**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА» (СПбГУТ)**

КАФЕДРА ПРОГРАММНОЙ ИНЖЕНЕРИИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Дисциплина: «WEB-технологии»

**Тема: «Приложение для ведения журнала питания»**

Выполнил:

Студент группы ИКПИ-25

Язенцев А. С.

Подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Принял:

Старший преподаватель кафедры ПИиВТ

Краева Е. В.

Подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оглавление

[Оглавление 2](#_Toc185256347)

[Введение 3](#_Toc185256348)

[1. Описание технологий для разработки веб-приложений, классификация информационных систем 4](#_Toc185256349)

[1.1. Существующие и используемые в работе технологии для разработки веб-приложений 4](#_Toc185256350)

[1.2. Существующие и используемые в работе среды разработки 7](#_Toc185256351)

[1.3. Классификация информационных систем и тип разрабатываемого приложения 8](#_Toc185256352)

[2. Разработка приложения управления бронированиями 9](#_Toc185256353)

[2.1. Разработка архитектуры приложения 9](#_Toc185256354)

[2.2. Разработка клиентской части приложения 10](#_Toc185256355)

[2.3. Разработка серверной части приложения 12](#_Toc185256356)

[2.4. Взаимодействие клиента и сервера 14](#_Toc185256357)

[2.5. Описание приложения 16](#_Toc185256358)

[2.5.1. Описание базы данных 16](#_Toc185256359)

[2.5.2. Инструкция запуска приложения. 18](#_Toc185256360)

[2.5.3. Инструкция пользователю 19](#_Toc185256361)

[2.6. Пути развития 27](#_Toc185256362)

[Список литературы 28](#_Toc185256363)

[Выводы 29](#_Toc185256364)

[Приложение 30](#_Toc185256365)

Введение

Цель работы состоит в разработке web-приложения для ведения журнала питания.

В процессе создания приложения необходимо решить следующие вопросы:

* Определить пользователей приложения
* Определить функционал приложения
* Определить структуру приложения и способы взаимодействия пользователей с приложением
* Определить технологии, используемые для разработки

После решения этих вопросов можно переходить к разработке.

Определимся с пользователями приложения. От этого зависит то, какое приложение разрабатывать. Пользователь приложения – любой человек, желающий отслеживать своё питание с какой-либо целью: набором, сбросом или поддержанием веса.

При этом для пользователей доступен следующий функционал:

Посетители имеют возможность:

* Создать аккаунт, войти в него
* Создать и удалять записи о приёме пищи
* Просмотреть записи о приёмах пищи
* Сохранить записи о приёме пищи в избранное
* Получить быстрый доступ к показателям приёма пищи из избранного для создания новых записей
* Просмотреть статистику по суммарным макронутриентам в записях за день, неделю или месяц
* Получить рекомендацию по количеству потребляемых макронутриентов для достижения выбранной пользователем цели

Описание технологий для разработки веб-приложений, классификация информационных систем

## Существующие и используемые в работе технологии для разработки веб-приложений

Для создания сайтов существует множество технологий и инструментов. Чтобы избавиться от программирования или же уменьшить его объём при создании сайта могут использоваться системы управления контентом или же CMS, такие как WordPress, Drupal, Joomla. Также для создания веб-приложений используются различные языки программирования, такие как C++, Go, Java, Python, PHP, JavaScript, TypeScript и другие. Различные языки имеют в своей основе некоторую философию программирования, базовый набор функций и разработанных библиотек. Также от выбора языка программирования для сервера зависит его быстродействие и скорость разработки. Так, принято считать, что разработка кода на С++ занимает больше времени, чем на Python, однако на С++ легче написать оптимизированный и быстродействующий код.

Для разработки клиентской части веб-приложений также используются различные инструменты, библиотеки и фреймворки, как-то: HTML, CSS, Bootstrap, jQuery, React, AngularJS и другие. Так, например Bootstrap и React предлагают библиотеки под лицензией MIT для ускорения разработки на основных языках клиентской части веб-приложений HTML, CSS и JavaScript. Bootstrap специализируется на дизайне приложений, когда React изначально нацелен больше на функциональность. Однако оба этих фреймворка имеют схожую область применения. В тоже время AngularJS и jQuery ставят своей основной целью изменить процесс программирования на JavaScript.

Фреймворки предлагают программистам ускорить процесс разработки благодаря встроенным библиотекам, а написанный на фреймворках код транслируется в HTML, CSS и JavaScript. Библиотеки фреймворков в некоторых случаях бывают настолько большими, что фреймворк, по сути, становится отдельным языком и требует затрат времени на его изучение.

Выбор технологий для создания веб-приложения зависит от требований к функциональности, сложности проекта, предпочтений разработчика и т.д. Кроме того, часто используются комбинации различных технологий, в особенности в больших командах разработки и сложных проектах. В таблице 1 показано сравнение описанных технологий.

Таблица 1. Сравнение технологий для разработки клиентской части веб-приложений

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Технология/Язык** | **Защита от ошибок** | **Оптимизация производительности** | **Скорость разработки** | **Гибкость разработки** | **Сложность освоения** |
| CMS (Wordperss, Durpal, Joomla) | Защита от ошибок полностью ложится на плечи системы управления контентом | Веб-приложения, разрабатываемые с использованием CMS трудно оптимизировать, поскольку нет доступа на низкий уровень | Высокая скорость разработки достигается путём упрощения создания веб-приложения | Если конкретная CMS не поддерживает некоторый функционал - его почти невозможно будет реализовать | CMS разрабатывались и являются простым способом создания веб-приложений и не требуют долгого изучения |
| Python (Djanjo, Flask) | Безопасность создается за счет удачного конфигурирования, ситуативно | Используется обширная коллекция библиотек и инструментов, что существенно замедляет работу | Большое число библиотек значительно ускоряет работу | Хоть фреймворки и позволяют вызывать все поддерживаемые функции браузера, однако не напрямую, что снижает гибкость разработки. | Большое число библиотек требует значительного времени на их изучение |
| WEB (HTML, CSS, JavaScript) / (Bootstrap, React) | Не гарантируют безопасность на этапе компиляции | Хорошо работают с большими нагрузками и имеют возможность переходить на максимально возможно низкий уровень - объекты DOM (Document object model) | Низкая, поскольку доступны лишь базовые функции, поддерживаемые браузером. Использование фреймворков позволяет ускорить работу | Полное и прямое управление поддерживаемыми функциями браузера. | Набор языков WEB нельзя назвать тяжёлыми языками, однако конечно требует некоторое время на освоение. Использование же фреймворков требует дополнительного изучения их функционала. |

Для разработки клиентской части приложения (т.н. «фронтэнд») было решено использовать стандартный набор языков разметки и программирования для создания web-приложений:

* HTML – HyperText Markup Language – язык гипертекстовой разметки документов для просмотра в браузере. Браузер пользователя получает HTML-файл от сервера и отображает его содержимое, что позволяет пользователю взаимодействовать с приложением.
* CSS – Cascading Style Sheets – каскадные таблицы стилей. В дополнение к HTML-файлу позволяют задать стили веб-страницы: цвет текста, его шрифт, цвета и размер кнопок, фон страницы и многое другое что позволит странице приятный глазу пользователей вид.
* JavaScript – язык программирования, позволяющий изменять содержимое HTML-страницы у пользователя, реализовывать алгоритмы, по которым приложение будет работать а так-же обмениваться запросами с сервером. Что в совокупности позволяет реализовать логику приложения, обрабатывая действия пользователей в приложении.

Выбор этого набора технологий обусловлен невысокой сложностью приложения и отсутствием действительного сокращения времени при разработке приложения командой разработки при использовании другого набора технологий. Команда разработки не имеет опыта работы с иными технологиями кроме как с вышеуказанными, поэтому использование иных инструментов потребовало бы их изучения, что могло занять значительный объём времени. Также от использования CMS решено отказаться и по причине желания группы разработки получить опыт разработки веб-приложений «с нуля». Использование фреймворков было допустимо, однако группа разработки решила при создании web-приложения изучить базовые технологии веб-приложений, прежде чем переходить к более высокому уровню. Именно поэтому клиентская часть приложения написана с помощью стандартного набора языков.

Для создания сервера был выбран язык Kotlin в паре с фреймворком Ktor, поскольку этот фреймворк лёгок в изучении, предназначен именно для разработки веб-серверов и содержит весь необходимый инструментарий и у группы разработки есть опыт работы с похожим на Kotlin языком – Java. Языков, на которых можно реализовать веб-сервер предостаточно, это может быть, например Go или Java. Однако поскольку требований к языку реализации не представлено, то выбор серверного языка это тоже по большей части воля команды разработки.

## Существующие и используемые в работе среды разработки

Выбор среды разработки по большей части — это вопрос удобства программиста. Выбор среды для разработки клиентской части основывается на критериях удобства запуска и отладки находящейся в разработке веб-страницы (для разработки клиентской части) или веб-сервера. Все популярные (Visual Studio, Visual Studio Code, IntelliJ IDEA, WebStorm, Sublime Text, Komodo IDE) на данный момент среды разработки поддерживают встроенную отладку, авто-дополнение, поддержку всех необходимых языков и фреймворков для разработки. Однако из-за того, что компиляция HTML, CSS и JavaScript в исполняемый файл не требуется, поскольку HTML и CSS это не языки программирования, а JavaScript – JIT-компилируемый язык (компиляция в момент исполнения) и отлаживается прямо в браузере, выбор среды для разработки клиентской части сужается до критериев лишь удобства написания кода и использования среды разработки.

Для разработки клиентской части веб-приложения была выбрана среда разработки IntelliJ IDEA. Она поддерживает интеграцию с GitHub, что было полезно в процессе разработки.

Приложение для запуска web-сервера это CLI (command line interface) приложение, которое использует исходные текстовые файлы. Сборка происходит в нём-же. Поэтому среда разработки состоит из приложения и выбранного редактора текстовых файлов. Для разработки серверной части была выбрана также IntelliJ IDEA.

## Классификация информационных систем и тип разрабатываемого приложения

Согласно ГОСТ РВ 51987 информационная система (ИС) – автоматизированная система, результатом функционирования которой является представление выходной информации для последующего использования. Таким образом задача любой информационной системы – предоставлять пользователям некоторую запрашиваемую ими информацию. Задачей нашей информационной системы является предоставление пользователям интерфейса для ведения журнала питания и предоставление информации о сохранённых данных. Информационные системы классифицируются по архитектуре, степени автоматизации, характеру обработки данных, области применения, масштабности.

Разрабатываемая информационная система классифицируется как распределённая клиент-серверная двухзвенная автоматизированное информационно-справочная экономическая групповая система, что и показано на рисунке 2.

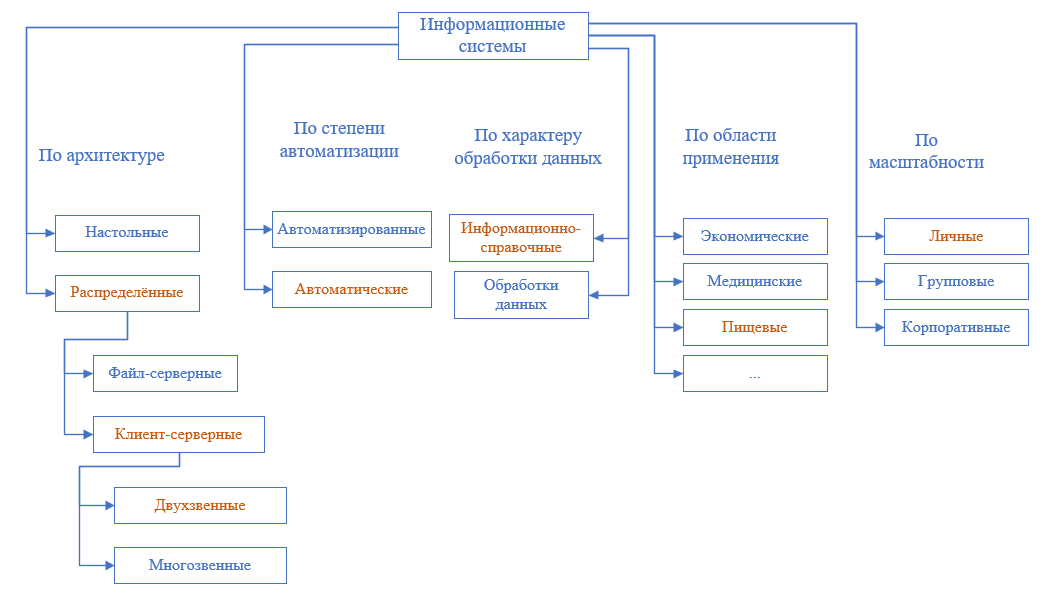


Рисунок 2. Классификация информационных систем, классификация разрабатываемой ИС

Разработка приложения управления бронированиями

## Разработка архитектуры приложения

Архитектура приложения состоит из браузера клиента, сервера и базы данных, хранящейся на сервере. Доступ сервера к базе данных MySQL осуществляется через интерфейс JDBC. Обмен информацией между клиентом и сервером осуществляется по протоколу HTTP.

