МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ЯДЕРНЫЙ УНИВЕРСИТЕТ «МИФИ»

Лабораторная работа на тему:
Программная реализация проекта
«Веб-сервис: игра крестики-нолики»

Выполнил студент:
Красников Кирилл Денисович
Группа: С24-702
Проверил:
Курасов Юрий Викторович

Оценка: _____
Дата: _____
Подпись: _____

Москва, 2025

К.Д. Красников, С24-702

Тема: Программная реализация проекта «Веб-сервис: игра крестики-нолики»

Инструменты разработки: компилятор языка C (GCC) и универсальный отладчик GNU Debugger (GDB) для ОС Linux Debian 12.

1. Задача

Реализовать веб-сервис, предоставляющий пользователям доступ к сетевой игре «Крестики-нолики». Клиент данной игры следует реализовать на языке JavaScript, используя возможности HTML5. Коммуникация между клиентом и сервером должна осуществляться через протокол HTTP и JSON. Сервер должен поддерживать неограниченное количество клиентских и игровых сессий.

2. Теоретическая часть

Для реализации веб-сервиса требуется параллельно обрабатывать несколько TCP соединений. Для этого можно использовать системный вызов `poll` ядра Linux.

Если отправлять от клиента на сервер данные в параметрах GET запроса, значительно упростится реализация сервера, так как, во-первых, нет необходимости в поддержке других методов HTTP, а во-вторых — чтение URL-параметров проще в реализации, чем чтение JSON. Для оптимизации и упрощения очистки памяти при чтении URL-параметров можно реализовать линейный аллокатор (арену).

Для отправки данных от сервера к клиенту следует использовать режим HTTP chunked transfer encoding, так как при его использовании не требуется знать заранее размер отправляемого сообщения. В таком случае мы можем использовать потоковую генерацию JSON. Чтобы минимизировать количество системных вызовов, можно использовать буферизацию.

С целью экономии памяти следует очищать сессии, которые были неактивны более пяти минут.

Для вывода на экран игрового поля в клиентской части можно воспользоваться Canvas API из спецификации HTML5.

3. Практическая часть

Было принято решение не использовать многопоточность. В структуре `http_server_t` хранятся дескриптор сокета сервера, структуры `pollfd` для всех подключенных в данный момент клиентов и их количество, а также всё, что нужно для обработки текущего соединения (дескриптор сокета текущего клиента, буфер, арена, путь, связный список пар ключ-значение URL-параметров). Память под список URL-параметров выделяется на арене.

Для упрощения отладки проекта был реализован макрос `log_message(LOG_LEVEL_INFO | LOG_LEVEL_ERROR, "module_name", "text", ...)`.

Были реализованы структура `json_writer_t` и набор функций: `init_json_writer`, `json_start_dict`, `json_stop_dict`, `json_start_array` и др., позволяющие пользователю использовать потоковую генерацию JSON. Для буферизации была создана структура `buffered_writer_t` и набор функций: `init_buffered_writer`, `buffered_writer_flush`, `buffered_writer_write` и `buffered_writer_end`. Пример использования `json_writer_t` представлен в приложении А.

Информация о клиентских и игровых сессиях хранится в структуре `session_manager_t` в виде связных списков. Структура `session_t` содержит сессионный ключ, имя пользователя, ссылку на игровую сессию, время последней активности пользователя, флаг, указывающий, что сессию можно удалить, и ссылку на следующую в списке сессию. В структуре `game_session_t` содержатся ссылки на игроков, вид игры (архитектура приложения предусматривает возможность добавления нескольких игровых режимов), ссылку на состояние игры (`tictactoe_game_t`), флаги, описывающие состояние игровой сессии, и ссылку на следующую в списке игровую сессию. Изменения состояний структуры `game_session_t` показаны на диаграмме в приложении В.

Маршрутизация происходит в функции `main`, обрабатывают запросы функции из файла `http_handlers.c`. Для идентификации пользовательской сессии используется сессионный ключ, который передаётся как URL-параметр `id`. Для получения доступа к ресурсам,

за исключением статических файлов и станицы авторизации, требуется наличие этого параметра. На диаграмме в приложении С показано, как обрабатываются запросы до момента подключения к игровой сессии.

Пример запроса клиента во время игры выглядит следующим образом: `GET /gameRequest?id=123&cell=2`, где `id` — это сессионный ключ, а опциональный параметр `cell` — клетка, на которую делает ход клиент-отправитель. Пример ответа сервера находится в приложении D.

Клиентская часть написана на JavaScript без использования сторонних библиотек. Вывод изображения на экран осуществляется с помощью Canvas API в режиме `CanvasRenderingContext2D`. Скриншоты интерфейса находятся в приложении E.

