МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Web технологии»
Тема: Тетрис

Студент гр. 9382 _____ Иерусалимов Н.

Преподаватель _____

Санкт-Петербург
2021

Цель работы.

Целью работы является изучение работы web-сервера nginx со статическими файлами и создание клиентских JavaScript web-приложений.

Задание.

Необходимо создать web-приложение – игру в тетрис. Основные требования:
– сервер – nginx, протокол взаимодействия – HTTPS; – отображается страница для ввода имени пользователя с использованием HTML-элементов;
– статическая страница отображает «стакан» для тетриса с использованием HTML-элемента
используется для отображения следующей фигуры, отображается имя пользователя;
– фигуры в игре – классические фигуры тетриса (7 шт. тетрамино); – случайным образом генерируется фигура и начинает падать в «стакан» (описание правил см., например, https://ru.wikipedia.org/wiki/Тетрис);
– пользователь имеет возможность двигать фигуру влево и вправо, повернуть на 90□ и «уронить»;
– если собралась целая «строка», она должна исчезнуть;
– при наборе некоторого заданного числа очков увеличивается уровень, что заключается в увеличении скорости игры;
– пользователь проигрывает, когда стакан «заполняется», после чего ему отображается локальная таблица рекордов;
– вся логика приложения написана на JavaScript

Описание структуры проекта

Были созданы следующие директории и файлы в проекте рис(1).

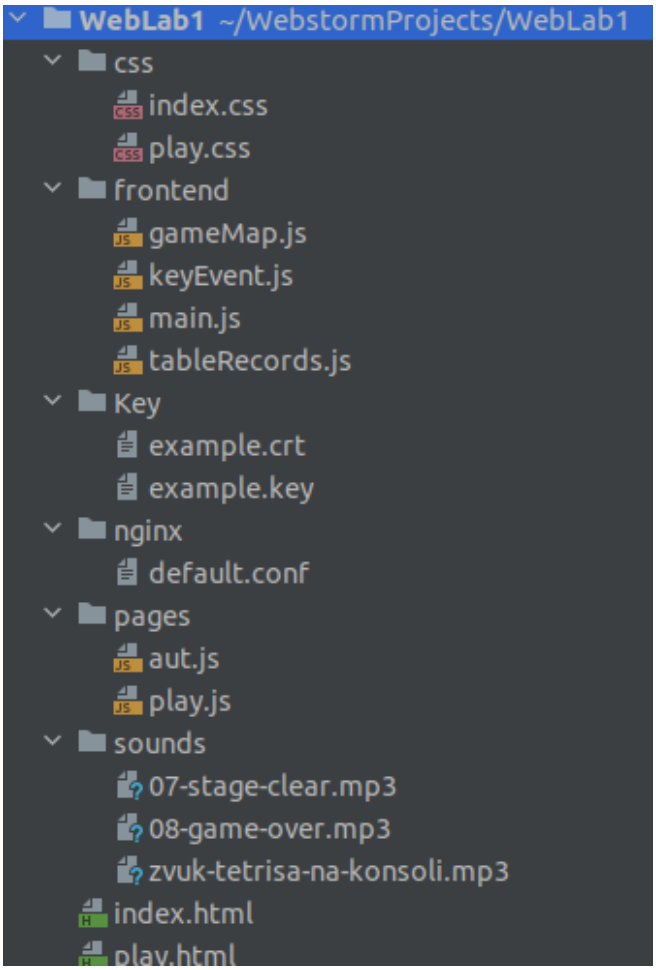

Рис.1<Файлы в проекте>

Описание директорий:

- css - хранит все css файлы для стилизации страниц.
- frontend - хранит .js файлы которые отвечают за логику игры.
    1. gameMap.js - в файле прописан класс GameMap который отвечает за взаимодействие с картой и ее отрисовкой, подробное описание методов класса прописаны ниже после описания директорий.
    2. keyEvent.js - файл отвечает за обработку нажатий клавиш.
    3. main.js - главный файл, в котором прописан gameLoop, описание фигур и нужные переменные.
    4. tableRecords.js - файл отвечает за создание, сортировки и отрисовки таблицы рекордов.

