МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**

Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА WEB-ПРИЛОЖЕНИЯ «ТРЕНАЖЕР ПАМЯТИ»
АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Прянишниковой Екатерины Викторовны

Научный руководитель

к. ф.-м. н., доцент

Ю. Н. Кондратова

Заведующий кафедрой

к. ф.-м. н., доцент

С. В. Миронов

ВВЕДЕНИЕ. Память – форма психического отражения, заключающаяся в закреплении, сохранении и последующем воспроизведении прошлого опыта, делающая возможным его повторное использование в деятельности или возвращение в сферу сознания. Память связывает прошлое субъекта с его настоящим и будущим и является важнейшей познавательной функцией, лежащей в основе развития и обучения.

Значение памяти в жизни человека очень велико. Абсолютно все, что мы знаем, умеем, есть следствие способности мозга запоминать и сохранять в памяти образы, мысли, пережитые чувства, движения и их системы. Память создает, сохраняет и обогащает наши знания, умения, навыки, без чего немислимы ни успешное учение, ни плодотворная деятельность. Представления памяти являются воспроизведением, более или менее точным, предметов или явлений, когда-то действовавших на наши органы чувств. Актуальность работы обусловлена возрастанием количества пользователей, заинтересованных в развитии памяти.

Материал для этой работы собирался за время прохождения преддипломной практики.

Целью работы является разработка web-приложения для тренировки памяти. Для достижения поставленной цели можно выделить следующие задачи:

- изучить теоретические основы развития памяти;
- изучить Node.js, Express.js, MS SQL Server, Typescript и другие технологии при разработке web-приложения;
- создать web-приложение для тренировки памяти, состоящее из клиентской и серверной частей, взаимодействующих через произвольный API;
- описать работу данного приложения.

Работа выполнена на 77 страницах машинописного текста, состоит из введения, 3 глав, заключения, содержит 16 рисунков, 10 приложений, список литературных источников содержит 20 наименований.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ. Проведем анализ существующих решений. Титулованный немецкий проект NeuroNation предлагает тренировать память, концентрацию и интеллект. После регистрации пользователь должен пройти тестирование, которое выявит его способности в нескольких сферах умственной деятельности. Затем ему предложат выбрать навыки, которые он хочет потренировать. Однако в бесплатной версии будет доступна только

часть мини-игр. Peak работает по тому же принципу, что и NeuroNation. Это набор упражнений на память, быстроту мышления, знание языка, внимательность, эмоциональность и навыки решения проблем. Чем дольше вы играете, тем выше сложность тренировок. Проект представлен только в виде мобильного приложения, и, судя по отзывам, работает не на всех смартфонах и не всегда стабильно. По данным создателей, регулярные занятия на тренажёрах «Викиум» позволяют улучшить скорость реакции на 17% за неделю. Память – на 19% за 2-3 недели. Проект представлен в виде веб-приложения и мобильного приложения. Тренажер платный. Сервис BrainApps составляет индивидуальную программу для тренировок по результатам первичного теста. На сайте собрано более 90 игр, в которых нужно разгадывать анаграммы, запоминать положение закрашенных клеток, находить пару предмету и так далее. Но не все из них доступны в бесплатной версии. Сайт и приложение Lumosity не русифицированы, но для тренировок школьного уровня английского языка его будет достаточно. Принцип действия сервиса всё тот же: с помощью головоломок вы развиваете память, внимание, скорость, гибкость ума и навык решения проблем. Сервис BrainExer 2.0 выглядит лаконично и просто, но при этом он довольно функционален. В отличие от предыдущих сервисов, в BrainExer 2.0 сложно будет сразу.

Основные недостатки аналогов – ограниченность бесплатных функций, нестабильная работа, отсутствие поддержки русского языка, веб-версии. Кроме того, данные приложения никак не учитывают возраст пользователя при подборе программы тренировок. Приложение, которое будет создано в рамках работы, должно исправить эти недостатки и адаптировать программу тренировок под возраст пользователя.

Для реализации серверной части приложения планируется использовать Node.js, Express.js, MS SQL Server, а для реализации клиентской части – Typescript, не считая различных вспомогательных библиотек. Данные технологии были выбраны, так как позволяют писать серверную и клиентскую части на одном языке (javascript), использовать статическую типизацию на клиенте, где находится основная бизнес-логика (typescript), работать с популярными и зарекомендовавшими себя технологиями.

