ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии

СОГЛАСОВАНО
Заместитель декана по учебно-методической работе
доцент департамента больших данных и информационного поиска факультета компьютерных наук

_____ И. Ю. Самоненко
«__» _____ 2020 г.

УТВЕРЖДАЮ
Академический руководитель образовательной программы «Программная инженерия»
профессор департамента программной инженерии, канд. техн. наук

_____ В. В. Шилов
«__» _____ 2020 г.

WEB ПРИЛОЖЕНИЕ ДЛЯ СОЗДАНИЯ ГЕНЕАЛОГИЧЕСКОГО ДРЕВА

Пояснительная записка

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.04.01-01 81 01-1-ЛУ

Исполнитель
студент группы БПИ173
_____ / Переплетчиков А. И. /
«__» _____ 2020 г.

Москва 2020

Инв. № подл	RU.17701729.04.01-01 81 01-1-ЛУ
Подп. и дата	
Взам. инв. №	
Инв. № дубл.	
Подп. и дата	

УТВЕРЖДЕН
RU.17701729.04.01-01 81 01-1-ЛУ

WEB ПРИЛОЖЕНИЕ ДЛЯ СОЗДАНИЯ ГЕНЕАЛОГИЧЕСКОГО ДРЕВА

Пояснительная записка

RU.17701729.04.01-01 81 01-1

Листов 20

Москва 2020

Инв. № подл	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата
RU.17701729.04.01-01 ПЗ 01-1				

Содержание

1. Введение

1.1. Наименование программы

Наименование программы: «Web приложение для создания генеалогического древа».

1.2. Основания для разработки

Основанием для разработки является приказ декана факультета компьютерных наук Национального исследовательского университета «Высшая школа экономики» № 2.3-02/1004-01 от 10.04.2020 «Об утверждении тем, руководителей курсовых работ студентов образовательной программы Программная инженерия факультета компьютерных наук».

Программа разрабатывается в рамках выполнения курсовой работы по теме «Web приложение для создания генеалогического древа» под руководством доцента департамента больших данных и информационного поиска ФКН И. Ю. Самоненко.

2. Назначение и область применения

2.1. Назначение разработки

2.1.1. Функциональное назначение

Программа представляет собой web-приложение, предназначенное для создания и визуализации генеалогического дерева пользователя. Программа сохраняет введенную пользователем информацию в облачной генеалогической базе данных, а затем визуализирует ее в виде семейного древа на интернет-странице.

2.1.2. Эксплуатационное назначение

Программа позволяет пользователю, проявляющему интерес к изучению и сохранению информации о своей семье, создать собственное генеалогическое древо, сохраненное в облачной базе данных. Программа позволяет отслеживать жизненный путь семьи, а также, в отличие от таблиц и документов, предоставляет информацию в более удобном для визуального считывания формате дерева.

2.2. Область применения

Программа может применяться в генеалогических исследованиях, а также помочь в популяризации изучения семейной истории. Информация, которую пользователи будут вносить при помощи данного приложения, может послужить ценной для формирования общего единого архива генеалогических баз данных в России и русскоязычных странах.

3. Технические характеристики

3.1. Постановка задачи на разработку программы

Программа должна обеспечивать возможность выполнения следующих функций:

1) регистрация новых пользователей, проверка корректности введенных данных;
2) авторизация пользователей, проверка соответствия данных, введенных при авторизации, с данными одного из зарегистрированных пользователей;
3) визуализация генеалогического древа на странице;
4) создание нового родственника, ввод информации о нем;
5) представление информации о каждом родственнике на отдельной странице;
6) указание родственных связей;
7) редактирование информации об уже имеющемся родственнике;
8) удаление родственника из дерева, проверка допустимости удаления (отсутствие разорванных родственных связей);
9) сохранение информации в облачной базе данных Firebase.

3.2. Описание и обоснование алгоритма и функционирования программы

3.2.1. Описание и обоснование общих принципов работы приложения

3.2.1.1. Описание общих принципов работы приложения

Программа реализует архитектуру одностраничного React-приложения, само приложение при этом не обрабатывает непосредственно бэкенд логику или базы данных. Программа осуществляет непосредственно сборку фронтенда, обращающегося к API Firebase для доступа к облачной базе данных и работы с ней в стиле REST.