- Key & nginx - хранят ключи для https протокола и дефолтный конфиг локального сервера соответственно.
- pages - страницы сайта.
    1. aut.js - файл проверяет введено ли имя и если введено отправляет его в localstorage в виде объекта с параметрами "name:" и "score:". Далее если нажат кнопка переводит на страницу 'play'
    2. play.js - запускает игру.
- sounds - хранит все звуки которые используются в игре. Название звуков соответсвуют их применению.
- index.html - страница авторизации.
- play.html - страница игры.

Описание функций и методов в проекте:

- main.js
    - function gameLoop() - Главный цикл игры.
    - export function start() - Запускает игру.
    - function restartGame() - Перезапускает игру путем обновления страницы.
    - function gameOver() - Завершает игру, останавливает анимацию и выводит на экран "GAME OVER"
    - function randomFigGenerate() - Генерирует 7 случайных фигур и кладет их в очередь.
    - function fillNextFigCanvas(clear) - Обновляет экран, где показывается след. фигура.
    - function bestScroe() - Проверяет является ли текущее количество очков рекордом для текущего пользователя.

- function isFiledLine() - проверяет заполнена ли линия фигурами, если заполнена удаляет и смещает верхние фигуры на пиксель ниже.
- function fallingFigures() - Опускает фигуру на 1 пиксель ниже.

- gameMap.js
  - constructor(context, colors) - конструктор который принимает экран с которым идет работа и массив цветов индексы которых соответствуют номерам фигур. Создает матрицу 10x20 и заполняет ее нулями.
  - addFigure(figure) - Принимает текущую фигуру и добавляет ее на карту не отрисовывая.
  - draw(figure) - Принимает текущую фигуру, рисует ее не записывая ее на карту.
  - canPlace(currentFig) - Принимает текущую фигуру и проверяет коснулась ли она краев экрана или другой фигурыю.
  - refreshMap() - Полностью перерисовывает карту.
  - deleteOldFig(currentFig) - Удаляет прошлую фигуры с экрана. Не с главной карты.

Описание логики игры.

С самого начала пользователь вводит имя и нажимает на кнопку "subbmit" если имя не было введено сайт выдаст предупреждение - "вы не ввели имя пользователя", если имя есть оно заносится в localstorage как объект с параметром name и score. Score с самого начала у каждого пользователя равен 0. После чего идет переадресация на страницу "../play" с помощью window.oopen('play') далее nginx слушает что было добавлено /play к главному url и переходит к файлу "play.html".

В файле play.html загружается вся внешняя составляющая и подключается скрипт play.js, в нем вызывается функция start() из файла main.js.

В функции start() генерируется случайная очередь из фигур с помощью randomFigGenerate() последняя фигура из очереди определяется как текущая далее запускается показатель след. фигуры с помощью fillNextFigCanvas() и запускается главный цикл игры.

В главном цикле запускается requestAnimationFrame(gameLoop) эта функция принимает в себя функцию callback (т.е ту функцию, которую она будет вызывать) в нашем случае это gameLoop. requestAnimationFrame это метод из js который на каждом кадре вызывает переданную ему функцию. Далее увеличиваем счетчик кадров при каждом вызове на 1, если счетчик меньше 60 ничего не делаем, если больше обнуляем счетчик проверяем можем ли поставить фигуры на текущих координатах canPlace(currentFig) если можем, удаляем с экрана старую фигуру и увеличиваем координаты по "y" у текущей фигуры, отрисовываем с помощью метода draw(), удаляем с помощью метода deleteOldFig(currentFig). Если же поставить фигуру нельзя тогда мы добавляем ее на карту проверяем заполнена ли линия sFiledLine(), обновляем карту, проверяем не коснулась ли фигура верхней границы если коснулась вызываем gameOver() если нет обновляем текущую фигуру беря следующую из очереди, если очередь пуста вызываем генератор очереди и потом обновляем экран с показателем следующей фигуры.