Node.js - это серверная JavaScript-платформа, предназначенная для создания масштабируемых распределенных сетевых приложений, использующая событийно-ориентированную архитектуру и неблокирующее асинхронное вза-

имодействие. Она основана на JavaScript-движке V8 и использует этот же JavaScript для создания приложений.

Express - это минималистичный и гибкий веб-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений. Имея в своем распоряжении множество служебных методов HTTP и промежуточных обработчиков, создать надежный API можно быстро и легко.

SQL Server является одной из наиболее популярных систем управления базами данных (СУБД) в мире. Данная СУБД подходит для самых различных проектов: от небольших приложений до больших высоконагруженных проектов.

Typescript – язык программирования, компилирующийся в JavaScript, который уже практически стал стандартом веб-разработки. Учитывая, что JavaScript имеет динамическую типизацию, что часто неудобно при работе над крупными проектами, в большинстве фронтенд-проектов используется Typescript, предоставляющий строгую статическую типизацию. В проектах на Angular он используется по умолчанию, в React-проектах довольно часто, особенно для крупных приложений с большим количеством логики.

TypeScript добавляет в JavaScript дополнительный синтаксис для поддержки более тесной интеграции с редактором. Отметим, что VS Code отлично поддерживает Typescript (после установки соответствующих плагинов).

В качестве менеджера пакетов рассмотрим npm, входящий в состав Node.js. В Node.js npm уже имеется по умолчанию, поэтому его отдельная установка не требуется. Все пакеты хранятся в облачной репозитории на официальном сайте. Node.js npm также избавляет разработчика от необходимости хранить все используемые сторонние модули в удаленных репозиториях. За это отвечает файл package.json.

В npm файл package.json используется для описания модулей, от которых зависимо ваше приложение, и содержит ряд метаданных о приложении. Модули – это приложения, которые реализовывают некоторый функционал и могут быть использованы для разработки более сложных приложений. Например, фреймворк Express является устанавливаемым модулем.

В package-lock.json содержится описание состояния текущей иерархии используемых в приложении модулей: их точные версии и точные версии используемых ими модулей и так до конца. При развертывании Node.js приложения в новой среде наличие файла package-lock.json гарантирует, что будут установ-

лены версии тех пакетов, которые использовались в разработке в последний раз.

Вспомогательные модули, используемые в проекте:

- bodyParser;
- tedious;
- events;
- cors.

BodyParser – библиотека, позволяющая приложению express.js читать тело (то есть, содержание) входящих запросов.

Модуль tedious — это реализация протокола TDS на языке JavaScript, который поддерживают все современные версии SQL Server.

Модуль Cors необходим для установки cors. Cross-origin resource sharing (совместное использование ресурсов между разными источниками) — технология современных браузеров, которая позволяет предоставить веб-страницам доступ к ресурсам другого домена.

подавляющее большинство функционала Node.js применяет асинхронную событийную архитектуру, которая использует специальные объекты – эмиттеры для генерации различных событий, которые обрабатываются специальными функциями – обработчиками или слушателями событий. Все объекты, которые генерируют события, представляют экземпляры класса EventEmitter. Весь необходимый функционал сосредоточен в модуле events.

Модуль body-parser необходим для корректной обработки передаваемых в теле данных.

Способы развития памяти будем изучать на основе на курсе лекций по психологии памяти Боднара Александра Михайловича и работы профессора Ричарда Аткинсона «Человеческая память и процесс обучения» по управлению кратковременной памятью и перезаписи информации в память долговременную.

Основная идея, проходящая через все работы Аткинсона, состоит в том, чтобы подчеркнуть тесную взаимосвязь теории и экспериментирования. Всякий раз, когда это представляется возможным, теория строится в терминах формального языка — либо как математическая модель, либо как машинная программа; затем из нее выводятся следствия (предсказания), которые служат основой для построения соответствующих экспериментов.

Развитие математической психологии и разработка моделей поведения с

помощью ЭВМ усилили интерес к двухкомпонентной концепции памяти, и сейчас она является объектом многочисленных теоретических и экспериментальных исследований. Особенно важное значение придается системе кратковременной памяти (КВП), или кратковременному хранилищу (КВХ). Это связано с тем, что процессы, протекающие в кратковременном хранилище, находятся под непосредственным контролем индивида и управляют потоком информации в системе памяти; они могут вводиться в действие по усмотрению индивида, оказывая очень большое влияние на его деятельность.