3.2.1.2. Обоснование выбора общих принципов работы приложения

В данном проекте для реализации была выбрана исключительно фронтенд-часть. У данного решения есть несколько обоснований:

- Бэкенд логика подобного приложения достаточно примитивна в сравнении с основной и наиболее трудоемкой фронтенд-частью: необходима простая база данных, способная хранить пользователей, родственников и родственные связи, такую БД и API для работы с ней предоставляет Firebase; в то время, как фронтенд часть подразумевает реализацию рендеринга дерева, механизма добавления новых узлов в дерево, доступ к полям ввода и т.д.
- Облегчение процесса разработки, при использовании Firebase отпадает необходимость повторно реализовывать стандартные механизмы авторизации, доступа к базе данных и прочую бэкенд-логику.
- Облегчение процесса развертывания приложения на сервере Firebase.

3.2.2. Описание способа хранения дерева в базе данных

Firebase предоставляет облачную СУБД NoSQL класса, удобную для использования в javascript-коде, т.к. позволяет нам напрямую сохранять программные объекты в json-документы.

У каждого пользователя в базе данных есть доступ к двум коллекциям: people и families. Коллекция people хранит в себе документы с информацией о родственниках. Документ в коллекции people имеет следующую структуру:

- string id – уникальный идентификатор документа
- string firstName – имя человека
- string secondName – отчество человека
- string lastName – фамилия человека
- map birthDate – дата рождения человека
  - string day – день рождения
  - string month – месяц рождения
  - string year – год рождения
- map deathDate – дата смерти человека
  - string day – день смерти
  - string month – месяц смерти
  - string year – год смерти
- string sex – пол человека («мужской» или «женский»)
- string nativeCity – место рождения человека
- string photo – url фотографии человека
- string bio – биография человека

Как видно из структуры документа в коллекции people, непосредственно сами люди не имеют никаких родственных связей (например, ссылки на родственников с указанием типа связи родитель/супруг/ребенок). Всю информацию о связях между людьми хранит в себе коллекция families. Документ коллекции families, представляющий одну минимально допустимую семейную ячейку (супруги и их дети), так называемую «малую семью», имеют следующую структуру:

- string id – уникальный идентификатор документа, начинается на ‘_’
- string husband – id мужа
- string wife – id жены
- string husbandFamily – id семьи родителей по мужской линии
- string wifeFamily – id семьи родителей по женской линии
- array children – массив с id детей, которые могут являться как отдельными людьми (в таком случае указывается id документа из коллекции people), так и самостоятельными семейными ячейками (в таком случае указывается id документа из коллекции families)