Обработка нажатий на клавиши. Был добавлен слушатель который при нажатии на любую клавишу вызывал стрелочную функцию аргумент у которой является текущая нажатая клавиша. С помощью switch case определяется что за клавиша была нажата и в зависимости от этого меняются координаты у текущей фигуры. Если были нажаты клавиши влево вправо тогда вызывалась функция keyPresed(x,y,keyUp) ее аргумен показывают смещение фигуры по координатам, keyUp булевый аргумент показывающий была ли нажата кнопка up. Если фигуру возможно сместить влево или вправо

к кординатам плюсуются переданные и рисуется новая фигура. Если Была нажата клавиша вверх вызывается метод keyPresed(0,0,true) - что говорит не двигай фигуру поверни ее, поворот реализован с помощь транспонирования матрицы текущей фигуры если фигуру можно повернуть, тогда рисуется фигура и задается новая матрица для конкретной фигуры.

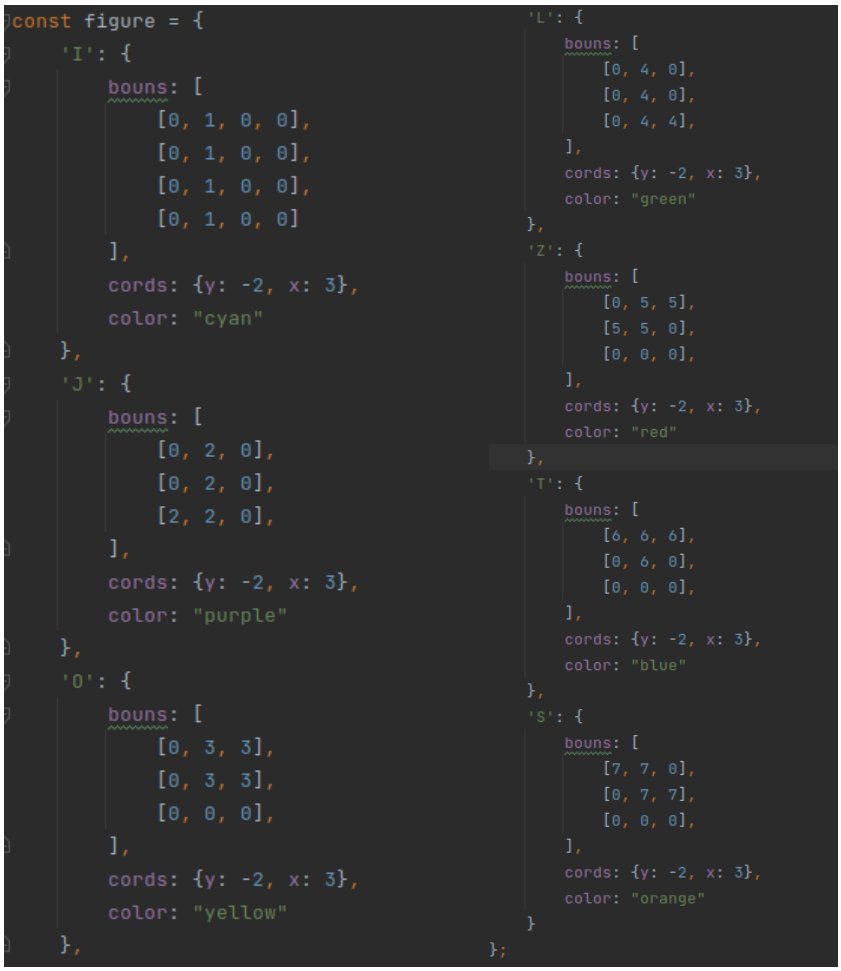

Рис.2<fig - описание фигур>

Тестирование.