Некоторые процессы управления используются всеми и во многих ситуациях, другие применяются только в особых обстоятельствах.

«Повторение» представляет собой скрытое (про себя) или открытое (вслух) неоднократное воспроизведение информации, например: при запоминании номера телефона, с тем чтобы записать его, или при запоминании имен людей, которым вы были только что представлены, или при переписывании отрывка из книги.

«Кодирование» относится к процессам управления такого типа, когда подлежащая запоминанию информация вводится в контекст дополнительной, легко извлекаемой информации – таковой может быть мнемоническая фраза или предложение.

«Представление» — это процесс управления, в результате которого вербальная информация запоминается с помощью зрительных образов. Цицерон, например, для того, чтобы лучше запомнить длинный список (или речь), рекомендовал мысленно пройти через хорошо знакомое здание, последовательно размещая элементы списка в каждом из его помещений.

Концепции Аткинсона довольно любопытны, однако он не учитывает зависимость их эффективности от возраста испытуемых. Рассмотрим еще одно исследование запоминания и памяти, учитывающее возраст испытуемых. Паралелограмм развития представляет собой графическое отображение результатов экспериментальных исследований одного из основных свойств высших психологических функций — опосредствованности. Эти исследования были проведены А. Н. Леонтьевым с опорой на идеи Л. С. Выготского о двух родах стимулов (стимулах-объектах и стимулах-средствах) и с применением созданного Выготским и его сотрудниками метода двойной стимуляции (исследование осуществляется с помощью двух рядов стимулов: один ряд стимулов выполняет функ-

цию объекта, на который направлена деятельность испытуемого, а другой ряд — функцию стимулов-средств (знаков), с помощью которых эта деятельность организуется). График, полученный в результате исследований, в действительности лишь напоминает параллелограмм: две кривые сближаются между собой и образуют фигуру, похожую на не вполне правильный параллелограмм с двумя отсеченными углами.

В исследовании приняли участие 1200 испытуемых: дети дошкольного возраста (4-5, 6-7 лет), дети школьного возраста (7-12 лет, 10-14 лет, 12-16 лет), студенты (22-28 лет). Эксперимент состоял из четырёх серий. В первой серии испытуемым предлагалось запомнить ряд из десяти бессмысленных слогов. Во всех остальных сериях в качестве материала для запоминания использовались ряды из 15 слов. В третьей и четвертой серии испытуемым предлагалось выбрать одну из 30 карточек с картинками, чтобы потом с ее помощью вспомнить слово. Картинки не были напрямую связаны по содержанию со словами.

Введение в эксперимент второго ряда стимулов-знаков (карточек с картинками) не повышает эффективности запоминания у дошкольников: запоминание все еще остается непосредственным. На следующем этапе развития запоминания введение стимулов знаков у школьников значительно повышает эффективность запоминания: это момент наибольшего расхождения показателей второй (непосредственное запоминание) и третьей (опосредованное запоминание) серии. Однако с этого момента темп возрастания показателей в обеих сериях изменяется: происходит более медленный рост показателей внешне опосредствованного запоминания, он как бы продолжает темп развития непосредственного запоминания. На следующем этапе (взрослые) за счет вращивания внешних стимулов-средств осуществляется переход к внешне непосредственному запоминанию и показатели двух серий снова сближаются.

Работа А.Н. Леонтьева «Развитие памяти» занимает важное место в эмпирическом фундаменте теории культурно-исторического развития высших психических функций (ВПФ). Это было первое крупномасштабное экспериментальное исследование, результаты которого подтверждали плодотворность и эвристичность теории Л.С. Выготского. Существует много изданий этой книги, но почти нет современных исследований, в которых делалась бы попытка повторить знаменитый «параллелограмм развития». Именно такое редкое исследование было проведено в статье Мещерякова, Моисенко, Канториной.

В данном исследовании были устранены неоднозначности исследования Леонтьева (в работе А.Н. Леонтьева, как справедливо заметил П.А. Мясоед, использовались широкие и перекрывающиеся возрастные диапазоны, что не позволяет однозначно сопоставлять группы с определенным возрастом) и изменен (осовременен) список слов для запоминания.