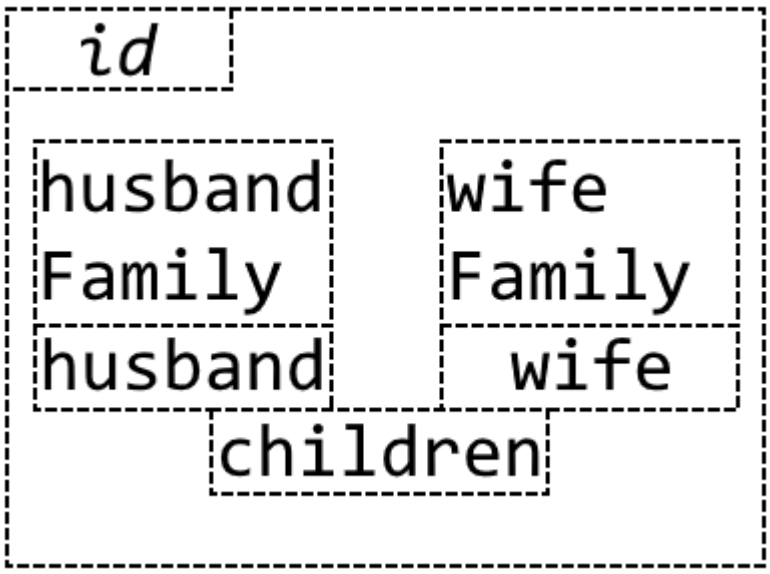

Рис. 1. Способ организации семейной ячейки «малой семби» в базе данных

3.2.3. Описание алгоритма рендеринга дерева на странице

React-компонент Tree, осуществляющий рендеринг дерева, получает коллекции people и families в качестве пропсов от родительского компонента, осуществляющего непосредственно выгрузку актуальных данных с сервера. Для того, чтобы отрендерить дерево на странице, программе нужно сгенерировать jsx-код, который библиотека React впоследствии превратит в HTML-разметку. Алгоритм рендеринга дерева делится на два этапа: 1) рекурсивный рендеринг отдельной семейной ветви, состоящих из идущих друг за другом связанных «малых семей», и 2) сбор всех ветвей древа в единую «большую семью», включающую всех родственников, связанных близким или шире-дальным родством. Отдельной семейной ветвью будем называть группу малых семей, связанных прямыми родственными связями, при этом:

- Либо не должно быть конфликтующих родительских ветвей, т.е. родительская семья указана лишь для одного из супругов или не указана вовсе.
- Либо, при наличии конфликтующих родительских семей (т.е. когда родители указаны и для мужа, и для жены), предпочтение отдается мужской ветви.

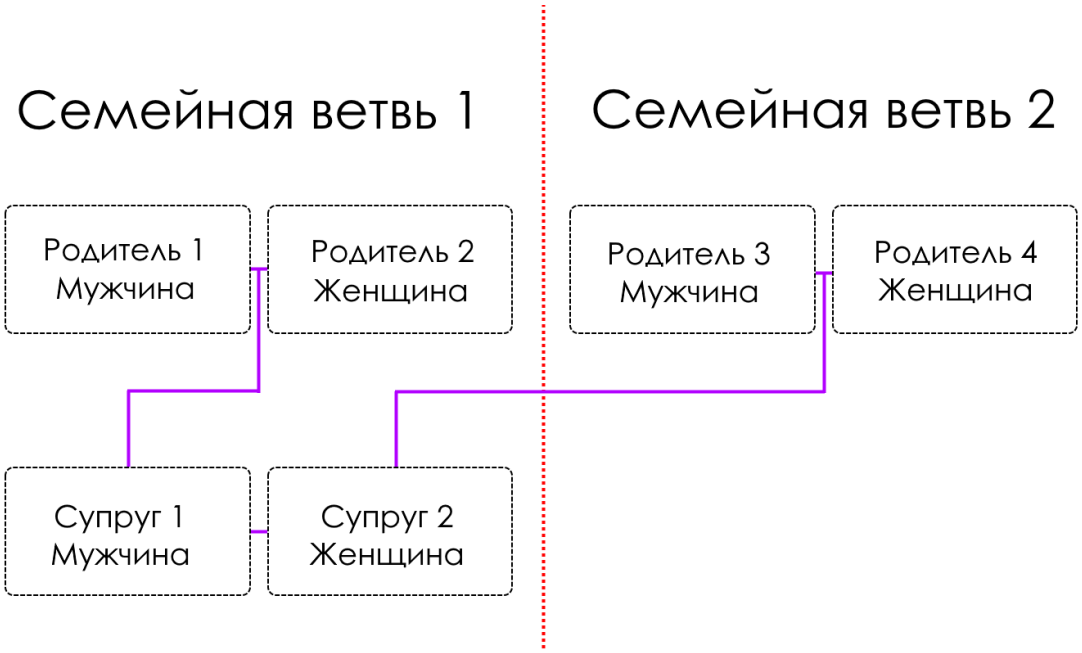

Рис. 2. Пояснение к объединению семейных ветвей (при объединении двух веток результат слияния отходит к мужской ветке)

За рендеринг отдельной семейной ветви отвечает метод familyLayout, получающий на вход документ family. В зависимости от условий, в которых находится текущая семья, familyLayout выбирает одну из опций:

1) Если родителей нет ни у одного из супругов, сохраняется глубина, на которую ушла вверх наша ветвь, она пригодится для дальнейшего вертикального ранжирования ветвей;
2) Если родительских семей нет или они уже отрисованы, отрисовываем непосредственно самую семью, и рекурсивно запускаем функцию для детских семей;
3) Если есть лишь одна родительская семья, и она еще не отрисована, запускаем функцию от нее, тем самым помещая ее в текущую ветвь;
4) Если есть обе родительские семьи, и они обе не отрисованы, переходим к семье по мужской линии;
5) Если есть обе родительские семьи, и мужская уже отрисована, помещаем женскую в очередь ветвей для рендеринга и отрисовываем текущую семью в данной ветке;
6) Если есть обе родительские семьи, и женская уже отрисована, помещаем мужскую в очередь ветей для рендеринга, а текущую семью просто не отрисовываем;
7) Если есть обе родительские ветви, и они уже отрисованы, отрисовываем текущую семью только, если в данный момент находимся в мужской ветке.

Для того, чтобы собрать все ветви дерева вместе и отобразить их на странице, используется метод treeLayout:

1) Метод treeLayout объявляет локальный массив renderedFamilies, каждый элемент которого хранит jsx-код, отображающий отдельную ветку дерева.;
2) В массив renderedFamilies помещается результат работы функции familyLayout от первого (нулевого) документа из коллекции families. Функция familyLayout возвращает jsx-код для отдельной ветки семейного древа (т.е. все семьи, связанные воедино родственными связями);
3) Затем, пока глобальный массив familiesToRender, представляющий очередь из id еще не отрисованных семей, не опустеет, вызывается функция familyLayout для каждого из id в очереди. Результат работы функции для каждого id складывается в массив renderedFamilies;
4) Для каждого из элементов массива renderedFamilies высчитывается глубина, на которую ушла вверх данная ветка;
5) Все элементы массива renderedFamilies оборачиваются в блоки div со стилем вертикального отступа (margin-top), пропорционального глубине, на которую ушла ветвь.

Затем библиотека React генерирует jsx в HTML-разметку, отображаемую на странице в веб-браузере.

3.2.4. Описание организации HTML-разметки дерева

Каждая семейная ячейка представляет собой div с классом family с двумя вложенными div-ами: parents и children. В div-е parents находятся карточки родителей, в div-е children – карточки детей (если они одни) или еще div-ы с классом family, если у детей есть собственные семьи.

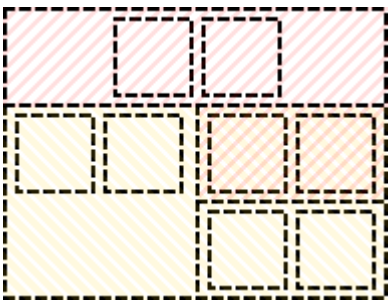

Рис. 3. Организация разметки: красным отмечены div-ы с классом parents, желтым – children. Все это составляет div family.

3.3. Описание и обоснование метода организации входных и выходных данных

3.3.1. Описание метода организации входных и выходных данных

3.3.1.1. Описание метода организации входных данных

Ввод данных происходит посредством стандартных html-форм: текст, число, кнопка, переключатели, поле со списком, загрузка файлов. Входные данные упаковывается в JSON документы и помещаются в коллекции NoSQL базы данных.

3.3.1.2. Описание метода организации выходных данных

Программа выводит семейное дерево в следующем виде:

- Узлы дерева – карточки с информацией о каждом родственнике (фотография, ФИО, место и дата рождения), при клике на карточку открывается страница с расширенной информацией о человеке.
- Ребра дерева – линии, отображающие родственные связи между людьми в узлах; ребра проводятся между двумя супругами и от родителя к ребенку.

Программа должна вносить изменения в БД в соответствии с введенными пользователем данными.

3.3.2. Обоснование метода организации входных и выходных данных

3.3.2.1. Обоснование метода организации входных данных

Выбранный метод организации входных данных является фактическим стандартом для интернет-страницы и поддерживается всеми современными веб-браузерами.