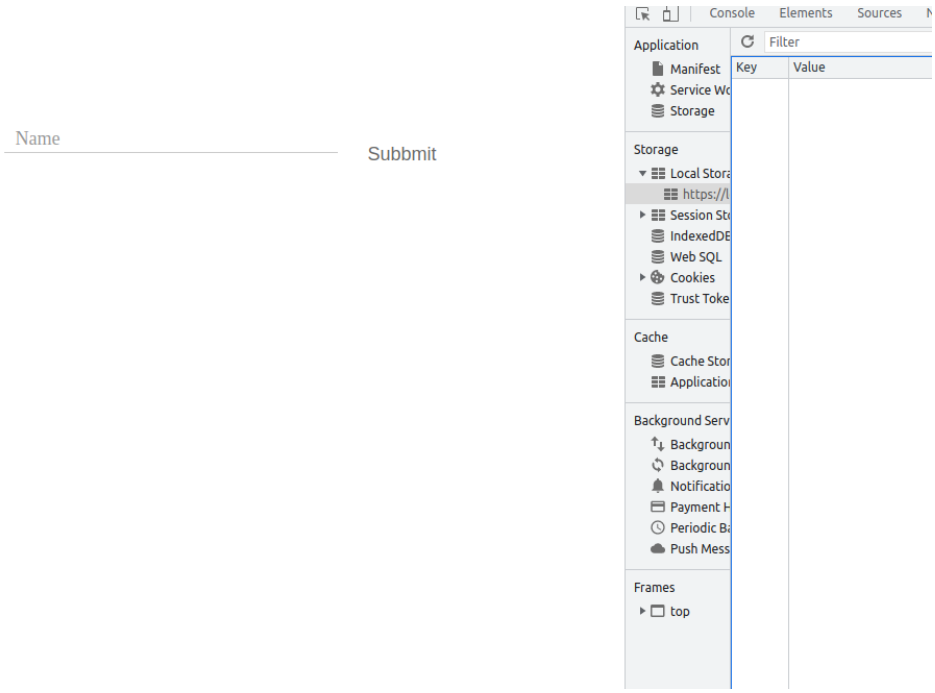

Рис.3<Экран авторизации>

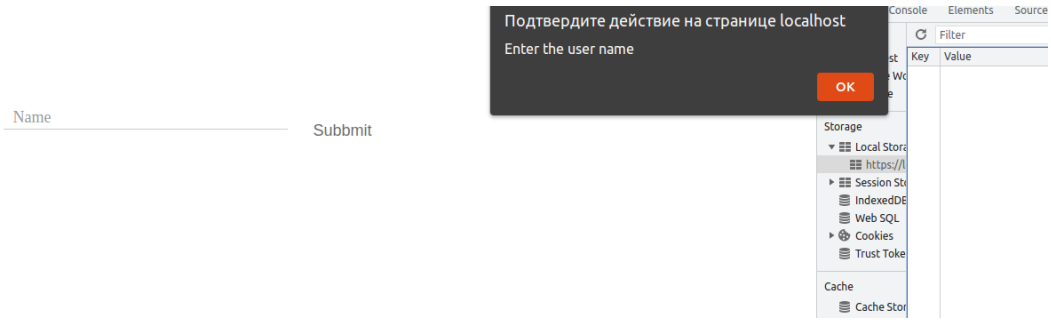

Рис.4<Вход без имени>

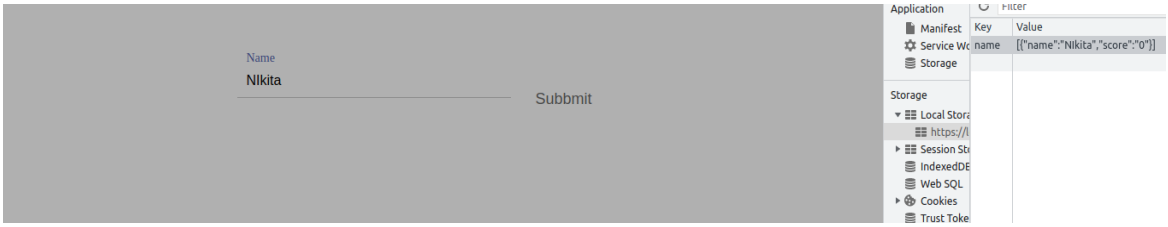

Рис.5<Данные занесены localstorage>