Доступ к приложению будет осуществляться через интернет c использованием HTTP запросов по ip-адресу сервера. На сервере же будет храниться база данных и статические файлы (HTML, CSS, JS) страниц приложения. Через web-приложение и его интерфейс будет осуществляться доступ к ресурсам на сервере. Описанная архитектура приложения показана на рисунке 3

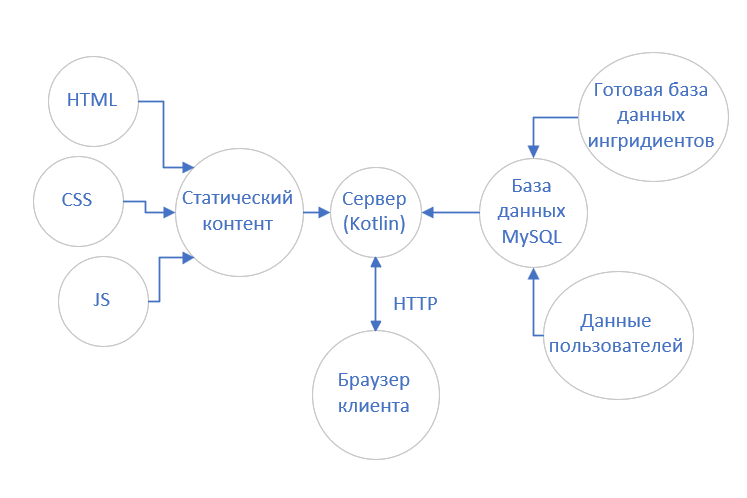


Рис 3. Архитектура приложения

## Разработка клиентской части приложения

При разработке клиентской части приложения необходимо разработать статические файлы (HTML, CSS, JS) для веб-страниц приложения. Было решено создавать веб-приложение, состоящее из двух страниц:

* Основная страница
* Страница входа в аккаунт

На главной странице будет располагаться основной функционал приложения, на странице входа пользователь сможет либо создать новый аккаунт, либо войти в существующий.

Для ранее описанных страниц веб-приложения разработаны следующие файлы:

* index.html (.css .js) – основная страница, её стили и скрипты.
* login.html (.js) – страница входа в аккаунт, её скрипты.

Также страница входа в аккаунт наследует стили основной страницы.

Файлы calendar.js и calendar.css содержат скрипты и стили для работы встроенного в приложение календаря. Эти скрипты используются как на основной странице, так и на странице администрирования.

Файлы с расширением .js написаны на языке JavaScript и описывают алгоритмы и логику работы клиентской части приложения. Файлы содержат функции, которые вызываются при некоторых событиях (нажатие кнопки, ввод сообщения). Внутри функций с помощью языка JavaScript и интерфейса DOM (Document Object Model) происходит взаимодействие с элементами веб-страницы, отображаются или скрываются её элементы, считываются и изменяются состояния объектов, а также отправляются запросы на сервер. Помимо функций в этих файлах есть строки, не являющиеся частью никакой функции. Это алгоритмы, которые необходимо выполнить один раз – при запуске приложения. Так, например, это получение данных о том, авторизирован ли пользователь или нет

После введения в строку адреса пользователем ip-адреса сервера сервер отправит пользователю файлы login.html, login.js, index.css и браузер пользователя отобразит страницу входа в аккаунт в соответствии с содержимым файлов. При успешной регистрации пользователь получит файлы основной страницы приложения и сможет взаимодействовать с интерфейсом журнала питания.

Клиенту не требуется выполнять сложных вычислений или хранить большие массивы информации. Каждая функция приложения заключается в обмене информацией между сервером и клиентом, и незначительной её фильтрации и обработке, поэтому клиент в данном приложении – «тонкий».

## Разработка серверной части приложения

Разработка полнофункционального HTTP-сервера на языке Kotlin предполагает обслуживание как статического контента, так и набора конечных API-точек (endpoints). Хотя и возможно ограничиться выдачей готовых шаблонизированных веб-страниц, такой подход был отвергнут в связи с осложнением разработки.

Сервер использует фреймворк Ktor. С его помощью функциональность разделяется на множество процедур-обработчиков, ответственных за обслуживание определённых URL-адресов. По назначению обработчики условно разделены следующим образом:

* Обработчики статического контента: сервер предоставляет заготовленные статические файлы (HTML, CSS, JavaScript, изображения) из определенной директории. Требования к таким обработчикам минимальны: необходимо отдать запрашиваемый ресурс, при необходимости проверив, аутентифицирован ли клиент.
* Обработчики конечных API-точек (endpoints): сервер предоставляет набор URL-путей для обращений на выполнение HTTP-запросов. Эти точки реализуют CRUD-модель: четыре принципа взаимодействия с данными. То есть, предоставляется интерфейс для создания, чтения, обновления и удаления ресурсов сервера. Например, клиент может отправить GET-запрос по пути “/incomes” для получения списка записей о приёмах пищи, а затем направить DELETE-запрос по пути “/incomes/5” для удаления приёма пищи с идентификатором “5”. В целях унификациии, сервер принимает и возвращает ресурсы исключительно в JSON-формате, аналогично большинству других HTTP-серверов. Функционал процедур, реализующих обработчики ограничен: в основном он состоит из сериализации JSON-данных и передачи их клиенту и / или базе данных, в зависимости от запроса.
* Безопасность достигается методами фреймворка и основывается на cookie-токенах, которые хранят в себе зашифрованные данные о пользователе и идентифицируют его при запросах на сервер. Пароли также хэшируются методами фреймворка, что обеспечивает дополнительную безопасность

В процессе разработки были определены обработчики, обслуживающие пути следующего вида:

* “/auth” для проверки валидности cookie-токена.
* “/login” для входа и выхода из аккаунта соответственно запросами POST и DELETE
* “/incomes” для получения всех записей о приёмах пищи от пользователя, создания изменения и удаления их
* ”/incomes/{id}” для удаления конкретной записи о приёме пищи по её идентификатору.
* “/incomes/{page}” для получения страницы с записями о приёмах пищи с указанием номера страницы. Размер страницы – 1000 записей.
* ”/preferences” – для получения всех избранных записей о приёмах пищи от пользователя, создания изменения и удаления их идентификатора брони, который практически невозможно подобрать;
* ”/preferences/{id}” – для удаления конкретной избранной записи о приёме пищи по её идентификатору.
* “/goals” – для получения, создания изменения и удаления данных о целях пользователя по макронутриентам на день.
* “/database” – для получения готовой базы данных блюд.

## Взаимодействие клиента и сервера

Клиент и сервер взаимодействуют между собой с помощью HTTP запросов. Запросы к серверу с клиентской части посылаются с помощью функции fetch. Формирование запроса на проверку того, вошёл ли пользователь в аккаунт или нет показано на рисунке 4.

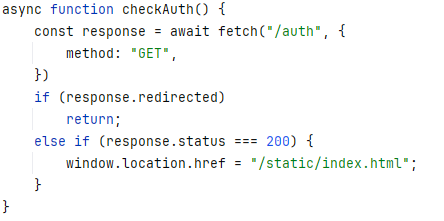


Рисунок 4. Формирование HTTP запроса к серверу.

Так, на первой строчке объявляется функция для проверки того, вошёл ли пользователь в аккаунт.

На второй строчке вызывается функция fetch и её результат присваивается константе response. Указывается адрес запрашиваемого ресурса.

В третьей строке указывается тип запроса - GET

В пятой и шестой строках, если сервер перенаправил запрос куда-либо, то выполнение функции завершается, перенаправление будет совершено автоматически.

Если же перенаправления не произошло, но пользователь авторизирован, то в седьмой и восьмой строчках клиент сам перенаправляется на страницу приложения.

Рассмотрим обработку запроса клиента по пути “/auth” со стороны сервера. Для обработки этого запроса используются возможности фреймворка, что показано на рисунке 5.

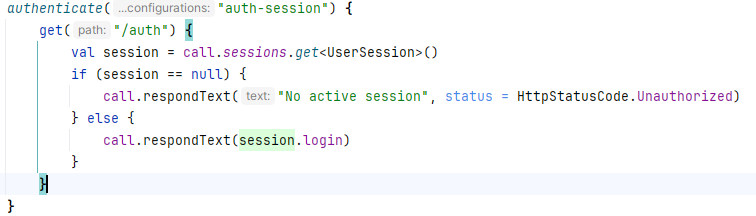


Рисунок 5. Обработка запроса на сервере

В первой строчке проверяется, имеет ли пользователь валидный cookie-токен с именем “auth-session”. Если же нет, запрос по этому адресу перенаправляется на “/static/login.html”, что показано на рисунке 6.



Рисунок 6. Проверка валидности сессии.

Функция authenticate на строке 4 проверяет, есть ли в базе данных запись о пользователе с такими логином и паролем. В случае успеха – сессия валидна. Иначе – нет и происходит переадресация на “/static/login/html”.

Вернёмся к рисунку 5. После проверки того, что у пользователя активна сессия, из неё сервер расшифровывает логин пользователя и возвращает его как ответ на запрос.

## Описание приложения

Приложение разработано для ведения журнала питания. Поставляется в виде .zip архива. После его запуска (см пункт 2.5.3) приложение будет доступно в сети интернет по ip-адресу того сервера, где оно было запущено. Инструкция пользователю указана в пункте 2.5.4. Функционал приложения описан в введении

Клиентская часть приложения составлена на языках HTML, CSS и JavaScript. С использованием среды разработки Visual Studio Code

Клиентская часть тестировалось на ОС Windows 11 x64 в браузерах Edge (Версия 125.0.2535.51) и Brave (1.66.113 Chromium: 125.0.6422.76) и на Samsung Galaxy A33 (ОС Android 12, Brave 1.66.113 Chromium: 125.0.6422.60)

Серверная часть тестировалась на OC Windows 10 x64, OC Windows 11 Home Single Language x64 23H2

* + 1. Описание базы данных

База данных имеет ключевую роль в работе приложения. При запросе клиентом репрезентации ресурса (записи о приёмах пищи, избранные блюда), сервер сформирует обращение в базу данных, рассчитывая на создание, чтение, обновление или удаление записи, согласно CRUD-модели.

В качестве системы управления базами данных выбрана MySQL.

В базе данных хранятся сущности четырёх типов:

1. Аккаунты пользователей в таблице “users”. Эта сущности имеет логин и пароль, хранящийся с захэшированном виде. Основной ключ - логин
2. Записи о приёмах пищи в таблице “incomes”. Эта сущность имеет уникальный идентификатор, логин пользователя, кому она принадлежит, а также JSON представление сущности.
3. Записи о избранных приёмах пищи в таблице “preferences”. Эта сущность имеет уникальный идентификатор, логин пользователя, кому она принадлежит, а также JSON представление сущности.
4. Записи о целях потребления макронутриентов за день пользователем в таблице “goals”. Эта сущность имеет уникальный идентификатор, логин пользователя, кому она принадлежит, а также JSON представление сущности.

Записи в базе данных хранятся согласно реляционной модели и структурированы по следующим таблицам:

* Таблица “users”:
  + login: логин пользователя
  + pass: Пароль пользователя в захэшированном виде
* Таблица ”incomes”:
  + id: уникальный идентификатор записи.
  + login: логин пользователя, сохранившего приём пищи.
  + json: json-представление приёма пищи.
* Таблица ”preferences”:
  + id: уникальный идентификатор записи.
  + login: логин пользователя, сохранившего приём пищи.
  + json: json-представление приёма пищи.
* Таблица ”goals”:
  + id: уникальный идентификатор записи.
  + login: логин пользователя, сохранившего цель по макронутриентам.
  + json: json-представление цели.

Связность таблиц обеспечивается foreign-ключами и ограничениями, не позволяющими создавать и оставлять произвольные записи:

* Таблица “incomes” связана с таблицей “users” по полю “login”.
* Таблица ”goals” связана с таблицей “users” по полю “login”.
* Таблица ”preferences” связана с таблицей “users” по полю “login”.

Описанная схема базы данных показана на рисунке 7.

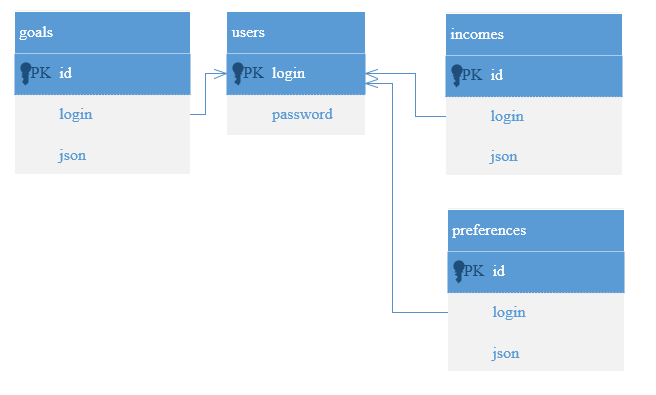


Рисунок 7. Схема базы данных.

* + 1. Инструкция запуска приложения.

Для запуска приложения в среде Microsoft Windows (x86-64) версии 10 и выше:

1. Выполните команду “.\gradlew build” из терминала в директории приложения.
2. Выполните команду “.\gradlew run” из терминала в директории приложения.
3. Приложение запущено по ip-адресу устройства, на котором оно запускалось и на порту номер 8080.
4. Основная страница приложения доступна по адресу http://localhost:8080/static/index.html
   * 1. Инструкция пользователю

После того как приложение будет запущено на сервере, пользователю необходимо ввести в адресной строке браузера ip-адрес сервера и URL запрашиваемой страницы. После этого пользователю будет доступна страница входа в приложение. Она показана на рисунке 8.



Рисунок 8. Страница входа в приложение.

После регистрации или входа пользователя перенаправит на основную страницу приложения. Она показана на рисунке 9.

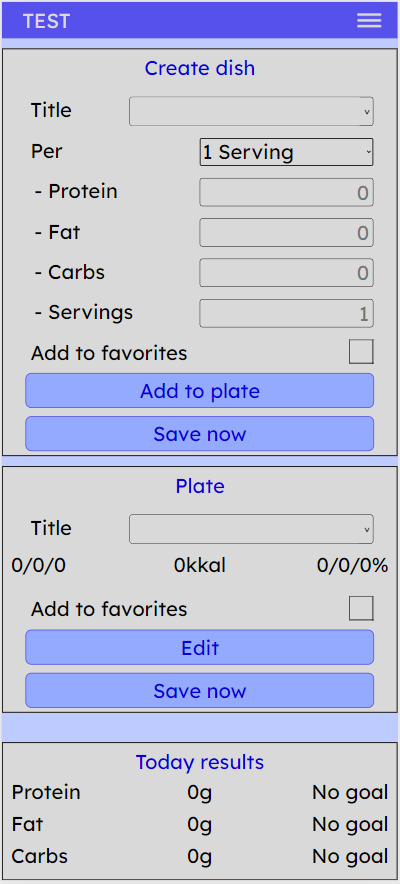


Рисунок 9. Основная страница приложения.

