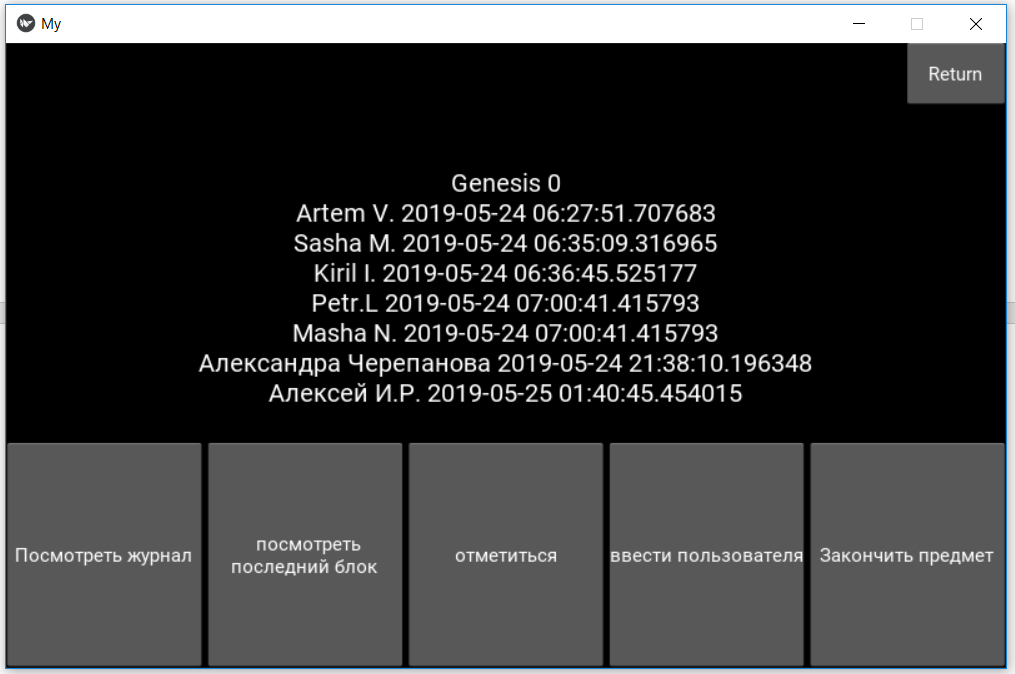
Что бы убедиться в том, что пользователь был добавлен, можно нажать кнопку «посмотреть последний блок». Нажав на нее, можно увидеть последнюю отметку в журнале Blockchain. Пример того, как выглядит блок информации изображен на рисунке (Рис. 6). Информация выдается так, как она записывается в сам Blockchain (это сделано с целью возможности просмотра детальной информации).



(Рис. 6) Вывод данных о последнем блоке

Здесь есть поле user\_name хранит имя последнего отмеченного пользователя, index, которое говорит о номере последнего блока, timestamp указывает время, в которое был добыт этот блок, proof это mac адрес устройства, которое отправляло сообщение или же mac адрес текущего пользователя и previous\_hash указывает вычисленный хеш предыдущего блока цепи.

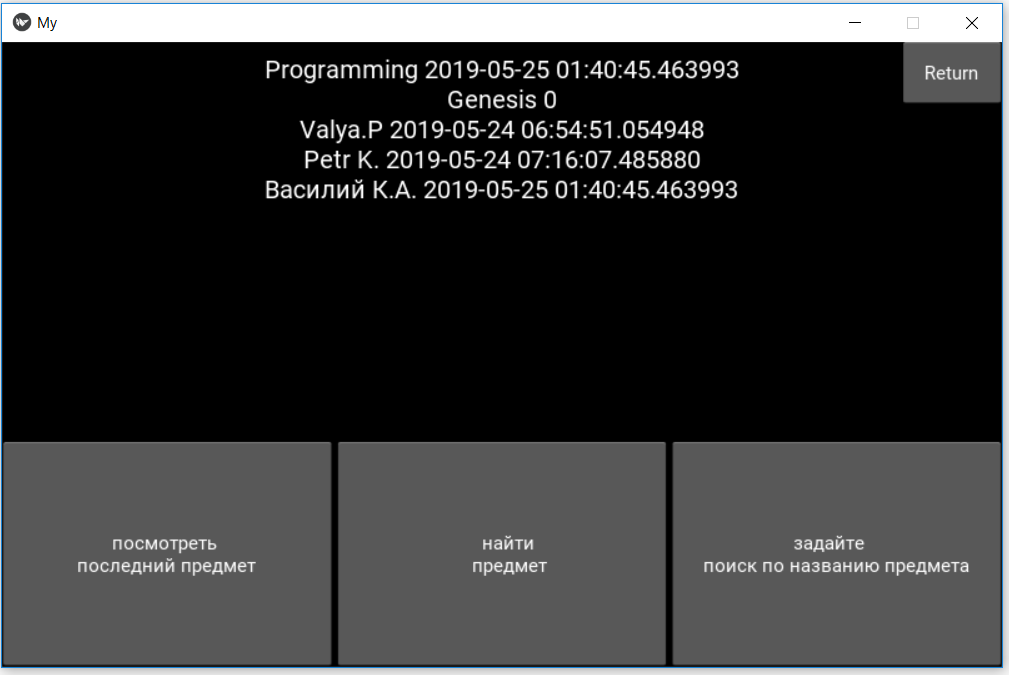
Нижу приведен результат нажатия кнопки «Посмотреть журнал» (Рис. 7). Здесь выводиться информация о всех отметках пользователей с указанием времени отметки.