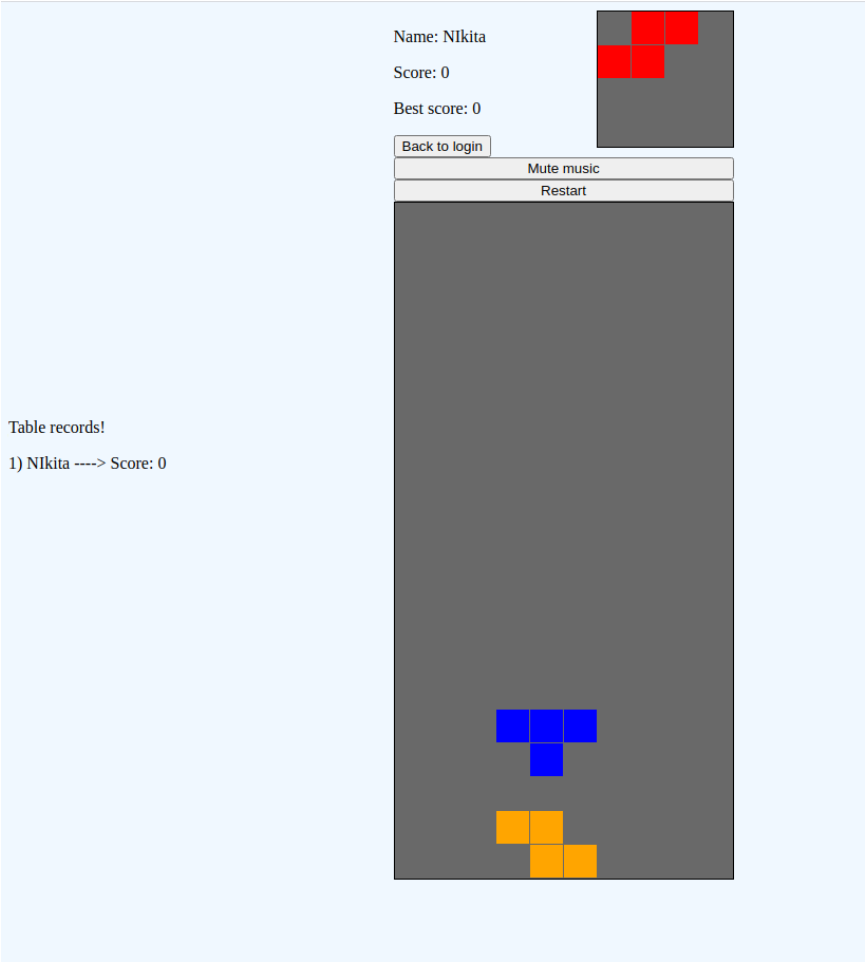

Рис.6<Начало игры>

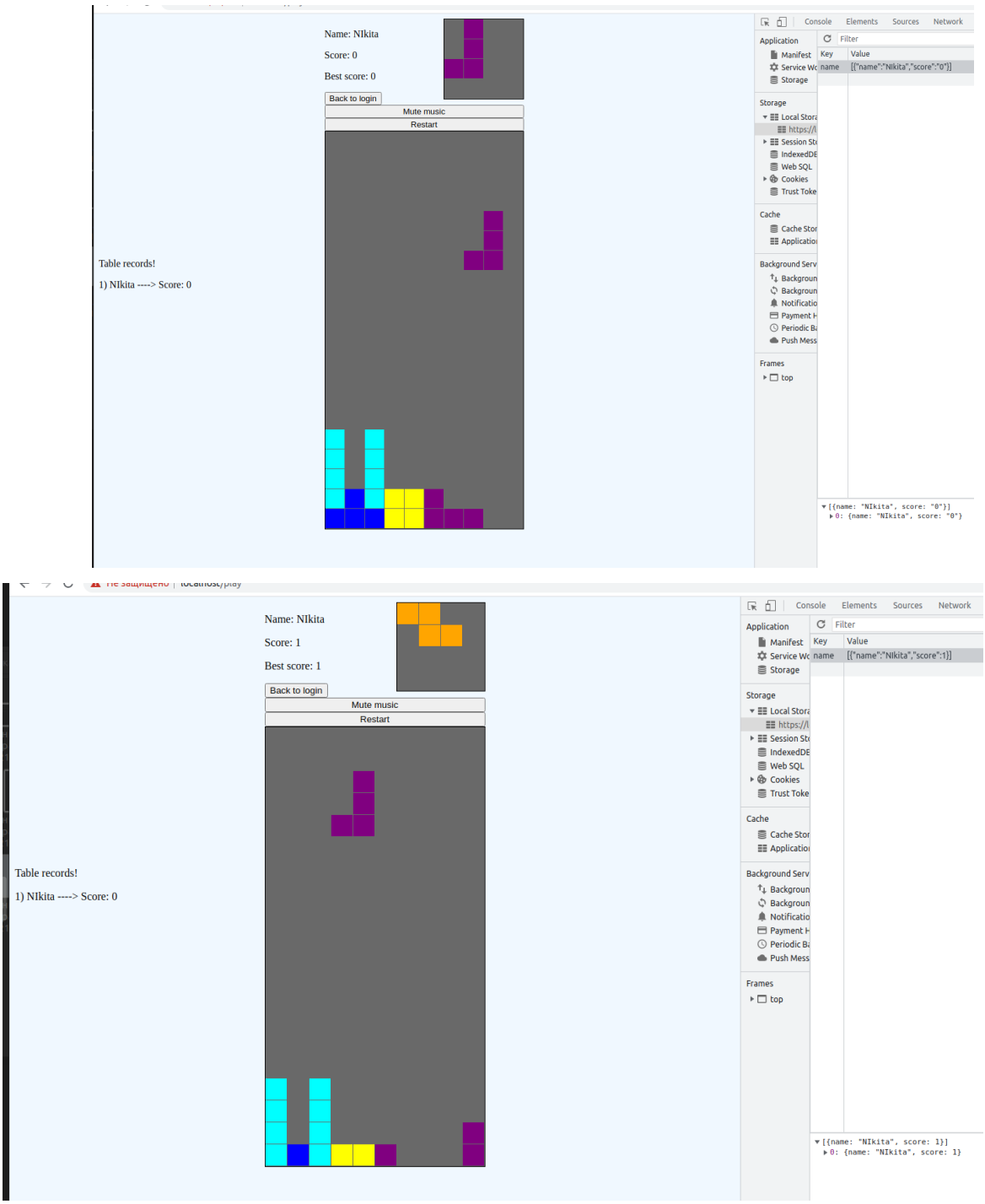

Рис. 7 <Очистка линии>

Рис. 8 <Конец игры>

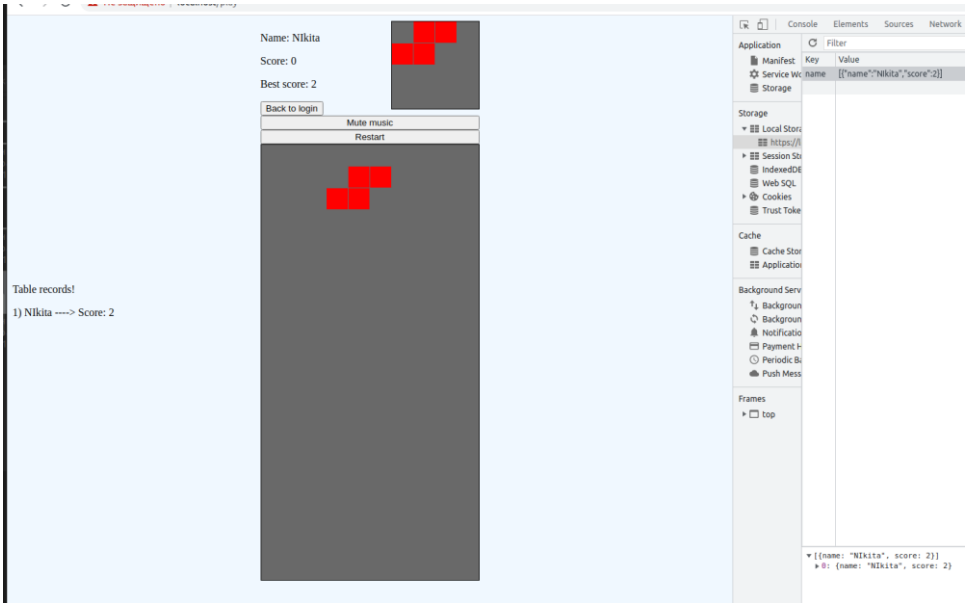