Вверху слева страницы указан логин авторизированного пользователя. На главной странице можно указать макронутриенты, единицы изменения блюда и его название в меню “Create dish”. При нажатии кнопки “Add to plate” Продукт добавится в «тарелку», в которой можно собрать итоговое блюдо из ингредиентов и сохранить его позже. Если будет нажата кнопка “Add to favorites”, то после сохранения записи о приёме пищи, это блюдо также будет сохранено в избранном. Ниже, чем “Plate” располагается статистика за лень по съеденным макронутриентам и профцентром соотношении от цели. При нажатии кнопки “Меню” Справа сверху, откроется меню приложения. Оно показано на рисунке 10.

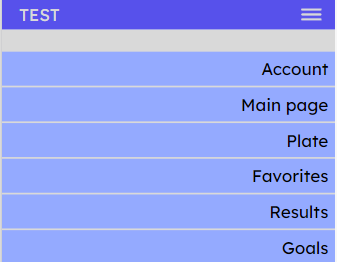


Рисунок 10. Меню приложения

Из меню можно перейти в подменю “Account”, где есть единственная кнопка – выйти из аккаунта. В окне “Plate” видно ингредиенты собираемого блюда. Их можно удалить. Также из подменю можно записать блюдо или часть его (например, 2/3 блюда было съедено). Также можно добавить блюдо в избранное аналогично, как и на главной странице. Окно “Plate” показано на рисунке 11.

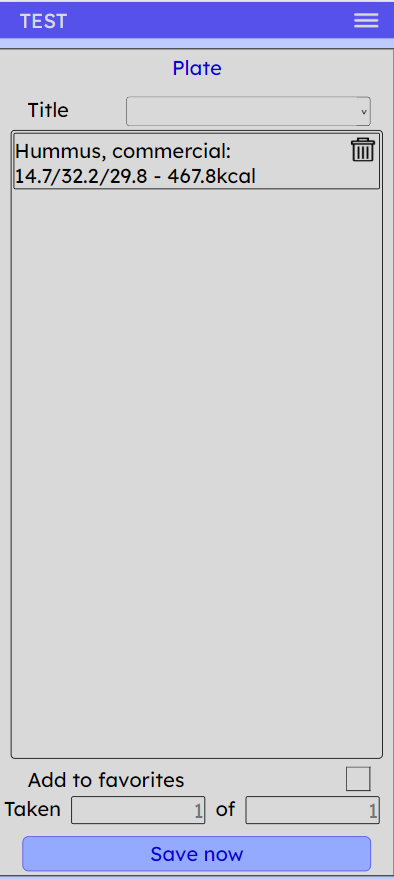


Рисунок 11. Окно “Plate”.

В окне “Favorites” пользователь может просмотреть и удалить свои избранные блюда. Это окно показано на рисунке 12.

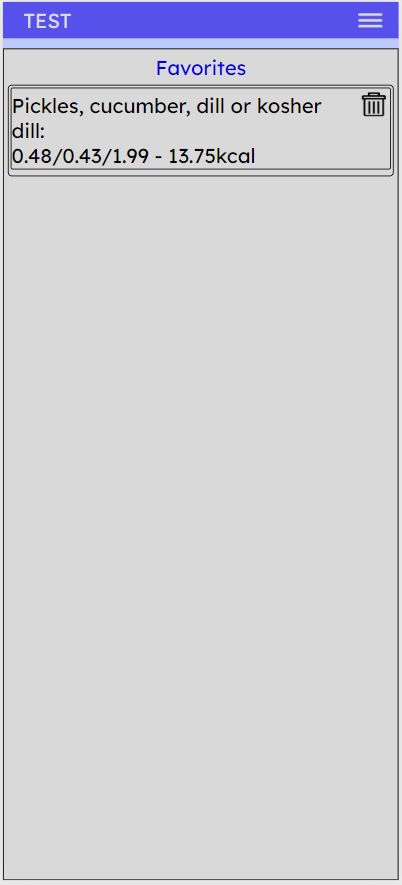


Рисунок 12. Окно “Favorites”.

В окне “Results” можно увидеть результаты за прошедшие 7 и 30 дней. А также сводку за выбранный день показано на рисунке 13.

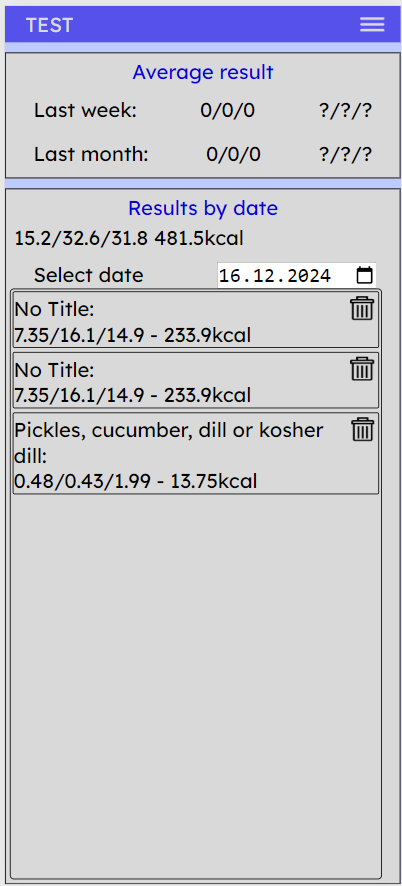


Рисунок 13. Окно “Results”.

В окне “Goals” можно просмотреть свою текущую цель по макронутриентам, а также рассчитать новую в соответствии с формулой Miffin St George. Ниже формы расчёта представлена справка по формуле. Окно “Goals” показано на рисунке 14.

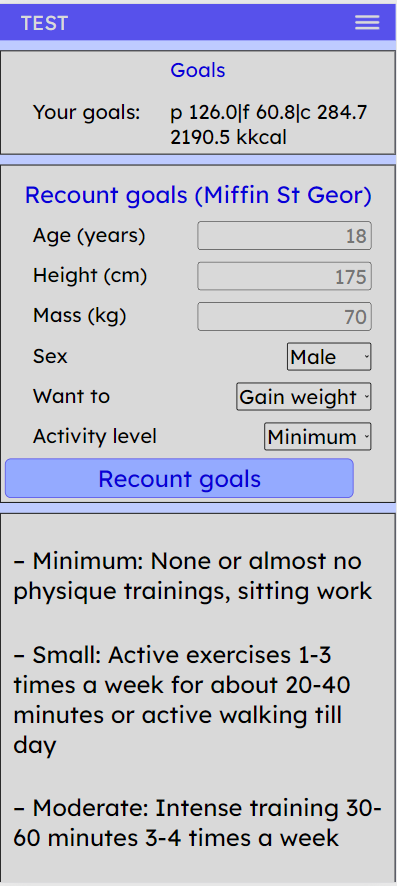


Рисунок 14. Окно “Goals”.

## Пути развития

Приложение можно дополнить, расширив его функционал, улучшив дизайн или оптимизировав логику работы. Возможные пути развития приложения:

* Интеграция с сервисами авторизации, как-то, ВКонтакте, Mail.ru.
* Добавление возможности отправлять изображения в дополнение к названию блюда
* Добавление перевода клиентской части на различные языки.
* Добавление форума в приложении для общения на темы питания
* Оптимизация формата хранимых данных, например сжатие
* Интеграция рекламы, платной версии.

Выводы

В процессе работы над приложением:

* Изучен и оценены существующие технологии разработки web-приложений. Был использован фреймворк Ktor.
* Рассмотрены существующие среды разработки веб-приложений. Их выбор основывался на личном удобстве, поэтому были выбраны привычные и изученные ранее среды разработки.
* Рассмотрена классификацию информационных систем и классифицирована разработанная информационная система
* Изучен протокол HTTP, языки HTML, CSS, JavaScript, рассмотрена архитектура веб-приложений, разработано приложение для ведения журнала питания с помощью указанных выше языков, а также Kotlin.

Список литературы

1. Дзен. «Толстый клиент и тонкий клиент: Различия, принцип работы, примеры применения, плюсы и минусы». URL: https://dzen.ru/a/Za5dUo\_5TGRr11P8 (дата обращения: 16.12.2024).

2]. Bootstrap. Домашняя страница. URL: https://getbootstrap.com/ (дата обращения: 16.12.2024).

3. React. Домашняя страница URL: https://ru.react.js.org/ (дата обращения: 16.12.2024).

4. jQuery. Домашняя страница URL: https://jquery.com/ (дата обращения: 16.12.2024).

5. Что такое CMS (Система Управления Контентом) URL: https://timeweb.com/ru/community/articles/chto-takoe-cms (дата обращения: 16.12.2024).

6. Ktor. Домашняя страница URL: https://ktor.io/ (дата обращения: 16.12.2024).

Приложение

Application.kt

package com.foodjournal  
  
import io.ktor.server.application.\*  
import io.ktor.server.auth.\*  
import io.ktor.server.response.\*  
import org.jetbrains.exposed.sql.Database  
  
fun main(args: Array<String>) {  
 io.ktor.server.netty.EngineMain.main(args)  
}  
  
fun Application.module() {  
 *//WHAT A HELL? I need to put it somewhere  
 //TODO: Replace it somewhere not here  
 install*(Authentication) **{** *session*<UserSession>("auth-session") **{** validate **{** session **->** if (*authenticate*(session.login, session.pass)) {  
 session  
 } else {  
 null  
 }  
 **}** challenge **{** call.respondRedirect("/static/login.html")  
 **}  
  
 }  
 }** *configureSerialization*()  
 *configureDatabases*()  
 *configureTemplating*()  
 *configureMonitoring*()  
 *configureHTTP*()  
 *configureSecurity*()  
 *configureRouting*()  
}

Databases.kt

package com.foodjournal  
  
import com.codahale.metrics.\*  
import io.ktor.http.\*  
import io.ktor.resources.\*  
import io.ktor.serialization.gson.\*  
import io.ktor.serialization.kotlinx.json.\*  
import io.ktor.server.application.\*  
import io.ktor.server.auth.\*  
import io.ktor.server.http.content.\*  
import io.ktor.server.metrics.dropwizard.\*  
import io.ktor.server.plugins.contentnegotiation.\*  
import io.ktor.server.plugins.cors.routing.\*  
import io.ktor.server.plugins.csrf.\*  
import io.ktor.server.plugins.statuspages.\*  
import io.ktor.server.request.\*  
import io.ktor.server.resources.\*  
import io.ktor.server.response.\*  
import io.ktor.server.routing.\*  
import io.ktor.server.sessions.\*  
import io.ktor.server.thymeleaf.*Thymeleaf*import io.ktor.server.thymeleaf.ThymeleafContent  
import java.util.concurrent.TimeUnit  
import kotlinx.serialization.Serializable  
import org.jetbrains.exposed.sql.\*  
import org.thymeleaf.templateresolver.ClassLoaderTemplateResolver  
  
@Serializable  
data class MyPassword(val pass: String)  
@Serializable  
data class MyJson(val json: String)  
@Serializable  
data class MyDateSpan(val dateStart: String, val dateEnd: String)  
  
fun Application.configureDatabases() {  
 val database = Database.connect(  
 url = "jdbc:mysql://localhost:3306/auth",  
 user = "root",  
 driver = "com.mysql.cj.jdbc.Driver",  
 password = "mysql",  
 )  
 */\*User management  
 /accounts <--- PUT/POST/DELETE ||| Change Password / Create Account / Delete Account  
 \*/* val userService = UserService(database)  
 routing **{** *//Create account* post("/accounts") **{** val user = call.receive<ExposedUser>()  
 try {  
 userService.create(user);  
 } catch (e: org.jetbrains.exposed.exceptions.ExposedSQLException) {  
 call.respondText("User already exists", status = HttpStatusCode.Conflict)  
 }  
 call.respondText("User created", status = HttpStatusCode.OK)  
 **}** *authenticate*("auth-session") **{** *//Change password* put("/accounts") **{** val pass = call.receive<MyPassword>().pass  
 val session = call.*sessions*.*get*<UserSession>()  
 if (session == null) {  
 call.respondText("No active session", status = HttpStatusCode.Unauthorized)  
 } else {  
 val login = session.login  
  
 val user = ExposedUser(login, pass);  
 userService.update(user);  
 }  
  
 **}** *//Delete account* delete("/accounts") **{** val session = call.*sessions*.*get*<UserSession>()  
 if (session == null) {  
 call.respondText("No active session", status = HttpStatusCode.Unauthorized)  
 } else {  
 val login = session.login  
 userService.delete(login);  
 }  
  
 **}  
  
 }  
 }** */\*  
 /incomes <--- GET/PUT/POST/DELETE  
 /preferences <--- GET/PUT/POST/DELETE  
 \*/* val incomesService = IncomesService(database)  
 routing **{** *authenticate*("auth-session") **{** get("/incomes") **{** val session = call.*sessions*.*get*<UserSession>()  
 if (session == null) {  
 call.respondText("No active session", status = HttpStatusCode.Unauthorized)  
 } else {  
 val login = session.login  
 call.respond(incomesService.read(login))  
 }  
 **}** get("/incomes/{page}") **{** val session = call.*sessions*.*get*<UserSession>()  
 if (session == null) {  
 call.respondText("No active session", status = HttpStatusCode.Unauthorized)  
 } else {  
 val page = call.parameters["page"]?.*toInt*() ?: throw IllegalArgumentException("Invalid Page")  
 val login = session.login  
 call.respond(incomesService.readPaged(login, page, 1000))  
 }  
 **}** delete("/incomes/{id}") **{** val session = call.*sessions*.*get*<UserSession>()  
 if (session == null) {  
 call.respondText("No active session", status = HttpStatusCode.Unauthorized)  
 } else {  
 val login = session.login  
 val id = call.parameters["id"]?.*toInt*() ?: throw IllegalArgumentException("Invalid ID")  
 incomesService.delete(id, login)  
 call.respondText("Deleted successfully", status = HttpStatusCode.OK)  
 }  
 **}** put("/incomes") **{** val session = call.*sessions*.*get*<UserSession>()  
 if (session == null) {  
 call.respondText("No active session", status = HttpStatusCode.Unauthorized)  
 } else {  
 val login = session.login  
 val income = call.receive<ExposedIncome>()  
 incomesService.update(ExposedIncome(income.id, login, income.json))  
 call.respondText("Updated successfully", status = HttpStatusCode.OK)  
 }  
 **}** post("/incomes") **{** val session = call.*sessions*.*get*<UserSession>()  
 if (session == null) {  
 call.respondText("No active session", status = HttpStatusCode.Unauthorized)  
 } else {  
 val json = call.receive<MyJson>().json  
 val income = InsertIncome(session.login, json)  
 val id = incomesService.create(income)  
 call.respondText(id.toString(), status = HttpStatusCode.OK)  
 }  
 **}  
  