3.3.2.2. Обоснование метода организации выходных данных

В генеалогии семейное древо является одним из наиболее простых и наглядных способов представления истории родственных групп семей. Оно дает наглядную инфографику родственных связей, отображает хронологический порядок появления новых членов семьи.

3.4. Описание и обоснование выбора состава технических и программных средств

3.4.1. Состав технических и программных средств

Для нормального функционирования программы требуется компьютер, оснащенный следующими техническими компонентами:

1) процессор не ниже Intel Pentium/Celeron, AMD K6/Athlon/Duron или совместимый с ними с тактовой частотой не ниже 1 ГГц;
2) 512 Мб ОЗУ или более;
3) не менее 10 Мб свободной памяти на жестком диске для хранения кэша приложения;
4) VGA-совместимые видеоадаптер и монитор с разрешением не ниже 1280x800;
5) интернет-канал со скоростью соединения не менее 1 Мбит/сек
6) клавиатура и мышь.

Для нормального функционирования программы требуется компьютер, оснащенный следующими программными компонентами:

1) операционная система Microsoft Windows XP (SP2, SP3) / Vista / 7 / 8 / 8.1 / 10;
2) любой современный десктопный веб-браузер (Google Chrome, Mozilla Firefox, Opera последних версий), поддерживающий технологии HTML5, CSS3 и JS ES6;

3) если пользователь хочет запустить локальный сервер на компьютере, требуется установленная программная платформа Node.js с пакетным менеджером npm, необходимые для запуска сервера на локальной машине.

3.4.2. Обоснование выбора состава технических и программных средств

Приложение осуществляет сборку фронтенда, для отображения и корректного функционирования которого требуется современный веб-браузер. Все минимальные необходимые технические и программные средства, определенные для корректной работы приложения, обусловлены техническими и программными средствами браузера.

Для запуска локального сервера на компьютере пользователя необходима установленная программная платформа Node.js, включающая пакетный менеджер npm, осуществляющий установку всех необходимых для работы приложения пакетов. Сама Node.js позволяет транслировать Javascript, на котором написана программа, в машинный код.

4. Технико-экономические показатели

4.1. Предполагаемая потребность

На момент разработки на русскоязычном и зарубежном рынках существует несколько компьютерных программ и веб-сайтов для создания и управления генеалогическими базами данных (Gramps, FamilySpace, GenoPro, Geni.com, Родственники и т.д.). Все эти продукты предоставляют широкий функционал для работы с родословной, некоторые из них предоставляют возможность поиска родственников по открытым базам данных. Тем не менее, те из них, что продолжают поддерживаться разработчиками и сообществом, являются сильно усложненными и перегруженными дополнительными возможностями (например, для того, чтобы воспользоваться программой Gramps, обязательно необходимо потратить время на изучение инструкции к ней вплоть до изучения специальных терминов), а те, кто, как и разрабатываемое приложение, предоставляет простой и самый необходимый функционал, уже безнадежно устарели и вряд ли могут удовлетворить потребности современных пользователей (например, программа «Родственники» была выпущена в 2008 году и с тех пор претерпела несильные изменения). Помимо этого, разрабатываемое приложения является бесплатным.

4.2. Ориентировочная экономическая эффективность

Программа предполагает функционирование в виде отдельного бесплатного веб-сайта с полностью бесплатным свободным доступом. Обеспечением работы бэкенда и хранения базы данных занимается компания Firebase, предоставляющая свои услуги (в необходимом для проекта объеме) бесплатно. Иные расчеты экономической эффективности не проводились.

Приложение 1. Список используемой литературы

1. ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
2. ГОСТ 19.301-79 Программа и методика испытаний. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3. ГОСТ 19.401-78 Текст программы. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4. ГОСТ 19.505-79 Руководство оператора. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5. ГОСТ 19.404-79 Пояснительная записка. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7. ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
8. React – JavaScript-библиотека для создания пользовательских интерфейсов [Электронный ресурс] // URL: https://ru.reactjs.org/ (Дата обращения: 11.04.2020, режим доступа: свободный).
9. Documentation | Firebase [Электронный ресурс] // URL: https://firebase.google.com/docs (Дата обращения: 11.04.2020, режим доступа: свободный).