Рис. 9<Обновление таблицы рекордов>

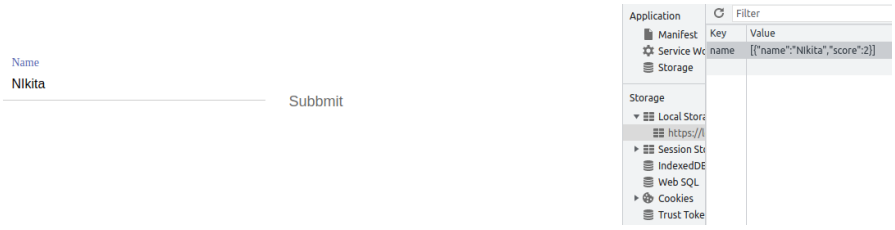

Рис. 10 <Нажатие кнопки Back to login>

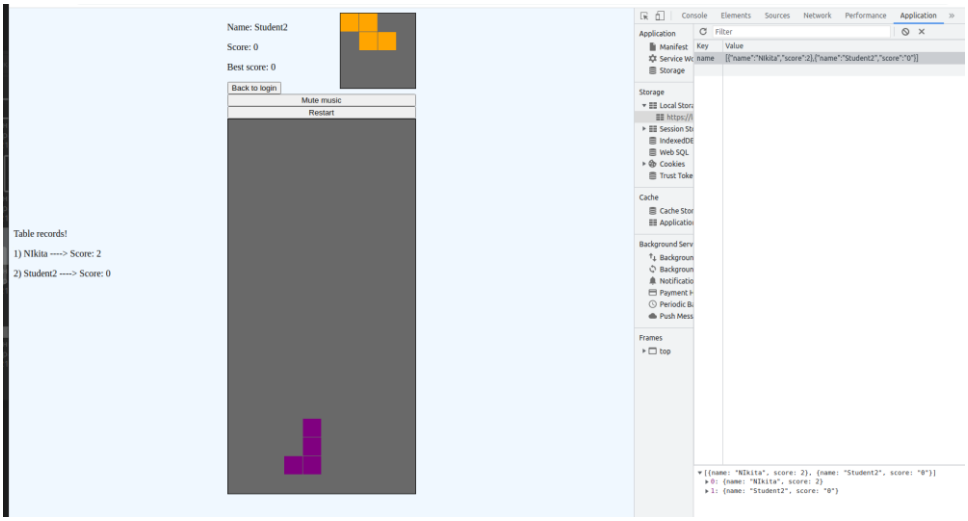

Рис. 11 <Создание нового пользователя>