В заключение хотелось бы отметить, как результаты данных исследований были применены при разработке приложения. Исследования Аткинсона подтвердило идею существования тренировочной игры «Пары слов» как обучающей платформы создания ассоциаций и эффективности метода ассоциаций в целом. Исследования Леонтьева и Мещерякова, Моисенко, Канториной дало идею для создания тренировочных игр «Буквенные ассоциации» (буквенно-опосредственное запоминание в терминах исследований) и «Пары картинок», во многом вдохновленное картинно-опосредственным запоминанием. Причем, буквенно-ориентированное запоминание абсолютно неэффективно у детей дошкольного возраста, поэтому в приложении оно предлагается только начиная со школьного возраста, ведь у школьников и взрослых оно довольно эффективно. «Пары картинок» доступны для пользователей всех возрастов, потому что эффективность картинно-ориентированного запоминания выше, чем непосредственного для всех возрастных групп, у подростков этот разрыв максимален. Более сложные игры («Запомни слова», «Запомни числа», «Пары слов») на построение неочевидных ассоциаций между словами, не имеющими четкой связи, и непосредственное запоминание доступны только подросткам (с 14 лет) и взрослым, так как их эффективность у дошкольников и младших школьников минимальна.

Таким образом, в работе будет реализовано клиент-серверное приложение для развития памяти с использованием следующих инструментов: Node.js, Express.js, MS SQL Server, Typescript. В данном приложении планируется создать тренировочные игры «Пары картинок» для пользователей всех возрастов, «Буквенные ассоциации» для пользователей с 7 лет, «Запомни слова», «Запомни числа», «Пары слов» для пользователей с 14 лет.

Приложение «Тренажер памяти» состоит из клиентской и серверной частей. Каждая часть приложения может существовать независимо, общаясь через определенный API. Это сделано для обеспечения возможности параллельной и независимой работы над приложением в будущем и упрощения поддержки и

масштабирования.

Оба приложения находятся в одном Git-монорепозитории. Основное преимущество, ради которого код хранится в единственной репозитории, состоит в том, что это позволяет синхронизировать версии одного проекта с другим без потери совместимости и предотвратить ад зависимостей (dependency hell) – глобальное разрастание графа зависимостей, добавляющее сложности при установке новых пакетов и обновлении старых.

Основная функция серверной части – обеспечение взаимодействия с базой данных. В качестве системы управления базами данных используется MS SQL Server.

Рассмотрим подробнее модель базы данных MemoryTrainer, созданную в данной работе.

База данных содержит 2 таблицы: Users, необходимая для хранения информации о пользователях, и BestResults, необходимая для хранения информации о лучших результатах пользователя в конкретной игре. По логике приложения сохраняются и обновляются только лучшие результаты для мотивации пользователя к их достижению.

Рассмотрим структуру проекта созданного сервера. Структура максимально упрощена за счет его небольшого размера. Основной функционал приложения находится на клиенте, сервер принимает запросы и взаимодействует с базой данных.

Папка проекта (server) содержит файлы index.js (основной файл приложения с кодом сервера), package.json (вспомогательный файл пакетного менеджера npm, включающий в себя список зависимостей с их версиями, скрипты, которые поддерживает сервер, различную вспомогательную информацию о проекте), package-lock.json (вспомогательный файл пакетного менеджера npm, включающий в себя все дерево зависимостей с их версиями для восстановления на другом компьютере), database-script.sql (скрипт SQL-кода базы данных), .gitignore (файл для скрытия файлов и папок от системы контроля версий Git), README.md (текстовый файл, который распространяется вместе с программным обеспечением и содержит информацию о нем).

Для работы приложения поддерживаются следующие маршруты:

- Login (get) для осуществления входа, т.е. получения информации о существовании пользователя с данным логином и проверка пароля, точнее его

- зашифрованной копии (сама аутентификация на фронтенде);
- `/register (post)` для осуществления регистрации;
- `/users (get)` для получения списка пользователей;
- `/results (get, post)` для получения информации о результатах всех пользователей и добавления нового результата;
- `/my-result` для получения результатов конкретного пользователя.