 }  
 }** val preferencesService = PreferencesService(database)  
 routing **{** *authenticate*("auth-session") **{** get("/preferences") **{** val session = call.*sessions*.*get*<UserSession>()  
 if (session == null) {  
 call.respondText("No active session", status = HttpStatusCode.Unauthorized)  
 } else {  
 val login = session.login  
 call.respond(preferencesService.read(login))  
 }  
 **}** delete("/preferences/{id}") **{** val session = call.*sessions*.*get*<UserSession>()  
 if (session == null) {  
 call.respondText("No active session", status = HttpStatusCode.Unauthorized)  
 } else {  
 val login = session.login  
 val id = call.parameters["id"]?.*toInt*() ?: throw IllegalArgumentException("Invalid ID")  
 preferencesService.delete(id, login)  
  
 call.respondText("Deleted successfully", status = HttpStatusCode.OK)  
 }  
 **}** put("/preferences") **{** val session = call.*sessions*.*get*<UserSession>()  
 if (session == null) {  
 call.respondText("No active session", status = HttpStatusCode.Unauthorized)  
 } else {  
 val login = session.login  
 val preference = call.receive<ExposedPreference>()  
 preferencesService.update(ExposedPreference(preference.id, login, preference.json))  
 call.respondText("Updated successfully", status = HttpStatusCode.OK)  
 }  
 **}** post("/preferences") **{** val session = call.*sessions*.*get*<UserSession>()  
 if (session == null) {  
 call.respondText("No active session", status = HttpStatusCode.Unauthorized)  
 } else {  
 val json = call.receive<MyJson>().json  
 val preference = InsertPreference(session.login, json)  
 preferencesService.create(preference)  
 call.respondText("Created successfully", status = HttpStatusCode.OK)  
 }  
 **}  
  