Приложение 2. Описание и функциональное назначение компонентов и функций

Табл. 1. Описание и функциональное назначение компонентов

Компонент	Аргументы (props)	Назначение
AuthProvider	children – дочерние компоненты, которым предоставляется контекст	Предоставляет контекст всем дочерним компонентам. Target контекста – константа currentUser, хранящая текущего авторизованного пользователя и loading, отображающая то, что в данный момент приложение получает информацию с сервера. В компонент AuthProvider обернуто все приложение.
Routes	-	Осуществляет маршрутизацию: по указанным маршрутам (Route) возвращает соответствующий нужный компонент (для главной страницы / - Main, для страницы добавления /add – Add и т.д.
Add	person – объект, представляющий добавляемого родственника	Страница добавления родственника. Содержит формы для добавления информации о нем, обрабатывает и передает управление глобальной функции addMethod.
Info	-	Страница с информацией о сайте. Ничего не обрабатывает, возвращает статичный HTML.
Loading	-	Компонент, отображающийся каждый раз, когда происходит выгрузка или загрузка данных на сервер.
Login	history – глобальный объект истории веб-браузера, нужен для того, чтобы перейти в компонент Main после успешной авторизации	Страница авторизации. Содержит форму авторизации и обработчик авторизации, создающий токен авторизации.
Main	returnFunction – функция, передаваемая, если компонент Main вызывается из компонента Add, вновь возвращает компонент Add (для возвращения к форме добавления)	Основная страница сайта. Отображает компонент Tree, а также обработчики для добавления/удаления людей из дерева.

	newPerson – объект, представляющий добавляемого родственника, если компонент Main вызывается из компонента Add для выбора родственника на дереве	
SignUp	history – глобальный объект истории веб-браузера, нужен для того, чтобы перейти в компонент Main после успешной регистрации	Страница регистрации. Содержит форму регистрации и обработчик регистрации, создающий нового пользователя в БД.
Menu	-	Компонент бокового меню.
Navbar	-	Компонент главной панели с логотипом сайта.
EditPerson	person – объект, представляющий редактируемого родственника	Страница редактирования информации о родственнике, вносит изменения в БД.
PersonCard	person – объект, представляющий отображаемого родственника	Компонент карточки родственника, отображающейся в дереве.
PersonPage	person – объект, представляющий отображаемого родственника	Страница родственника, открывающаяся при клике на карточке родственника, содержит более полную информацию
SelectDate	personDay – день (если он уже выбран и нужно отобразить его в форме) personMonth – месяц (если он уже выбран и нужно отобразить его в форме) personYear – день (если он уже выбран и нужно отобразить его в форме) required – булевый флаг, отображающий обязательно ли форма для заполнения typeOfDate – ‘birth’ или ‘death’, т.к. компоненты SelectDate используются в одних и тех же формах и необходимо понимать, за какую дату они отвечают	Вспомогательный компонент-форма для ввода дат. Осуществляет проверка корректности ввода.
Tree	people – коллекция родственников из базы данных families – коллекция родственных связей из базы данных candidate – объект родственника, который будет добавлен в дерево sendRelative – функция, предоставляемая родительским	Основной компонент приложения, отображающий генеалогическое дерево. Также содержит обработчики функций выбора родственника и типа родственной связи, добавления нового родственника.

	компонентом Main, отправляет id выбранного родственника и выбранный тип связи returnFunction – функция, возвращающая компонент Add	

Табл. 2. Описание и функциональное назначение функций компонента Add

Функция	Аргументы	Возвращаемое значение	Назначение
handleInput	event – событие, вызвавшее выполнение функции	-	Обработчик изменения значения в одном из полей формы
handleAdd	event – событие, вызвавшее выполнение функции	-	Обработчик нажатия на кнопку «Добавить»

Табл. 3. Описание и функциональное назначение функций компонента Login

Функция	Аргументы	Возвращаемое значение	Назначение
handleLogin	event – событие, вызвавшее выполнение функции	-	Обработчик нажатия на кнопку «Войти», перенаправляет на главную страницу сайта