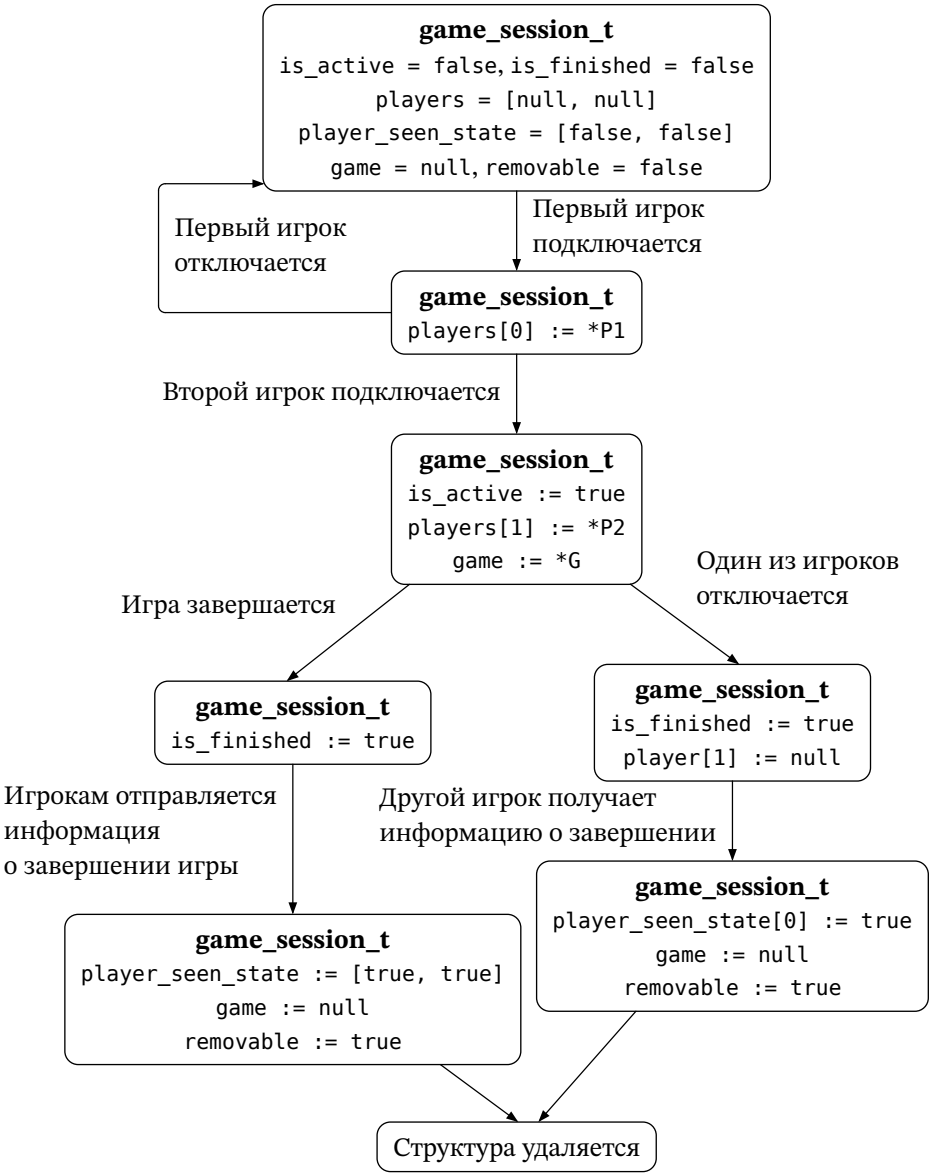
4. Содержание

5. Приложения

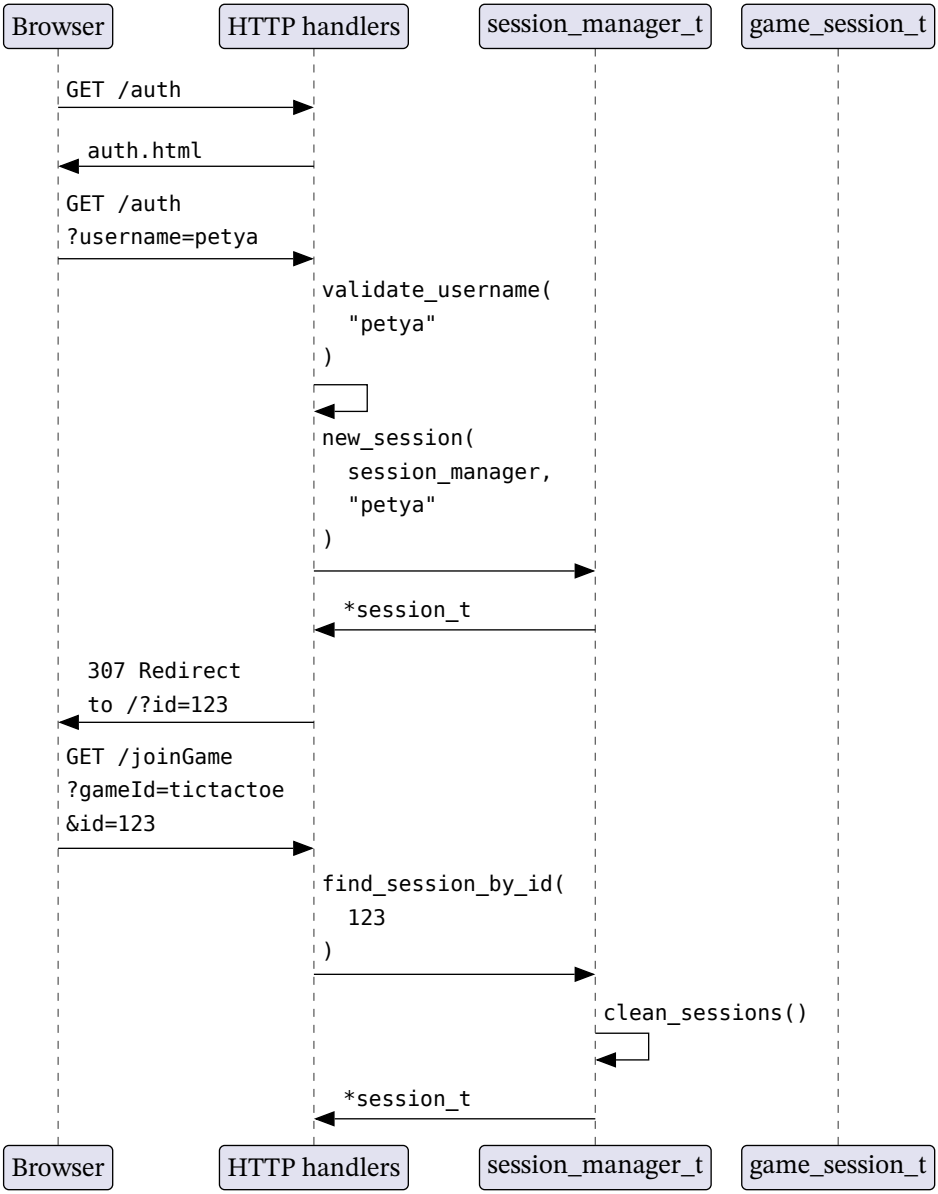
Приложение А

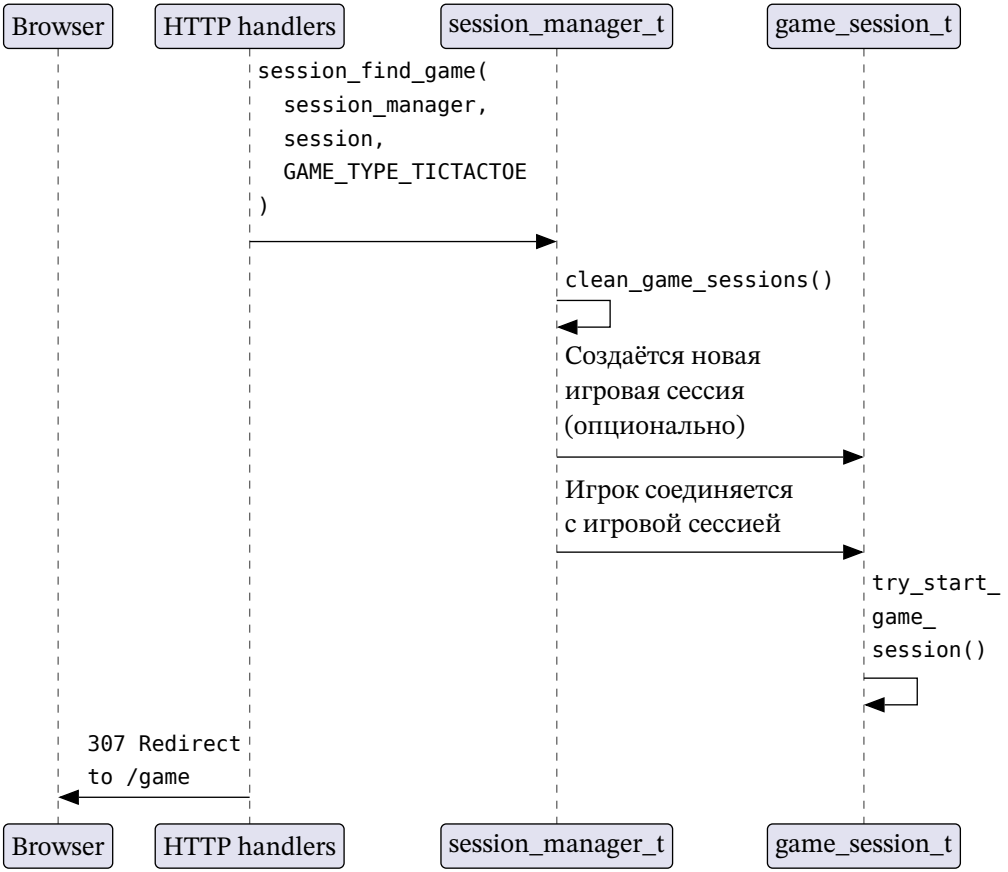
```
void handle_session_list_request(
  session_manager_t *session_manager,
  json_writer_t *json_writer) {
  clean_sessions(session_manager);
  json_start(json_writer);
  json_start_array(json_writer);
  session_t *session = session_manager->sessions;
  while(session != NULL) {
    if (!session->is_active) {
      continue;
    }
    json_start_dict(json_writer);
    json_write_key(json_writer, "id");
    json_write_number(json_writer, session->id);
    json_write_key(json_writer, "name");
    json_write_string(json_writer, session->username);
    json_write_key(json_writer, "player_index");
    json_write_number(json_writer, session->player_index);
    json_write_key(json_writer, "game_session");
    if (session->game_session == NULL) {
      json_write_null(json_writer);
    } else {
      json_start_dict(json_writer);
      json_write_key(json_writer, "is_active");
      json_write_bool(
        json_writer, session->game_session->is_active
      );
      json_write_key(json_writer, "game_type");
      json_write_number(
        json_writer, session->game_session->game_type
      );
      json_stop_dict(json_writer);
    }
    json_stop_dict(json_writer);
    session = session->next_session;
  }
  json_stop_array(json_writer);
  json_end(json_writer);
}
```