 }  
 }** val goalsService = GoalsService(database)  
 routing **{** *authenticate*("auth-session") **{** get("/goals") **{** val session = call.*sessions*.*get*<UserSession>()  
 if (session == null) {  
 call.respondText("No active session", status = HttpStatusCode.Unauthorized)  
 } else {  
 val login = session.login  
 val goal = goalsService.read(login)  
 if (goal == null) {  
 call.respondText("No current goal", status = HttpStatusCode.NoContent)  
 }else {  
 call.respond(goal)  
 }  
 }  
 **}** post("/goals") **{** val session = call.*sessions*.*get*<UserSession>()  
 if (session == null) {  
 call.respondText("No active session", status = HttpStatusCode.Unauthorized)  
 } else {  
 val json = call.receive<MyJson>().json  
 val goal = InsertGoal(session.login, json)  
 goalsService.delete(session.login)  
 goalsService.create(goal)  
 call.respondText("Created successfully", status = HttpStatusCode.OK)  
 }  
 **}** delete("/goals") **{** val session = call.*sessions*.*get*<UserSession>()  
 if (session == null) {  
 call.respondText("No active session", status = HttpStatusCode.Unauthorized)  
 } else {  
 val login = session.login  
 goalsService.delete(login)  
 call.respondText("Deleted successfully", status = HttpStatusCode.OK)  
 }  
 **}** get("/database") **{** call.respondText("[{\"title\":\"Hummus, commercial\",\"protein\":7.35,\"fat\":16.1,\"carbs\":14.9},{\"title\":\"Tomatoes, grape, raw\",\"fat\":0.63,\"protein\":0.83,\"carbs\":5.51},{\"title\":\"Beans, snap, green, canned, regular pack, drained solids\",\"fat\":0.39,\"carbs\":4.11,\"protein\":1.04},{\"title\":\"Frankfurter, beef, unheated\",\"fat\":26,\"carbs\":2.89,\"protein\":11.7},{\"title\":\"Nuts, almonds, dry roasted, with salt added\",\"fat\":53.4,\"carbs\":16.2,\"protein\":20.4},{\"title\":\"Kale, raw\",\"protein\":2.92,\"fat\":1.49,\"carbs\":4.42},{\"title\":\"Egg, whole, raw, frozen, pasteurized\",\"protein\":12.3,\"fat\":10.3,\"carbs\":0.91},{\"title\":\"Egg, white, raw, frozen, pasteurized\",\"fat\":0.16,\"carbs\":0.74,\"protein\":10.1},{\"title\":\"Egg, white, dried\",\"protein\":79.9,\"fat\":0.65,\"carbs\":6.02},{\"title\":\"Onion rings, breaded, par fried, frozen, prepared, heated in oven\",\"fat\":12.6,\"carbs\":36.3,\"protein\":4.52},{\"title\":\"Pickles, cucumber, dill or kosher dill\",\"fat\":0.43,\"carbs\":1.99,\"protein\":0.48},{\"title\":\"Cheese, parmesan, grated\",\"fat\":24,\"carbs\":12.4,\"protein\":29.6},{\"title\":\"Cheese, pasteurized process, American, vitamin D fortified\",\"protein\":18,\"fat\":27.6,\"carbs\":5.27},{\"title\":\"Grapefruit juice, white, canned or bottled, unsweetened\",\"protein\":0.55,\"fat\":0.7,\"carbs\":7.59},{\"title\":\"Peaches, yellow, raw\",\"carbs\":10.1,\"protein\":0.91,\"fat\":0.27},{\"title\":\"Seeds, sunflower seed kernels, dry roasted, with salt added\",\"fat\":52.1,\"carbs\":17.1,\"protein\":21},{\"title\":\"Bread, white, commercially prepared\",\"fat\":3.45,\"carbs\":49.2,\"protein\":9.43},{\"title\":\"Kale, frozen, cooked, boiled, drained, without salt\",\"fat\":1.21,\"carbs\":5.3,\"protein\":2.94},{\"title\":\"Mustard, prepared, yellow\",\"protein\":4.25,\"fat\":3.76,\"carbs\":5.3},{\"title\":\"Kiwifruit, green, raw\",\"fat\":0.44,\"protein\":1.06,\"carbs\":14},{\"title\":\"Nectarines, raw\",\"carbs\":9.18,\"fat\":0.13,\"protein\":1.06},{\"title\":\"Cheese, cheddar\",\"carbs\":2.44,\"protein\":23.3,\"fat\":29},{\"title\":\"Cheese, cottage, lowfat, 2% milkfat\",\"fat\":1.87,\"carbs\":4.31,\"protein\":11},{\"title\":\"Cheese, mozzarella, low moisture, part-skim\",\"protein\":23.7,\"fat\":17.8,\"carbs\":4.44},{\"title\":\"Egg, whole, dried\",\"protein\":48.1,\"fat\":39.8,\"carbs\":1.87},{\"title\":\"Egg, yolk, raw, frozen, pasteurized\",\"fat\":25.1,\"carbs\":0.59,\"protein\":15.6},{\"title\":\"Egg, yolk, dried\",\"fat\":55.5,\"carbs\":1.07,\"protein\":34.2},{\"title\":\"Yogurt, Greek, plain, nonfat\",\"protein\":10.3,\"fat\":0.17,\"carbs\":3.64},{\"title\":\"Yogurt, Greek, strawberry, nonfat\",\"fat\":0.15,\"carbs\":12.2,\"protein\":8.06},{\"title\":\"Oil, coconut\",\"protein\":0,\"fat\":90.5,\"carbs\":0.84},{\"title\":\"Chicken, broilers or fryers, drumstick, meat only, cooked, braised\",\"protein\":23.9,\"fat\":5.23,\"carbs\":0},{\"title\":\"Chicken, broiler or fryers, breast, skinless, boneless, meat only, cooked, braised\",\"fat\":3.05,\"carbs\":0,\"protein\":32.1},{\"title\":\"Sauce, pasta, spaghetti/marinara, ready-to-serve\",\"fat\":1.05,\"carbs\":8.05,\"protein\":1.41},{\"title\":\"Ham, sliced, pre-packaged, deli meat (96%fat free, water added)\",\"protein\":16.7,\"fat\":3.15,\"carbs\":0.27},{\"title\":\"Olives, green, Manzanilla, stuffed with pimiento\",\"fat\":11.8,\"protein\":1.15,\"carbs\":4.96},{\"title\":\"Cookies, oatmeal, soft, with raisins\",\"fat\":13.7,\"protein\":5.79,\"carbs\":69.6},{\"title\":\"Tomatoes, canned, red, ripe, diced\",\"fat\":0.5,\"protein\":0.84,\"carbs\":3.32},{\"title\":\"Fish, haddock, raw\",\"fat\":0.32,\"protein\":16.3,\"carbs\":0},{\"title\":\"Fish, pollock, raw\",\"fat\":0.4,\"protein\":12.3,\"carbs\":0},{\"title\":\"Fish, tuna, light, canned in water, drained solids\",\"fat\":0.6,\"protein\":19,\"carbs\":0.08},{\"title\":\"Restaurant, Chinese, fried rice, without meat\",\"fat\":2.22,\"carbs\":32.5,\"protein\":3.84},{\"title\":\"Restaurant, Latino, tamale, pork\",\"protein\":7.38,\"fat\":7.98,\"carbs\":15.8},{\"title\":\"Restaurant, Latino, pupusas con frijoles (pupusas, bean)\",\"protein\":5.59,\"fat\":8.08,\"carbs\":31.5},{\"title\":\"Bread, whole-wheat, commercially prepared\",\"fat\":2.98,\"carbs\":43.1,\"protein\":12.3},{\"title\":\"Beef, loin, tenderloin roast, separable lean only, boneless, trimmed to 0 fat, select, cooked, roasted\",\"fat\":5.56,\"carbs\":0,\"protein\":27.7},{\"title\":\"Beef, loin, top loin steak, boneless, lip-on, separable lean only, trimmed to 1/8 fat, choice, raw\",\"fat\":5.93,\"carbs\":0,\"protein\":22.8},{\"title\":\"Beef, round, eye of round roast, boneless, separable lean only, trimmed to 0 fat, select, raw\",\"protein\":23.4,\"fat\":2.23,\"carbs\":0},{\"title\":\"Beef, round, top round roast, boneless, separable lean only, trimmed to 0 fat, select, raw\",\"fat\":2.14,\"carbs\":0,\"protein\":23.7},{\"title\":\"Beef, short loin, porterhouse steak, separable lean only, trimmed to 1/8 fat, select, raw\",\"fat\":4.5,\"protein\":22.7,\"carbs\":0},{\"title\":\"Beef, short loin, t-bone steak, bone-in, separable lean only, trimmed to 1/8 fat, choice, cooked, grilled\",\"protein\":27.3,\"fat\":10.5,\"carbs\":0},{\"title\":\"Carrots, frozen, unprepared\",\"fat\":0.33,\"protein\":0.81,\"carbs\":7.92},{\"title\":\"Cheese, dry white, queso seco\",\"protein\":24.5,\"fat\":21.4,\"carbs\":2.07},{\"title\":\"Cheese, ricotta, whole milk\",\"protein\":7.81,\"fat\":10.3,\"carbs\":6.86},{\"title\":\"Cheese, swiss\",\"fat\":27.6,\"carbs\":1.44,\"protein\":27},{\"title\":\"Figs, dried, uncooked\",\"carbs\":63.9,\"protein\":3.3,\"fat\":0.92},{\"title\":\"Lettuce, cos or romaine, raw\",\"carbs\":3.24,\"fat\":0.26,\"protein\":1.24},{\"title\":\"Melons, cantaloupe, raw\",\"carbs\":8.16,\"protein\":0.82,\"fat\":0.18},{\"title\":\"Oranges, raw, navels\",\"protein\":0.91,\"fat\":0.15,\"carbs\":11.8},{\"title\":\"Milk, lowfat, fluid, 1% milkfat, with added vitamin A and vitamin D\",\"protein\":3.38,\"fat\":0.85,\"carbs\":5.18},{\"title\":\"Pears, raw, bartlett\",\"protein\":0.38,\"fat\":0.16,\"carbs\":15.1},{\"title\":\"Restaurant, Chinese, sweet and sour pork\",\"protein\":8.88,\"fat\":13.3,\"carbs\":25.5},{\"title\":\"Salt, table, iodized\"},{\"title\":\"Milk, nonfat, fluid, with added vitamin A and vitamin D (fat free or skim)\",\"fat\":0.07,\"protein\":3.43,\"carbs\":4.92},{\"title\":\"Sauce, salsa, ready-to-serve\",\"protein\":1.44,\"carbs\":6.74,\"fat\":0.19},{\"title\":\"Milk, reduced fat, fluid, 2% milkfat, with added vitamin A and vitamin D\",\"protein\":3.36,\"carbs\":4.9,\"fat\":1.64},{\"title\":\"Sausage, breakfast sausage, beef, pre-cooked, unprepared\",\"fat\":27.1,\"carbs\":3.37,\"protein\":13.3},{\"title\":\"Sausage, Italian, pork, mild, cooked, pan-fried\",\"fat\":25.8,\"carbs\":2.15,\"protein\":18.2},{\"title\":\"Sausage, pork, chorizo, link or ground, cooked, pan-fried\",\"fat\":26,\"protein\":19.3,\"carbs\":2.63},{\"title\":\"Milk, whole, 3.25% milkfat, with added vitamin D\",\"fat\":2.77,\"carbs\":4.63,\"protein\":3.27},{\"title\":\"Sausage, turkey, breakfast links, mild, raw\",\"protein\":16.7,\"fat\":8.86,\"carbs\":0.93},{\"title\":\"Sugars, granulated\",\"protein\":0,\"fat\":0.32,\"carbs\":99.6},{\"title\":\"Turkey, ground, 93% lean, 7% fat, pan-broiled crumbles\",\"fat\":10.4,\"protein\":27.1,\"carbs\":0},{\"title\":\"Ham, sliced, restaurant\",\"fat\":3.54,\"protein\":19.6,\"carbs\":2.36},{\"title\":\"Cheese, American, restaurant\",\"fat\":26.6,\"protein\":17.5,\"carbs\":6.35},{\"title\":\"Beans, Dry, Medium Red (0% moisture)\",\"protein\":25.5,\"fat\":1.04},{\"title\":\"Beans, Dry, Red (0% moisture)\",\"protein\":21.3,\"fat\":1.16},{\"title\":\"Beans, Dry, Flor de Mayo (0% moisture)\",\"protein\":23.3,\"fat\":0.86},{\"title\":\"Beans, Dry, Brown (0% moisture)\",\"protein\":25.6,\"fat\":1.12},{\"title\":\"Beans, Dry, Tan (0% moisture)\",\"protein\":26.8,\"fat\":1.14},{\"title\":\"Beans, Dry, Light Tan (0% moisture)\",\"protein\":24.6,\"fat\":1.28},{\"title\":\"Beans, Dry, Carioca (0% moisture)\",\"protein\":25.2,\"fat\":1.44},{\"title\":\"Beans, Dry, Cranberry (0% moisture)\",\"protein\":24.4,\"fat\":1.23},{\"title\":\"Beans, Dry, Light Red Kidney (0% moisture)\",\"protein\":25,\"fat\":1.03},{\"title\":\"Beans, Dry, Pink (0% moisture)\",\"protein\":23.4,\"fat\":1.2},{\"title\":\"Beans, Dry, Dark Red Kidney (0% moisture)\",\"protein\":25.9,\"fat\":1.31},{\"title\":\"Beans, Dry, Navy (0% moisture)\",\"protein\":24.1,\"fat\":1.51},{\"title\":\"Beans, Dry, Small White (0% moisture)\",\"protein\":24.5,\"fat\":1.32},{\"title\":\"Beans, Dry, Small Red (0% moisture)\",\"protein\":23.5,\"fat\":1.28},{\"title\":\"Beans, Dry, Black (0% moisture)\",\"protein\":24.4,\"fat\":1.45},{\"title\":\"Beans, Dry, Pinto (0% moisture)\",\"protein\":23.7,\"fat\":1.24},{\"title\":\"Beans, Dry, Great Northern (0% moisture)\",\"protein\":24.7,\"fat\":1.24},{\"title\":\"Broccoli, raw\",\"carbs\":6.27,\"fat\":0.07,\"protein\":2.57},{\"title\":\"Ketchup, restaurant\",\"fat\":0.55,\"protein\":1.11,\"carbs\":26.8},{\"title\":\"Eggs, Grade A, Large, egg white\",\"protein\":10.7,\"fat\":0,\"carbs\":2.36},{\"title\":\"Eggs, Grade A, Large, egg yolk\",\"fat\":28.8,\"protein\":16.2,\"carbs\":1.02},{\"title\":\"Oil, canola\",\"fat\":94.5},{\"title\":\"Oil, corn\",\"fat\":94},{\"title\":\"Oil, soybean\",\"fat\":94.6},{\"title\":\"Oil, olive, extra virgin\",\"fat\":93.7},{\"title\":\"Eggs, Grade A, Large, egg whole\",\"protein\":12.4,\"fat\":8.65,\"carbs\":0.96},{\"title\":\"Pork, cured, bacon, cooked, restaurant\",\"fat\":34.6,\"protein\":40.9,\"carbs\":2.1},{\"title\":\"Butter, stick, unsalted\",\"fat\":81.5},{\"title\":\"Flour, wheat, all-purpose, enriched, bleached\",\"protein\":10.9,\"fat\":1.48,\"carbs\":77.3},{\"title\":\"Flour, wheat, all-purpose, enriched, unbleached\",\"fat\":1.48,\"carbs\":73.2,\"protein\":13.1},{\"title\":\"Flour, wheat, all-purpose, unenriched, unbleached\",\"fat\":1.7,\"carbs\":74.6,\"protein\":12},{\"title\":\"Flour, whole wheat, unenriched\",\"fat\":2.73,\"carbs\":71.2,\"protein\":15.1},{\"title\":\"Flour, bread, white, enriched, unbleached\",\"protein\":14.3,\"fat\":1.65,\"carbs\":72.8},{\"title\":\"Flour, rice, white, unenriched\",\"fat\":1.3,\"carbs\":79.8,\"protein\":6.94},{\"title\":\"Flour, corn, yellow, fine meal, enriched\",\"fat\":1.74,\"protein\":6.2,\"carbs\":80.8},{\"title\":\"Butter, stick, salted\",\"fat\":65},{\"title\":\"Onions, red, raw\",\"protein\":0.94,\"fat\":0.1,\"carbs\":9.93},{\"title\":\"Onions, yellow, raw\",\"protein\":0.83,\"fat\":0.05,\"carbs\":8.61},{\"title\":\"Garlic, raw\",\"fat\":0.38,\"protein\":6.62,\"carbs\":28.2},{\"title\":\"Flour, soy, defatted\",\"fat\":3.33,\"carbs\":32.9,\"protein\":51.1},{\"title\":\"Flour, soy, full-fat\",\"carbs\":27.9,\"protein\":38.6,\"fat\":20.7},{\"title\":\"Flour, rice, brown\",\"protein\":7.19,\"fat\":3.85,\"carbs\":75.5},{\"title\":\"Flour, rice, glutinous\",\"fat\":1.16,\"protein\":6.69,\"carbs\":80.1},{\"title\":\"Flour, pastry, unenriched, unbleached\",\"fat\":1.64,\"protein\":8.75,\"carbs\":77.2},{\"title\":\"Onions, white, raw\",\"fat\":0.13,\"protein\":0.89,\"carbs\":7.68},{\"title\":\"Bananas, overripe, raw\",\"fat\":0.22,\"protein\":0.73,\"carbs\":20.1},{\"title\":\"Bananas, ripe and slightly ripe, raw\",\"carbs\":23,\"protein\":0.74,\"fat\":0.29},{\"title\":\"Apples, red delicious, with skin, raw\",\"fat\":0.212,\"protein\":0.188,\"carbs\":14.8},{\"title\":\"Apples, fuji, with skin, raw\",\"fat\":0.162,\"protein\":0.148,\"carbs\":15.7},{\"title\":\"Apples, gala, with skin, raw\",\"fat\":0.15,\"protein\":0.133,\"carbs\":14.8},{\"title\":\"Apples, granny smith, with skin, raw\",\"fat\":0.138,\"protein\":0.266,\"carbs\":14.1},{\"title\":\"Apples, honeycrisp, with skin, raw\",\"fat\":0.1,\"protein\":0.102,\"carbs\":14.7},{\"title\":\"Oil, peanut\",\"fat\":93.4},{\"title\":\"Oil, sunflower\",\"fat\":93.2},{\"title\":\"Oil, safflower\",\"fat\":93.2},{\"title\":\"Oil, olive, extra light\",\"fat\":92.9},{\"title\":\"Mushroom, lion's mane\",\"fat\":0.256,\"protein\":2.5,\"carbs\":7.59},{\"title\":\"Mushroom, oyster\",\"fat\":0.188,\"protein\":2.9,\"carbs\":6.94},{\"title\":\"Mushrooms, shiitake\",\"fat\":0.195,\"protein\":2.41,\"carbs\":8.17},{\"title\":\"Mushrooms, white button\",\"fat\":0.371,\"protein\":2.89,\"carbs\":4.08},{\"title\":\"Soy milk, unsweetened, plain, shelf stable\",\"fat\":1.88,\"protein\":3.55,\"carbs\":1.29},{\"title\":\"Almond milk, unsweetened, plain, shelf stable\",\"fat\":1.11,\"protein\":0.555,\"carbs\":0.337},{\"title\":\"Spinach, baby\",\"fat\":0.619,\"protein\":2.85,\"carbs\":2.41},{\"title\":\"Spinach, mature\",\"fat\":0.604,\"protein\":2.91,\"carbs\":2.64},{\"title\":\"Tomato, roma\",\"fat\":0.425,\"protein\":0.696,\"carbs\":3.84},{\"title\":\"Flour, 00\",\"fat\":1.52,\"protein\":11.4,\"carbs\":74.4},{\"title\":\"Flour, spelt, whole grain\",\"fat\":2.54,\"protein\":14.5,\"carbs\":70.7},{\"title\":\"Flour, semolina, coarse and semi-coarse\",\"fat\":1.6,\"protein\":11.7,\"carbs\":73.8},{\"title\":\"Flour, semolina, fine\",\"fat\":1.84,\"protein\":13.3,\"carbs\":72},{\"title\":\"Apple juice, with added vitamin C, from concentrate, shelf stable\",\"fat\":0.286,\"protein\":0.086,\"carbs\":11.4},{\"title\":\"Orange juice, no pulp, not fortified, from concentrate, refrigerated\",\"fat\":0.325,\"protein\":0.734,\"carbs\":10.3},{\"title\":\"Grape juice, purple, with added vitamin C, from concentrate, shelf stable\",\"fat\":0.288,\"protein\":0.258,\"carbs\":15.6},{\"title\":\"Grape juice, white, with added vitamin C, from concentrate, shelf stable\",\"fat\":0.265,\"protein\":0.094,\"carbs\":15.8},{\"title\":\"Cranberry juice, not fortified, from concentrate, shelf stable\",\"fat\":0.338,\"protein\":0,\"carbs\":7.26},{\"title\":\"Grapefruit juice, red, not fortified, not from concentrate, refrigerated\",\"fat\":0.267,\"protein\":0.57,\"carbs\":9.1},{\"title\":\"Tomato juice, with added ingredients, from concentrate, shelf