Основной функционал приложения находится на клиенте, поэтому важно выбрать такую архитектуру проекта, которая будет простой и максимально гибкой и позволит успешно масштабировать приложение, многократно используя повторяющиеся участки кода. Структуры проектов довольно разнообразны: тут и группировка по типу файлов, и модульная структура проекта, и атомарный подход, и разделение компонентов на умные (`smart`) и тупые (`dumb`). В качестве решения данного вопроса было решено группировать по типу файлов в связи с простотой метода. Однако повторяющиеся компоненты вынесены в отдельную папку для повторного использования.

На верхнем уровне созданного проекта (в папке `client`) находятся папки `public`, `src` и файлы `.editorconfig` (конфигурация редактора VS Code), `.eslintrc` (конфигурация ESLint, линтера, используемого для проверки следования кодом лучшим практикам), `.gitignore` (файл для скрытия файлов и папок от системы контроля версий Git), `package.json` (вспомогательный файл пакетного менеджера npm, включающий в себя список зависимостей с их версиями, скрипты, которые поддерживает сервер, различную вспомогательную информацию о проекте), `package-lock.json` (вспомогательный файл пакетного менеджера npm, включающий в себя все дерево зависимостей с их версиями для восстановления на другом компьютере), `.prettierrc` (конфигурация Prettier, используемого для автоматического форматирования кода), `README.md` (текстовый файл, который распространяется вместе с программным обеспечением и содержит информацию о нем), `tsconfig.json` (конфигурационный файл Typescript), `webpack.config.js` (конфигурация сборщика модулей Webpack).

Директория `src` содержит основной код проекта в следующих файлах и папках:

- папка `api` содержит код обращения к серверу;
- папка `assets` используется для иконок и картинок;
- папка `models` включает в себя модели сущностей (типы и интерфейсы,

- используемые в проекте);
- папка `components` содержит компоненты;
- в папку `shared` помещают любой разделяемый для повторного использования код (как компоненты, так и отдельные функции);
- `app.ts` – главный компонент приложения, синхронизирующий его работу;
- `declaration.d.ts` – декларативный или заголовочный файл, используемый для установки связи с внешними файлами (в данном случае, картинками);
- `index.html` – единственный `html`-файл приложения, куда вставляются код и стили (как у всех SPA (Single Page Application));
- `index.ts` – главный `ts`-файл приложения;
- `routing.ts` – файл роутера;
- `styles.scss` – глобальные стили для всего приложения.

Итогом разработки является web-приложение для тренировки памяти, включающее в себя регистрацию пользователя, развивающие игры, рейтинг пользователей и тест.

Приложение состоит из клиентской и серверной частей. Клиентское и серверное приложения работают независимо друг от друга и общаются по API.

При регистрации у пользователя спрашивается адрес электронной почты, дату рождения, пароль, адреса почты должны быть уникальны (один адрес = один аккаунт). Есть возможность загрузки фото профиля.

Развивающие игры разработаны в процессе изучения дополнительной литературы по принципам развития памяти. Приложение адаптировано для пользователей разных возрастов, используя для каждого возраста наиболее эффективные подходы.

Краткая статистика тренировки (процент правильности ответов / время, затраченное на прохождение) показывается после каждой игры. Кроме того, разработан рейтинг пользователей для мотивации к росту и тест для проверки эффективности тренировок.

ЗАКЛЮЧЕНИЕ. В данной работе было реализовано web-приложение «Тренажер памяти» с помощью стека технологий Node.js, Express.js, MS SQL Server, Typescript, описана работа приложения.

Созданное приложение отличается от аналогов своим подходом, учитывающим возраст пользователя и выдающим только те тренировочные игры, которые зарекомендовали себя в соответствующих исследованиях для пользова-

телей данной возрастной группы, что выгодно отличает его от аналогов. Стоит также отметить, что большинство конкурентов приложения доступны только по платной подписке, не русифицированы или работают нестабильно.

По результатам проведенного исследования можно сделать следующие выводы:

- изучены теоретические основы развития памяти;
- создано web-приложение для тренировки памяти, состоящее из клиентской и серверной частей, взаимодействующих через произвольный API;
- описана работа данного приложения;
- изучены Node.js, Express.js, MS SQL Server, Typescript и другие технологии в рамках разработки web-приложения «Тренажер памяти».