(Рис. 7) Вызов кнопки посмотреть журнал

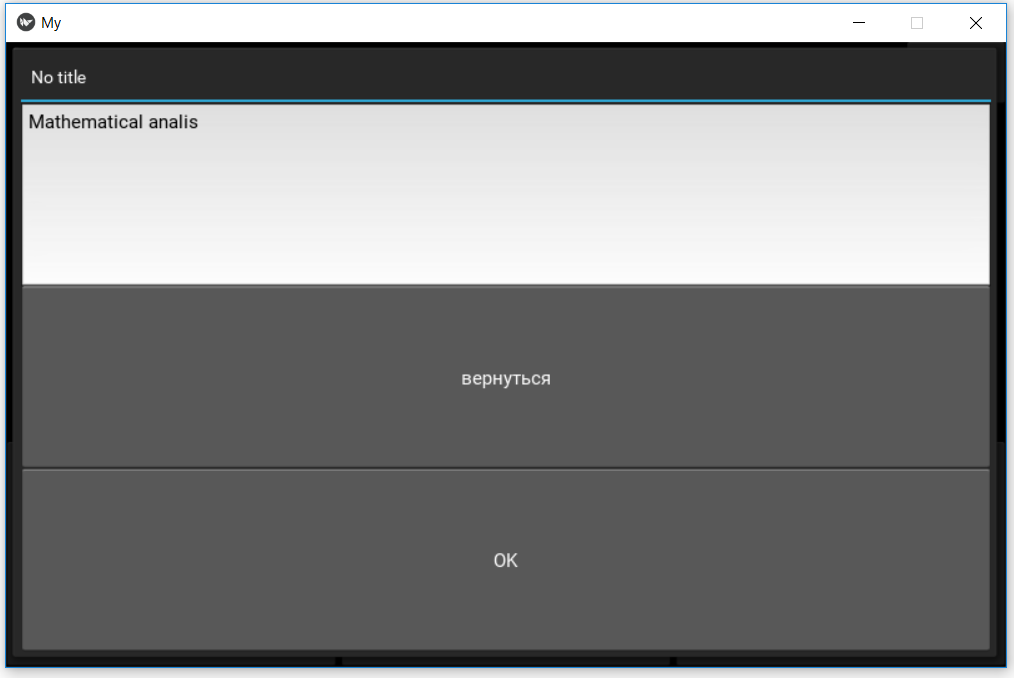
Кнопка «Закончить предмет» выполняет вызов функции finish\_add\_gen где выполняется формирование и добавление блока в общую цепь предметов.

Приведем пример того, как будет выглядеть одна из добавленных цепей предметов в основную. Для это сделаем вызов кнопки «добавить в главный блок» в каком-нибудь другом предмете и перейдем по кнопке All subjects из основного меню. В появившемся окне (Рис. 8) нажмем «посмотреть последний предмет». На экране появится информация о самом предмете и времени его добавления. Далее идут все отметки, которые было сделаны студентами за все время обучения.

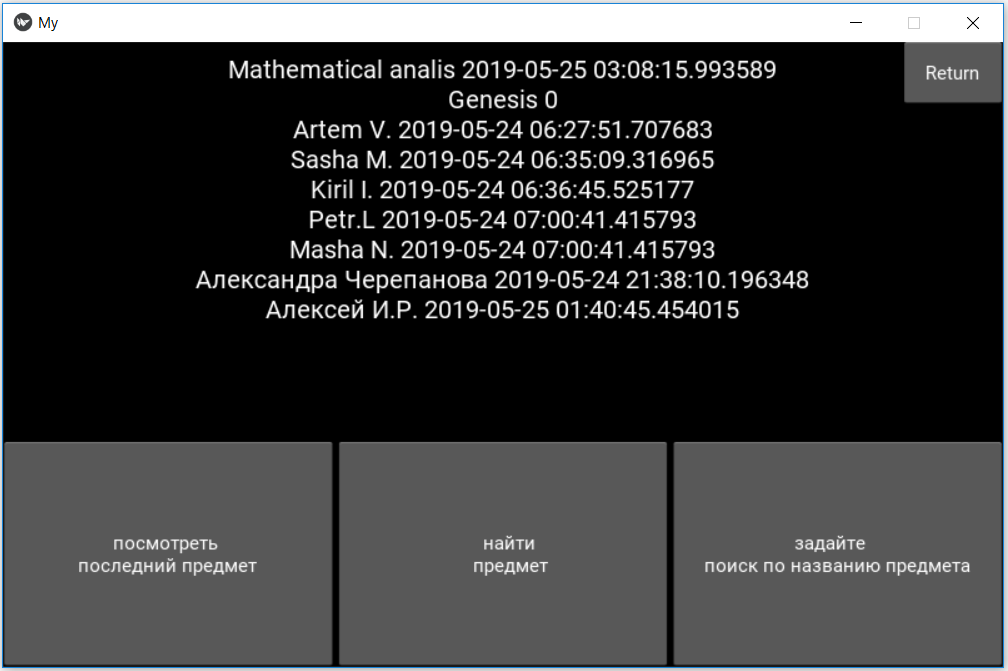


(Рис. 8) Экран всех предметов (All subjects)

При необходимости нахождения информации об уже законченном предмете, необходимо нажать кнопку «задайте поиск по названию предмета». В появившемся окне (Рис. 9) введите название необходимого предмета и нажмите кнопку «ОК».



(Рис. 9) Всплывающее окно ввода названия искомого предмета



(Рис. 10) Найденный предмет

После ввода необходимого предмета необходимо вернуться обратно и нажать кнопку «найти предмет». Искомый предмет и информации о нем отобразиться пользователю на экране (Рис. 10).