stable\",\"fat\":0.288,\"protein\":0.859,\"carbs\":4.32},{\"title\":\"Orange juice, no pulp, not fortified, not from concentrate, refrigerated\",\"fat\":0.356,\"protein\":0.812,\"carbs\":10},{\"title\":\"Mushroom, portabella\",\"fat\":0.312,\"protein\":2.75,\"carbs\":4.66},{\"title\":\"Mushroom, king oyster\",\"fat\":0.308,\"protein\":2.41,\"carbs\":8.5},{\"title\":\"Mushroom, enoki\",\"fat\":0.245,\"protein\":2.42,\"carbs\":8.14},{\"title\":\"Mushroom, crimini\",\"fat\":0.197,\"protein\":3.09,\"carbs\":4.01},{\"title\":\"Mushroom, maitake\",\"fat\":0.265,\"protein\":2.2,\"carbs\":6.6},{\"title\":\"Mushroom, beech\",\"fat\":0.449,\"protein\":2.18,\"carbs\":6.76},{\"title\":\"Mushroom, pioppini\",\"fat\":0.24,\"protein\":3.5,\"carbs\":5.76},{\"title\":\"Soy milk, sweetened, plain, refrigerated\",\"fat\":1.96,\"protein\":2.78,\"carbs\":3},{\"title\":\"Almond milk, unsweetened, plain, refrigerated\",\"fat\":1.56,\"protein\":0.656,\"carbs\":0.671},{\"title\":\"Oat milk, unsweetened, plain, refrigerated\",\"fat\":2.75,\"protein\":0.797,\"carbs\":5.1},{\"title\":\"Carrots, mature, raw\",\"fat\":0.351,\"protein\":0.941,\"carbs\":10.3},{\"title\":\"Carrots, baby, raw\",\"fat\":0.138,\"protein\":0.805,\"carbs\":9.08},{\"title\":\"Peppers, bell, green, raw\",\"fat\":0.106,\"protein\":0.715,\"carbs\":4.78},{\"title\":\"Peppers, bell, yellow, raw\",\"fat\":0.121,\"carbs\":6.6,\"protein\":0.819},{\"title\":\"Peppers, bell, red, raw\",\"fat\":0.126,\"carbs\":6.65,\"protein\":0.896},{\"title\":\"Peppers, bell, orange, raw\",\"fat\":0.156,\"protein\":0.882,\"carbs\":6.7},{\"title\":\"Buttermilk, low fat\",\"fat\":1.08,\"protein\":3.46,\"carbs\":4.81},{\"title\":\"Yogurt, plain, whole milk\",\"fat\":4.48,\"carbs\":5.57,\"protein\":3.82},{\"title\":\"Yogurt, Greek, plain, whole milk\",\"fat\":4.39,\"carbs\":4.75,\"protein\":8.78},{\"title\":\"Cheese, parmesan, grated, refrigerated\",\"fat\":29.5,\"carbs\":4.33,\"protein\":30.1},{\"title\":\"Cheese, feta, whole milk, crumbled\",\"fat\":19.1,\"carbs\":5.58,\"protein\":19.7},{\"title\":\"Flour, almond\",\"fat\":50.2,\"protein\":26.2,\"carbs\":16.2},{\"title\":\"Flour, oat, whole grain\",\"fat\":6.31,\"carbs\":69.9,\"protein\":13.2},{\"title\":\"Flour, potato\",\"fat\":0.951,\"carbs\":79.9,\"protein\":8.11},{\"title\":\"Peanut butter, creamy\",\"fat\":49.4,\"protein\":24,\"carbs\":22.7},{\"title\":\"Sesame butter, creamy\",\"fat\":62.4,\"carbs\":14.2,\"protein\":19.7},{\"title\":\"Almond butter, creamy\",\"fat\":53,\"carbs\":21.2,\"protein\":20.8},{\"title\":\"Flaxseed, ground\",\"fat\":37.3,\"carbs\":34.4,\"protein\":18},{\"title\":\"Cottage cheese, full fat, large or small curd\",\"fat\":4.22,\"protein\":11.6,\"carbs\":4.6},{\"title\":\"Cream cheese, full fat, block\",\"fat\":33.5,\"carbs\":4.56,\"protein\":5.79},{\"title\":\"Cream, heavy\",\"fat\":35.6,\"protein\":2.02,\"carbs\":3.8},{\"title\":\"Cream, sour, full fat\",\"fat\":18,\"protein\":3.07,\"carbs\":5.56},{\"title\":\"Lettuce, iceberg, raw\",\"fat\":0.074,\"protein\":0.742,\"carbs\":3.37},{\"title\":\"Lettuce, romaine, green, raw\",\"fat\":0.071,\"protein\":0.977,\"carbs\":4.06},{\"title\":\"Lettuce, leaf, red, raw\",\"fat\":0.106,\"protein\":0.883,\"carbs\":3.26},{\"title\":\"Lettuce, leaf, green, raw\",\"fat\":0.156,\"carbs\":4.07,\"protein\":1.09},{\"title\":\"Nuts, pine nuts, raw\",\"fat\":61.3,\"carbs\":18.6,\"protein\":15.7},{\"title\":\"Nuts, almonds, whole, raw\",\"fat\":51.1,\"carbs\":20,\"protein\":21.5},{\"title\":\"Nuts, walnuts, English, halves, raw\",\"fat\":69.7,\"carbs\":10.9,\"protein\":14.6},{\"title\":\"Nuts, pecans, halves, raw\",\"fat\":73.3,\"carbs\":12.7,\"protein\":9.96},{\"title\":\"Oats, whole grain, rolled, old fashioned\",\"fat\":5.89,\"carbs\":68.7,\"protein\":13.5},{\"title\":\"Oats, whole grain, steel cut\",\"fat\":5.8,\"carbs\":69.8,\"protein\":12.5},{\"title\":\"Pineapple, raw\",\"fat\":0.211,\"carbs\":14.1,\"protein\":0.461},{\"title\":\"Cherries, sweet, dark red, raw\",\"fat\":0.192,\"protein\":1.04,\"carbs\":16.2},{\"title\":\"Beans, snap, green, raw\",\"fat\":0.275,\"protein\":1.97,\"carbs\":7.41},{\"title\":\"Potatoes, russet, without skin, raw\",\"fat\":0.36,\"carbs\":17.8,\"protein\":2.27},{\"title\":\"Potatoes, red, without skin, raw\",\"fat\":0.248,\"protein\":2.06,\"carbs\":16.3},{\"title\":\"Potatoes, gold, without skin, raw\",\"fat\":0.264,\"protein\":1.81,\"carbs\":16},{\"title\":\"Sweet potatoes, orange flesh, without skin, raw\",\"fat\":0.375,\"carbs\":17.3,\"protein\":1.58},{\"title\":\"Celery, raw\",\"fat\":0.162,\"protein\":0.492,\"carbs\":3.32},{\"title\":\"Cucumber, with peel, raw\",\"fat\":0.178,\"protein\":0.625,\"carbs\":2.95},{\"title\":\"Cabbage, green, raw\",\"fat\":0.228,\"protein\":0.961,\"carbs\":6.38},{\"title\":\"Cabbage, red, raw\",\"fat\":0.214,\"protein\":1.24,\"carbs\":6.79},{\"title\":\"Strawberries, raw\",\"fat\":0.22,\"carbs\":7.96,\"protein\":0.641},{\"title\":\"Raspberries, raw\",\"fat\":0.188,\"carbs\":12.9,\"protein\":1.01},{\"title\":\"Blueberries, raw\",\"fat\":0.306,\"protein\":0.703,\"carbs\":14.6},{\"title\":\"Grapes, red, seedless, raw\",\"fat\":0.164,\"protein\":0.914,\"carbs\":20.2},{\"title\":\"Grapes, green, seedless, raw\",\"fat\":0.232,\"protein\":0.899,\"carbs\":18.6},{\"title\":\"Applesauce, unsweetened, with added vitamin C\",\"fat\":0.162,\"carbs\":12.3,\"protein\":0.273},{\"title\":\"Flour, amaranth\",\"fat\":6.24,\"protein\":13.2,\"carbs\":68.8},{\"title\":\"Flour, quinoa\",\"fat\":6.6,\"protein\":11.9,\"carbs\":69.5},{\"title\":\"Flour, sorghum\",\"fat\":3.59,\"protein\":8.27,\"carbs\":77.4},{\"title\":\"Flour, buckwheat\",\"fat\":2.48,\"protein\":8.88,\"carbs\":75},{\"title\":\"Flour, rye\",\"fat\":1.91,\"protein\":8.4,\"carbs\":77.2},{\"title\":\"Flour, barley\",\"fat\":2.45,\"protein\":8.72,\"carbs\":77.4},{\"title\":\"Flour, cassava\",\"fat\":0.494,\"protein\":0.918,\"carbs\":87.3},{\"title\":\"Buckwheat, whole grain\",\"fat\":3.04,\"protein\":11.1,\"carbs\":71.1},{\"title\":\"Millet, whole grain\",\"fat\":4.19,\"protein\":10,\"carbs\":74.4},{\"title\":\"Rice, brown, long grain, unenriched, raw\",\"fat\":3.31,\"protein\":7.25,\"carbs\":76.7},{\"title\":\"Rice, white, long grain, unenriched, raw\",\"fat\":1.03,\"carbs\":80.3,\"protein\":7.04},{\"title\":\"Beef, ground, 90% lean meat / 10% fat, raw\",\"fat\":12.8,\"protein\":18.2,\"carbs\":0},{\"title\":\"Beef, ground, 80% lean meat / 20% fat, raw\",\"fat\":19.4,\"protein\":17.5,\"carbs\":0},{\"title\":\"Pork, ground, raw\",\"fat\":17.5,\"protein\":17.8,\"carbs\":0},{\"title\":\"Chicken, ground, with additives, raw\",\"fat\":7.16,\"protein\":17.9,\"carbs\":0},{\"title\":\"Turkey, ground, 93% lean/ 7% fat, raw\",\"fat\":9.59,\"protein\":17.3,\"carbs\":0},{\"title\":\"Nuts, brazilnuts, raw\",\"fat\":57.4,\"protein\":15,\"carbs\":21.6},{\"title\":\"Nuts, cashew nuts, raw\",\"fat\":38.9,\"protein\":17.4,\"carbs\":36.3},{\"title\":\"Nuts, hazelnuts or filberts, raw\",\"fat\":53.5,\"protein\":13.5,\"carbs\":26.5},{\"title\":\"Peanuts, raw\",\"fat\":43.3,\"protein\":23.2,\"carbs\":26.5},{\"title\":\"Flour, chestnut\",\"fat\":4.64,\"protein\":5.29,\"carbs\":80.5},{\"title\":\"Nuts, macadamia nuts, raw\",\"fat\":64.9,\"protein\":7.79,\"carbs\":24.1},{\"title\":\"Nuts, pistachio nuts, raw\",\"fat\":45,\"protein\":20.5,\"carbs\":27.7},{\"title\":\"Seeds, pumpkin seeds (pepitas), raw\",\"fat\":40,\"protein\":29.9,\"carbs\":18.7},{\"title\":\"Seeds, sunflower seed, kernel, raw\",\"fat\":48.4,\"protein\":18.9,\"carbs\":24.5},{\"title\":\"Flour, coconut\",\"fat\":15.3,\"protein\":16.1,\"carbs\":58.9},{\"title\":\"Beans, cannellini, dry\",\"fat\":2.2,\"protein\":21.6,\"carbs\":59.8},{\"title\":\"Chickpeas, (garbanzo beans, bengal gram), dry\",\"fat\":6.27,\"carbs\":60.4,\"protein\":21.3},{\"title\":\"Lentils, dry\",\"fat\":1.92,\"protein\":23.6,\"carbs\":62.2},{\"title\":\"Blackeye pea, dry\",\"fat\":2.42,\"protein\":21.2,\"carbs\":61.8},{\"title\":\"Beans, black, canned, sodium added, drained and rinsed\",\"fat\":1.27,\"protein\":6.91,\"carbs\":19.8},{\"title\":\"Beans, navy, canned, sodium added, drained and rinsed\",\"fat\":1.4,\"protein\":6.57,\"carbs\":20},{\"title\":\"Beans, cannellini, canned, sodium added, drained and rinsed\",\"fat\":1.17,\"protein\":7.41,\"carbs\":18.8},{\"title\":\"Chickpeas (garbanzo beans, bengal gram), canned, sodium added, drained and rinsed\",\"fat\":3.1,\"protein\":7.02,\"carbs\":20.3},{\"title\":\"Beans, kidney, dark red, canned, sodium added, sugar added, drained and rinsed\",\"fat\":1.26,\"protein\":7.8,\"carbs\":21},{\"title\":\"Beans, kidney, light red, canned, sodium added, sugar added, drained and rinsed\",\"fat\":1.3,\"protein\":7.31,\"carbs\":21.4},{\"title\":\"Peas, green, sweet, canned, sodium added, sugar added, drained and rinsed\",\"fat\":1.15,\"protein\":4.73,\"carbs\":12.7},{\"title\":\"Beans, pinto, canned, sodium added, drained and rinsed\",\"fat\":1.27,\"carbs\":19.6,\"protein\":6.69},{\"title\":\"Blackeye pea, canned, sodium added, drained and rinsed\",\"fat\":1.3,\"carbs\":19.2,\"protein\":6.92},{\"title\":\"Beans, great northern, canned, sodium added, drained and rinsed\",\"fat\":1.27,\"carbs\":19.3,\"protein\":7.03},{\"title\":\"Pork, loin, boneless, raw\",\"fat\":9.47,\"protein\":21.1,\"carbs\":0},{\"title\":\"Pork, loin, tenderloin, boneless, raw\",\"fat\":3.9,\"protein\":21.6,\"carbs\":0},{\"title\":\"Chicken, breast, boneless, skinless, raw\",\"fat\":1.93,\"protein\":22.5,\"carbs\":0},{\"title\":\"Chicken, thigh, boneless, skinless, raw\",\"fat\":7.92,\"protein\":18.6,\"carbs\":0},{\"title\":\"Beef, ribeye, steak, boneless, choice, raw\",\"fat\":20,\"protein\":18.7,\"carbs\":0},{\"title\":\"Beef, round, top round, boneless, choice, raw\",\"fat\":5.7,\"protein\":21.5,\"carbs\":0.852},{\"title\":\"Beef, chuck, roast, boneless, choice, raw\",\"fat\":17.8,\"protein\":18.4,\"carbs\":0},{\"title\":\"Beef, flank, steak, boneless, choice, raw\",\"fat\":9.4,\"protein\":20.1,\"carbs\":0},{\"title\":\"Yogurt, plain, nonfat\",\"fat\":0.087,\"protein\":4.23,\"carbs\":8.08},{\"title\":\"Cheese, monterey jack, solid\",\"fat\":32.6,\"carbs\":1.9,\"protein\":22.6},{\"title\":\"Cheese, pasteurized process cheese food or product, American, singles\",\"fat\":23.9,\"carbs\":8.19,\"protein\":15.6},{\"title\":\"Cheese, provolone, sliced\",\"fat\":28.1,\"protein\":23.5,\"carbs\":2.45},{\"title\":\"Cheese, oaxaca, solid\",\"fat\":22.1,\"carbs\":2.4,\"protein\":22.1},{\"title\":\"Cheese, queso fresco, solid\",\"fat\":23.4,\"carbs\":2.96,\"protein\":18.9},{\"title\":\"Cheese, cotija, solid\",\"fat\":27.2,\"protein\":23.8,\"carbs\":2.72},{\"title\":\"Fish, salmon, sockeye, wild caught, raw\",\"fat\":4.94,\"protein\":22.3,\"carbs\":0},{\"title\":\"Fish, salmon, Atlantic, farm raised, raw\",\"fat\":13.1,\"protein\":20.3,\"carbs\":0},{\"title\":\"Fish, tilapia, farm raised, raw\",\"fat\":2.48,\"carbs\":0,\"protein\":19},{\"title\":\"Crustaceans, shrimp, farm raised, raw\",\"fat\":0.801,\"protein\":15.6,\"carbs\":0.485},{\"title\":\"Fish, cod, Atlantic, wild caught, raw\",\"fat\":0.668,\"protein\":16.1,\"carbs\":0},{\"title\":\"Fish, catfish, farm raised, raw\",\"fat\":7.31,\"protein\":16.5,\"carbs\":0},{\"title\":\"Crustaceans, crab, blue swimming, lump, pasteurized, refrigerated\",\"fat\":0.808,\"carbs\":0,\"protein\":18.6},{\"title\":\"Squash, summer, green, zucchini, includes skin, raw\",\"fat\":0.205,\"protein\":0.984},{\"title\":\"Squash, summer, yellow, includes skin, raw\",\"fat\":0.135,\"protein\":0.891},{\"title\":\"Squash, winter, butternut, raw\",\"fat\":0.168,\"protein\":1.15,\"carbs\":10.5},{\"title\":\"Squash, winter, acorn, raw\",\"fat\":0.182,\"protein\":1.25,\"carbs\":10.5},{\"title\":\"Cabbage, bok choy, raw\",\"fat\":0.234,\"protein\":1.02,\"carbs\":3.51},{\"title\":\"Cauliflower, raw\",\"fat\":0.238,\"protein\":1.64,\"carbs\":4.72},{\"title\":\"Collards, raw\",\"fat\":0.77,\"protein\":2.97,\"carbs\":7.02},{\"title\":\"Brussels sprouts, raw\",\"fat\":0.565,\"protein\":3.98,\"carbs\":9.62},{\"title\":\"Beets, raw\",\"fat\":0.302,\"carbs\":8.79,\"protein\":1.69},{\"title\":\"Eggplant, raw\",\"fat\":0.12,\"protein\":0.852,\"carbs\":5.4},{\"title\":\"Tomatoes, whole, canned, solids and liquids, with salt added\",\"fat\":0.206,\"protein\":0.868,\"carbs\":4.29},{\"title\":\"Tomato, sauce, canned, with salt added\",\"fat\":0.382,\"protein\":1.35,\"carbs\":6.33},{\"title\":\"Tomato, paste, canned, without salt added\",\"fat\":0.732,\"protein\":4.23,\"carbs\":20.2},{\"title\":\"Tomatoes, crushed, canned\",\"fat\":0.398,\"protein\":1.44,\"carbs\":7.14},{\"title\":\"Tomato, puree, canned\",\"fat\":0.265,\"protein\":1.58,\"carbs\":8.04},{\"title\":\"Apricot, with skin, raw\",\"fat\":0.405,\"protein\":0.961,\"carbs\":10.2},{\"title\":\"Melons, honeydew, raw\",\"fat\":0.216,\"protein\":0.531,\"carbs\":8.15},{\"title\":\"Plantains, ripe, raw\",\"fat\":0.893,\"protein\":1.16,\"carbs\":31},{\"title\":\"Plantains, underripe, raw\",\"fat\":0.685,\"protein\":1.23,\"carbs\":33.6},{\"title\":\"Chia seeds, dry, raw\",\"fat\":32.9,\"protein\":17,\"carbs\":38.3},{\"title\":\"Bulgur, dry, raw\",\"fat\":2.42,\"protein\":11.8,\"carbs\":75.9},{\"title\":\"Wild rice, dry, raw\",\"fat\":1.7,\"protein\":12.8,\"carbs\":75.7},{\"title\":\"Arugula, baby, raw\",\"fat\":0.325,\"protein\":1.65,\"carbs\":5.37},{\"title\":\"Asparagus, green, raw\",\"fat\":0.216,\"protein\":1.44,\"carbs\":5.1},{\"title\":\"Avocado, Hass, peeled, raw\",\"fat\":20.3,\"protein\":1.81,\"carbs\":8.32},{\"title\":\"Rice, black, unenriched, raw\",\"fat\":3.44,\"protein\":7.57,\"carbs\":77.2},{\"title\":\"Corn, sweet, yellow and white kernels,  fresh, raw\",\"fat\":1.63,\"protein\":2.79,\"carbs\":14.7},{\"title\":\"Einkorn, grain, dry, raw\",\"fat\":3.81,\"protein\":15.1,\"carbs\":68.7},{\"title\":\"Farro, pearled, dry, raw\",\"fat\":3.1,\"protein\":12.6,\"carbs\":72.1},{\"title\":\"Fonio, grain, dry, raw\",\"fat\":1.69,\"protein\":7.17,\"carbs\":81.3},{\"title\":\"Khorasan, grain, dry, raw\",\"fat\":2.8,\"protein\":14.8,\"carbs\":71.8},{\"title\":\"Kiwifruit (kiwi), green, peeled, raw\",\"fat\":0.64,\"protein\":1.01,\"carbs\":13.8},{\"title\":\"Mandarin, seedless, peeled, raw\",\"fat\":0.458,\"protein\":1.04,\"carbs\":13.4},{\"title\":\"Mango, Tommy Atkins, peeled, raw\",\"fat\":0.572,\"protein\":0.562,\"carbs\":15.3},{\"title\":\"Mango, Ataulfo, peeled, raw\",\"fat\":0.681,\"protein\":0.688,\"carbs\":17.4},{\"title\":\"Corn flour, masa harina, white or yellow, dry, raw\",\"fat\":4.34,\"protein\":7.56,\"carbs\":76.7},{\"title\":\"Pear, Anjou, green, with skin, raw\",\"fat\":0.371,\"protein\":0.312,\"carbs\":14.8},{\"title\":\"Plum, black, with skin, raw\",\"fat\":0.282,\"protein\":0.578,\"carbs\":13.5},{\"title\":\"Rice, red, unenriched, dry, raw\",\"fat\":3.44,\"protein\":8.56,\"carbs\":76.2},{\"title\":\"Sorghum bran, white, unenriched, dry, raw\",\"fat\":9.26,\"protein\":11.2,\"carbs\":68.7},{\"title\":\"Sorghum flour, white, pearled, unenriched, dry, raw\",\"fat\":3.24,\"protein\":10.2,\"carbs\":73.5},{\"title\":\"Sorghum grain, white, pearled, unenriched, dry, raw\",\"fat\":3.26,\"protein\":10.2,\"carbs\":74.9},{\"title\":\"Sorghum, whole grain, white, dry, raw\",\"fat\":4.22,\"carbs\":73.6,\"protein\":10.1},{\"title\":\"Plantains, overripe, raw\",\"fat\":0.99,\"protein\":1.17,\"carbs\":29.2}]")  
  