Приложение В



Приложение С




Browser
HTTP handlers
session_manager_t
game_session_t
session_find_game(
session_manager,
session,
GAME_TYPE_TICTACTOE
)
clean_game_sessions()
Создаётся новая
игровая сессия
(опционально)
Игрок соединяется
с игровой сессией
try_start_
game_
session()
307 Redirect
to /game
Browser
HTTP handlers
session_manager_t
game_session_t

Приложение D


```
{
  "status": "game_active",
  "your_index": 0.000000,
  "player_names": ["kirill", "fedya"],
  "state": {
    "active_player_index": 1.000000,
    "winner_index": -1.000000,
    "cells": ["X", " ", " ", " ", " ", " ", " ", " ", " "]
  }
}
```


Приложение E

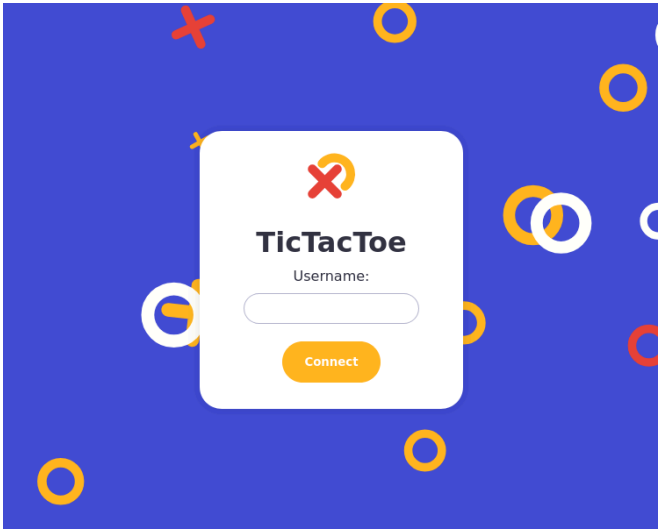


Рис. 1. Страница авторизации

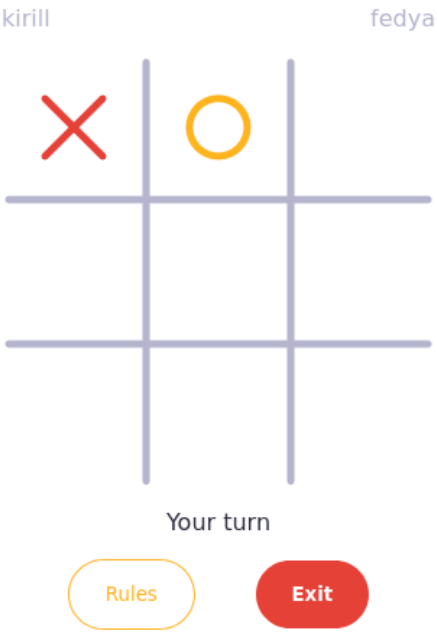


Рис. 2. Страница с игрой