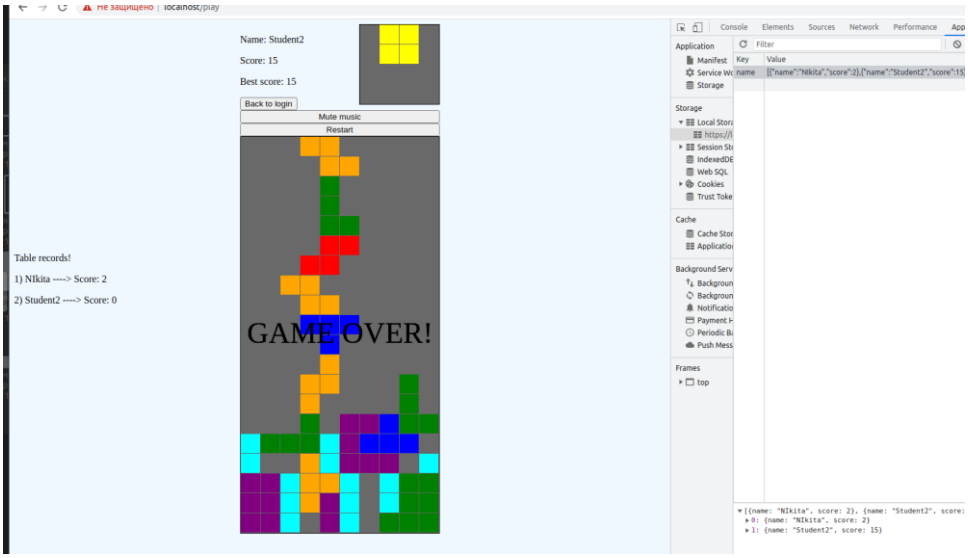

Рис. 11 <Новый рекорд>

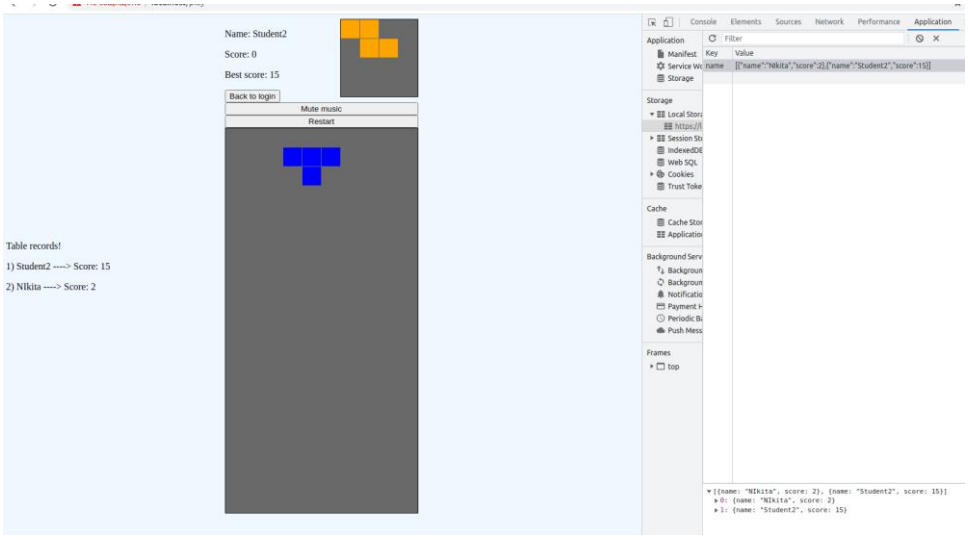

Рис. 11 <Проверка таблицы рекордов>

Выводы.

В процессе выполнения работы была изучена работа web-сервера nginx со статическими файлами и были созданы клиентские JavaScript web-приложений. Познакомились с языком JavaScript изучили интересные инструменты на подобие localstorage, timer, requestAnimation, context и т.д