  
 **}  
 }  
  
 }**}

GoalsSchema.kt

package com.foodjournal  
  
import kotlinx.coroutines.Dispatchers  
import kotlinx.serialization.Serializable  
import org.jetbrains.exposed.sql.\*  
import org.jetbrains.exposed.sql.SqlExpressionBuilder.eq  
import org.jetbrains.exposed.sql.transactions.experimental.newSuspendedTransaction  
import org.jetbrains.exposed.sql.transactions.transaction  
  
@Serializable  
data class ExposedGoal(val login: String, val json: String)  
@Serializable  
data class InsertGoal(val login: String, val json: String)  
  
class GoalsService(database: Database) {  
 object Goals : Table() {  
 val login = varchar("login", length = 128).uniqueIndex().references(UserService.Users.login)  
 val json = text("json")  
  
 override val primaryKey = PrimaryKey(login) *// Set login as the primary key* }  
  
 init {  
 transaction(database) {  
 SchemaUtils.create(Goals)  
 }  
 }  
  
 suspend fun create(goal: InsertGoal): String = dbQuery {  
 Goals.insert {  
 it[login] = goal.login  
 it[json] = goal.json  
 }[Goals.login] *// Return the login as the identifier* }  
  
 suspend fun read(login: String): ExposedGoal? {  
 return dbQuery {  
 Goals.selectAll()  
 .where {Goals.login eq login }  
 .map { ExposedGoal(it[Goals.login], it[Goals.json]) }  
 .singleOrNull() *// Return a single goal or null if not found* }  
 }  
  
 suspend fun update(goal: ExposedGoal) {  
 dbQuery {  
 Goals.update({ Goals.login eq goal.login }) {  
 it[json] = goal.json  
 }  
 }  
 }  
  
 suspend fun delete(login: String) {  
 dbQuery {  
 Goals.deleteWhere { Goals.login eq login }  
 }  
 }  
  
 private suspend fun <T> dbQuery(block: suspend () -> T): T =  
 newSuspendedTransaction(Dispatchers.IO) { block() }  
}

HTTP.kt

package com.foodjournal  
  
import com.codahale.metrics.\*  
import io.ktor.http.\*  
import io.ktor.resources.\*  
import io.ktor.serialization.gson.\*  
import io.ktor.serialization.kotlinx.json.\*  
import io.ktor.server.application.\*  
import io.ktor.server.http.content.\*  
import io.ktor.server.metrics.dropwizard.\*  
import io.ktor.server.plugins.contentnegotiation.\*  
import io.ktor.server.plugins.cors.routing.\*  
import io.ktor.server.plugins.csrf.\*  
import io.ktor.server.plugins.statuspages.\*  
import io.ktor.server.request.\*  
import io.ktor.server.resources.\*  
import io.ktor.server.response.\*  
import io.ktor.server.routing.\*  
import io.ktor.server.thymeleaf.Thymeleaf  
import io.ktor.server.thymeleaf.ThymeleafContent  
import java.util.concurrent.TimeUnit  
import kotlinx.serialization.Serializable  
import org.jetbrains.exposed.sql.\*  
import org.thymeleaf.templateresolver.ClassLoaderTemplateResolver  
  
fun Application.configureHTTP() {  
 install(CORS) {  
 allowMethod(HttpMethod.Options)  
 allowMethod(HttpMethod.Put)  
 allowMethod(HttpMethod.Delete)  
 allowMethod(HttpMethod.Patch)  
 allowHeader(HttpHeaders.Authorization)  
 allowHeader("MyCustomHeader")  
 anyHost() *// @TODO: Don't do this in production if possible. Try to limit it.* }  
}

IncomesSchema.kt

package com.foodjournal  
  
import kotlinx.coroutines.Dispatchers  
import kotlinx.serialization.Serializable  
import org.jetbrains.exposed.sql.\*  
import org.jetbrains.exposed.sql.SqlExpressionBuilder.eq  
import org.jetbrains.exposed.sql.transactions.experimental.newSuspendedTransaction  
import org.jetbrains.exposed.sql.transactions.transaction  
  
@Serializable  
data class ExposedIncome(val id: Int, val login: String, val json: String)  
@Serializable  
data class InsertIncome(val login: String, val json: String)  
  
  
class IncomesService(database: Database) {  
 object Incomes : Table() {  
 val id = integer("id").*uniqueIndex*().*autoIncrement*()  
 val login = varchar("login", length = 128).*references*(UserService.Users.login)  
 val json = text("json")  
  
 override val primaryKey = PrimaryKey(id)  
 }  
  
 init {  
 *transaction*(database) **{** SchemaUtils.create(Incomes)  
 **}** }  
  
 suspend fun create(income: InsertIncome): Int = dbQuery **{** Incomes.*insert* **{  
 it**[login] = income.login  
 **it**[json] = income.json  
 **}**[Incomes.id]  
 **}** suspend fun read(login: String): List<ExposedIncome> {  
 return dbQuery **{** Incomes.*selectAll*()  
 .where **{** Incomes.login *eq* login **}** .*map* **{** ExposedIncome(**it**[Incomes.id], **it**[Incomes.login], **it**[Incomes.json]) **}  
 }** }  
 suspend fun readPaged(login: String, page: Int, pageSize: Int): List<ExposedIncome> {  
 return dbQuery **{** var offset = (page - 1) \* pageSize  
 Incomes.*selectAll*()  
 .where **{** Incomes.login *eq* login **}** .limit(pageSize).offset(offset.toLong())  
 .*map* **{** ExposedIncome(**it**[Incomes.id], **it**[Incomes.login], **it**[Incomes.json]) **}  
 }** }  
 */\*  
 suspend fun readByDateSpan(login: String, dateSpan: MyDateSpan): List<ExposedIncome> {  
 return dbQuery {  
 Incomes.selectAll()  
 .where { (Incomes.login eq login) and (dateSpan) }  
 .map { ExposedIncome(it[Incomes.id], it[Incomes.login], it[Incomes.json]) }  
 }  
 }  
  
 \*/  
 //Mess* suspend fun update(income: ExposedIncome) {  
 dbQuery **{** Incomes.*update*(**{** (Incomes.id *eq* income.id) *and* (Incomes.login *eq* income.login)**}**) **{  
 it**[json] = income.json  
 **}  
 }** }  
  
 suspend fun delete(id: Int, login: String) {  
 dbQuery **{** Incomes.*deleteWhere* **{** (Incomes.id *eq* id) *and* (Incomes.login *eq* login)**}  
 }** }  
  
 private suspend fun <T> dbQuery(block: suspend () -> T): T =  
 newSuspendedTransaction(Dispatchers.IO) **{** block() **}**}

Monitoring.kt

package com.foodjournal  
  
import com.codahale.metrics.\*  
import io.ktor.http.\*  
import io.ktor.resources.\*  
import io.ktor.serialization.gson.\*  
import io.ktor.serialization.kotlinx.json.\*  
import io.ktor.server.application.\*  
import io.ktor.server.http.content.\*  
import io.ktor.server.metrics.dropwizard.\*  
import io.ktor.server.plugins.contentnegotiation.\*  
import io.ktor.server.plugins.cors.routing.\*  
import io.ktor.server.plugins.csrf.\*  
import io.ktor.server.plugins.statuspages.\*  
import io.ktor.server.request.\*  
import io.ktor.server.resources.\*  
import io.ktor.server.response.\*  
import io.ktor.server.routing.\*  
import io.ktor.server.thymeleaf.Thymeleaf  
import io.ktor.server.thymeleaf.ThymeleafContent  
import java.util.concurrent.TimeUnit  
import kotlinx.serialization.Serializable  
import org.jetbrains.exposed.sql.\*  
import org.thymeleaf.templateresolver.ClassLoaderTemplateResolver  
  
fun Application.configureMonitoring() {  
 install(DropwizardMetrics) **{** Slf4jReporter.forRegistry(registry)  
 .outputTo(this@configureMonitoring.log)  
 .convertRatesTo(TimeUnit.SECONDS)  
 .convertDurationsTo(TimeUnit.MILLISECONDS)  
 .build()  
 .start(10, TimeUnit.SECONDS)  
 **}**}

PreferencesSchema.kt

package com.foodjournal  
  
import com.foodjournal.IncomesService.Incomes  
import kotlinx.coroutines.Dispatchers  
import kotlinx.serialization.Serializable  
import org.jetbrains.exposed.sql.\*  
import org.jetbrains.exposed.sql.SqlExpressionBuilder.eq  
import org.jetbrains.exposed.sql.transactions.experimental.newSuspendedTransaction  
import org.jetbrains.exposed.sql.transactions.transaction  
  