Табл. 4. Описание и функциональное назначение функций компонента Main

Функция	Аргументы	Возвращаемое значение	Назначение
addToTree	newPerson – добавляемый в дерево человек	-	Получает снапшоты базы данных с сервера и передает управление в глобальную функцию addMethod

Табл. 5. Описание и функциональное назначение функций компонента SignUp

Функция	Аргументы	Возвращаемое значение	Назначение
handleSignUp	event – событие, вызвавшее выполнение функции	-	Обработчик нажатия на кнопку «Регистрация», создает нового пользователя в БД, перенаправляет на главную страницу сайта

Табл. 6. Описание и функциональное назначение функций компонента EditPerson

Функция	Аргументы	Возвращаемое значение	Назначение

handleInput	event – событие, вызвавшее выполнение функции	-	Обработчик изменения значения в одном из полей формы
handleEdit	event – событие, вызвавшее выполнение функции	-	Обработчик нажатия на кнопку «Изменить». Сохраняет изменения в БД.

Табл. 7. Описание и функциональное назначение функций компонента PersonCard

Функция	Аргументы	Возвращаемое значение	Назначение
handleBodyClick	-	-	Функция, реагирующая на клик по карточке человека. Изменяет стили карточки.
handleRelationshipClick	msg – сообщение, передаваемое функции, может принимать одно из трех значений: ‘parent’, ‘spouse’, ‘child’	-	Функция, вызывающаяся при выборе одной из опций для выбора родственника: родитель/супруг/ребенок и вызывающая функцию-обработчик родительского компонента

Табл. 2. Описание и функциональное назначение функций компонента SelectDate

Функция	Аргументы	Возвращаемое значение	Назначение
handleChange	event – событие, вызвавшее выполнение функции	-	Изменяет состояние компонента SelectDate в соответствии с выбранной при помощи формы даты

Табл. 8. Описание и функциональное назначение функций компонента Tree

Функция	Аргументы	Возвращаемое значение	Назначение
edges	-	jsx	Рендерит связи между родственниками. Связи – ломанные линии толщиной 1px, являющиеся компонентами модуля SteppedLine
family	family – документ малой семьи из коллекции families,	jsx	Рендерит отдельную «малую семью», включающую карточки

	depth – глубина, на которой находится семья в дереве		родителей и детей, рекурсивно вызывает функции family от связанных с этой семьей других семей
handlePersonSelection	id – уникальный идентификатор выбранного человека, relationship – выбранный тип связи	-	Обработчик нажатия на карточке одного из родственников, сохраняет в состоянии компонента Tree выбранные id и relationship для дальнейшей работы
handleClick	-	-	Обработчик нажатия на кнопку «Добавить», передает id и relationship в родительский объект
findById	id	Документ из нужной БД	Вспомогательный метод для поиска людей и семей в коллекциях people и families соответственно
familyLayout	family – документ малой семьи из коллекции families, depth – глубина, на которой находится семейная ветка в дереве, parentsId – id семьи родителей	jsx	Рендерит отдельную семейную ветвь, состоящую из множества близко-родственных малых семей, на нужной высоте
treeLayout	-	jsx	Рендерит дерево, состоящее из всех семейных ветвей на нужных высотах

Табл. 9. Описание и функциональное назначение глобальных функций

Функция	Аргументы	Возвращаемое значение	Назначение
removeMethod	personId – id удаляемого родственника currentUserId – uid текущего пользователя, к которому привязана БД	-	Удаляет родственника из базы данных. Проверяет и сообщает, если сделать это невозможно (разрыв связей).

addMethod	newPerson – добавляемый родственник relativeId – id родственника для связи relationship – тип связи people – коллекция людей families – коллекция семей	-	Добавляет нового родственника в базу данных. Сообщает о возможных ошибках.

Лист регистрации изменений

Изм.	Номера листов				Всего листов (страниц) в документе	№ документа	Входящий № сопроводит. докум. и дата	Подпись	Дата
	измененных	замененных	новых	аннулированных					