  
@Serializable  
data class ExposedPreference(val id: Int, val login: String, val json: String)  
data class InsertPreference(val login: String, val json: String)  
  
class PreferencesService(database: Database) {  
 object Preferences : Table() {  
 val id = integer("id").uniqueIndex().autoIncrement()  
 val login = varchar("login", length = 128).references(UserService.Users.login)  
 val json = text("json")  
  
 override val primaryKey = PrimaryKey(id)  
 }  
  
 init {  
 transaction(database) **{** SchemaUtils.create(Preferences)  
 **}** }  
  
 suspend fun create(preference: InsertPreference): Int = dbQuery **{** Preferences.insert **{** it[login] = preference.login  
 it[json] = preference.json  
 **}**[Preferences.id]  
 **}** suspend fun read(login: String): List<ExposedPreference> {  
 return dbQuery **{** Preferences.selectAll()  
 .where **{** Preferences.login eq login **}** .map **{** ExposedPreference(it[Preferences.id], it[Preferences.login], it[Preferences.json]) **}  
 }** }  
  
 suspend fun update(preference: ExposedPreference) {  
 dbQuery **{** Preferences.update(**{** (Preferences.id eq preference.id) and (Preferences.login eq preference.login)**}**) **{** it[json] = preference.json  
 **}  
 }** }  
  
 suspend fun delete(id: Int, login: String) {  
 dbQuery **{** Preferences.deleteWhere **{** (Preferences.id eq id) and (Preferences.login eq login)**}  
 }** }  
  
 private suspend fun <T> dbQuery(block: suspend () -> T): T =  
 newSuspendedTransaction(Dispatchers.IO) **{** block() **}**}

Routing.kt

package com.foodjournal  
  
import com.codahale.metrics.\*  
import io.ktor.http.\*  
import io.ktor.resources.\*  
import io.ktor.resources.Resources  
import io.ktor.serialization.gson.\*  
import io.ktor.serialization.kotlinx.json.\*  
import io.ktor.server.application.\*  
import io.ktor.server.http.content.\*  
import io.ktor.server.metrics.dropwizard.\*  
import io.ktor.server.plugins.contentnegotiation.\*  
import io.ktor.server.plugins.cors.routing.\*  
import io.ktor.server.plugins.csrf.\*  
import io.ktor.server.plugins.statuspages.\*  
import io.ktor.server.request.\*  
import io.ktor.server.resources.\*  
import io.ktor.server.response.\*  
import io.ktor.server.routing.\*  
import io.ktor.server.thymeleaf.Thymeleaf  
import io.ktor.server.thymeleaf.ThymeleafContent  
import java.util.concurrent.TimeUnit  
import kotlinx.serialization.Serializable  
import org.jetbrains.exposed.sql.\*  
import org.thymeleaf.templateresolver.ClassLoaderTemplateResolver  
import io.ktor.server.application.\*  
import io.ktor.server.auth.\*  
import io.ktor.server.sessions.\*  
import org.jetbrains.exposed.sql.\*  
import org.jetbrains.exposed.sql.transactions.transaction  
import org.mindrot.jbcrypt.BCrypt;  
import io.ktor.server.resources.\*  
import io.ktor.server.routing.\*  
  
fun Application.configureRouting() {  
 install(io.ktor.server.resources.Resources)  
  
 install(StatusPages) **{** exception<Throwable> **{** call, cause **->** call.respondText(text = "500: $cause" , status = HttpStatusCode.InternalServerError)  
 **}  
 }** routing **{** authenticate("auth-session") **{** get("/auth") **{** val session = call.sessions.get<UserSession>()  
 if (session == null) {  
 call.respondText("No active session", status = HttpStatusCode.Unauthorized)  
 } else {  
 call.respondText(session.login)  
 }  
 **}  
 }** staticResources("/static", "static")  
  
 *//Login to app* post("/login") **{** val user = call.receive<ExposedUser>()  
 if (authenticate(user.login, user.pass)) {  
 call.sessions.set(UserSession(user.login, user.pass))  
 val login = user.login  
 call.respondText("Logged in as $login")  
 } else {  
 call.respondText("Invalid credentials", status = HttpStatusCode.Unauthorized)  
 }  
 **}** *//logoff* delete("/login") **{** call.sessions.clear<UserSession>()  
 call.respondText("Logged out successfully", status = HttpStatusCode.OK)  
 **}  
 }**}

Security.kt

package com.foodjournal  
import io.ktor.server.application.\*  
import io.ktor.server.auth.\*  
import com.codahale.metrics.\*  
import io.ktor.http.\*  
import io.ktor.resources.\*  
import io.ktor.serialization.gson.\*  
import io.ktor.serialization.kotlinx.json.\*  
import io.ktor.server.application.\*  
import io.ktor.server.http.content.\*  
import io.ktor.server.metrics.dropwizard.\*  
import io.ktor.server.plugins.contentnegotiation.\*  
import io.ktor.server.plugins.cors.routing.\*  
import io.ktor.server.plugins.csrf.\*  
import io.ktor.server.plugins.statuspages.\*  
import io.ktor.server.request.\*  
import io.ktor.server.resources.\*  
import io.ktor.server.response.\*  
import io.ktor.server.routing.\*  
import io.ktor.server.thymeleaf.*Thymeleaf*import io.ktor.server.thymeleaf.ThymeleafContent  
import java.util.concurrent.TimeUnit  
import kotlinx.serialization.Serializable  
import org.jetbrains.exposed.sql.\*  
import org.thymeleaf.templateresolver.ClassLoaderTemplateResolver  
import io.ktor.server.sessions.\*  
import io.ktor.server.application.\*  
import io.ktor.server.auth.\*  
import org.jetbrains.exposed.sql.transactions.transaction  
import org.mindrot.jbcrypt.BCrypt  
  
@Serializable  
data class UserSession(val login: String, val pass: String)  
  
fun Application.configureSecurity() {  
  
 *install*(*CSRF*) **{** *// tests Origin is an expected value* allowOrigin("http://localhost:8080")  
  
 *// tests Origin matches Host header* originMatchesHost()  
  
 **}** *install*(*Sessions*) **{** *cookie*<UserSession>("user\_session") **{** cookie.path = "/"  
 cookie.maxAgeInSeconds = 60\*60\*24\*30  
 **}  
 }**}  
  
  
fun authenticate(login: String, pass: String): Boolean {  
 return *transaction* **{** UserService.Users.*selectAll*()  
 .where **{** UserService.Users.login *eq* login **}** .*map* **{ it**[UserService.Users.pass] **}** .*singleOrNull*()  
 ?.*let* **{** storedPassword **->** *verifyPassword*(pass, storedPassword)  
 **}** ?: false  
 **}**}  
fun hashPassword(password: String): String {  
 return BCrypt.hashpw(password, BCrypt.gensalt())  
}  
  
fun verifyPassword(plainPassword: String, hashedPassword: String): Boolean {  
 return BCrypt.checkpw(plainPassword, hashedPassword)  
}

Serialization.kt

package com.foodjournal  
  
import com.codahale.metrics.\*  
import io.ktor.http.\*  
import io.ktor.resources.\*  
import io.ktor.serialization.gson.\*  
import io.ktor.serialization.kotlinx.json.\*  
import io.ktor.server.application.\*  
import io.ktor.server.http.content.\*  
import io.ktor.server.metrics.dropwizard.\*  
import io.ktor.server.plugins.contentnegotiation.\*  
import io.ktor.server.plugins.cors.routing.\*  
import io.ktor.server.plugins.csrf.\*  
import io.ktor.server.plugins.statuspages.\*  
import io.ktor.server.request.\*  
import io.ktor.server.resources.\*  
import io.ktor.server.response.\*  
import io.ktor.server.routing.\*  
import io.ktor.server.thymeleaf.*Thymeleaf*import io.ktor.server.thymeleaf.ThymeleafContent  
import java.util.concurrent.TimeUnit  
import kotlinx.serialization.Serializable  
import org.jetbrains.exposed.sql.\*  
import org.thymeleaf.templateresolver.ClassLoaderTemplateResolver  
  
fun Application.configureSerialization() {  
 *install*(*ContentNegotiation*) **{** *gson* **{  
 }  
 }**}

Templating.kt

package com.foodjournal  
  
import com.codahale.metrics.\*  
import io.ktor.http.\*  
import io.ktor.resources.\*  
import io.ktor.serialization.gson.\*  
import io.ktor.serialization.kotlinx.json.\*  
import io.ktor.server.application.\*  
import io.ktor.server.http.content.\*  
import io.ktor.server.metrics.dropwizard.\*  
import io.ktor.server.plugins.contentnegotiation.\*  
import io.ktor.server.plugins.cors.routing.\*  
import io.ktor.server.plugins.csrf.\*  
import io.ktor.server.plugins.statuspages.\*  
import io.ktor.server.request.\*  
import io.ktor.server.resources.\*  
import io.ktor.server.response.\*  
import io.ktor.server.routing.\*  
import io.ktor.server.thymeleaf.Thymeleaf  
import io.ktor.server.thymeleaf.ThymeleafContent  
import java.util.concurrent.TimeUnit  
import kotlinx.serialization.Serializable  
import org.jetbrains.exposed.sql.\*  
import org.thymeleaf.templateresolver.ClassLoaderTemplateResolver  
  
fun Application.configureTemplating() {  
 install(Thymeleaf) {  
 setTemplateResolver(ClassLoaderTemplateResolver().apply {  
 prefix = "templates/thymeleaf/"  
 suffix = ".html"  
 characterEncoding = "utf-8"  
 })  
 }  
 routing {  
 get("/html-thymeleaf") {  
 call.respond(ThymeleafContent("index", mapOf("user" to ThymeleafUser(1, "user1"))))  
 }  
 }  
}  
data class ThymeleafUser(val id: Int, val name: String)

UsersSchema.kt

package com.foodjournal  
  
import kotlinx.coroutines.Dispatchers  
import kotlinx.serialization.Serializable  
import org.jetbrains.exposed.sql.\*  
import org.jetbrains.exposed.sql.SqlExpressionBuilder.eq  
import org.jetbrains.exposed.sql.transactions.experimental.newSuspendedTransaction  
import org.jetbrains.exposed.sql.transactions.transaction  
  
@Serializable  
data class ExposedUser(val login: String, val pass: String)  
  
class UserService(database: Database) {  
 object Users : Table() {  
 val login = varchar("login", length = 128).uniqueIndex()  
 val pass = varchar("pass", length = 128)  
  
 override val primaryKey = PrimaryKey(login)  
 }  
  
 init {  
 transaction(database) {  
 SchemaUtils.create(Users)  
 }  
 }  
  
 suspend fun create(user: ExposedUser) = dbQuery {  
 Users.insert {  
 it[pass] = hashPassword(user.pass)  
 it[login] = user.login  
 }  
 }  
  
 suspend fun read(login: String): ExposedUser? {  
 return dbQuery {  
 Users.selectAll()  
 .where { Users.login eq login }  
 .map { ExposedUser(it[Users.login], it[Users.pass]) }  
 .singleOrNull()  
 }  
 }  
  
 suspend fun update(user: ExposedUser) {  
 dbQuery {  
 Users.update({ Users.login eq user.login }) {  
 it[pass] = hashPassword(user.pass)  
 }  
 }  
 }  
  
  
 suspend fun delete(login: String) {  
 dbQuery {  
 Users.deleteWhere { Users.login eq login }  
 }  
 }  
  
 private suspend fun <T> dbQuery(block: suspend () -> T): T =  
 newSuspendedTransaction(Dispatchers.IO) { block() }  
}

Imdex.html

<html>  
 <head>  
 <title>Журнал питания</title>  
 <link rel="stylesheet" href="assets/css/index.css" />  
 <link rel="icon" type="image/x-icon" href="assets/images/favicon.ico">  
 </head>  
 <body style="font-family: 'Lexend Deca'; display: flex; flex-direction: column; height: 100%;">  
 <div id = "header", class = "header">  
 <div id = "headerDivisorManager" class = "headerDivisorManager">  
 <div id = "helperButtonWrapper" class = "helperButtonWrapper" onclick="forceClose();showMenu()">  
 <img class = "buttonImg", style = "width: 60px" src = "assets/images/menu.png">  
 </div>  
 <div id = "accountWrapper" class = "accountWrapper">  
 <div id = "usernameSign" class = "marginedText">Account  
 </div>  
 </div>  
 </div>  
 </div>  
 <div id = "mainWindow", class = "mainWindow">  
   
   
  
 <div class = "subBlock">  
 <div class = "verticalFlexer">  
 <center>  
 <div class = "title">  
 Create dish  
 </div>  
 </center>  
 <div class = "horizontalFlexer textinput margined">  
 <div class = "marginedText black">  
 Title  
 </div>  
 <div class = "select long" id = "sl3">  
 <input id = titi class = "select" onclick="manageClick('titi', 'dropdownTitleDish')" oninput = "showPrefs('dropdownTitleDish', 'titi')">  
 <img class = "buttonImg small titleselect" src = "assets/images/subinput.png">  
 </div>  
 </div>  
 <div class = "dropdown" style ="display:flex; flex-direction: column; align-items: flex-end; margin-right: 50px">  
 <div id="dropdownTitleDish" data-opened="0" class="dropdown-content" style = "display:none; flex-direction: column">  
 </div>  
 </div>  
 <div class = "horizontalFlexer textinput margined">  
 <div class = "marginedText black">  
 Per  
 </div>  
 <div class = "selectbox">  
 <select id = "selm" style = "width: 100%" onchange = "changedMeasurement()">  
 <option value="serv">1 Serving</option>  
 <option value="gram">100 g</option>  
 </select>  
 </div>  
 </div>  
 <div class = "horizontalFlexer textinput margined">  
 <div class = "marginedText black padded">  
 - Protein  
 </div>  
 <input id = "proi" class = "int", placeholder = "0", type = "number" >  
 </div>  
 <div class = "horizontalFlexer textinput margined">  
 <div class = "marginedText black padded">  
 - Fat  
 </div>  
 <input id = "fati" class = "int", placeholder = "0", type = "number" >  
 </div>  
 <div class = "horizontalFlexer textinput margined">  
 <div class = "marginedText black padded">  
 - Carbs  
 </div>  
 <input id = "cari" class = "int", placeholder = "0", type = "number" >  
 </div>  
 <div class = "horizontalFlexer textinput margined">  
 <div id = meastext class = "marginedText black padded">  
 - Servings  
 </div>  
 <input id = "amni" class = "int", placeholder = "1", pattern="">  
 </div>  
 <div class = "realDiv" style = "flex:1"></div>  
  
 <div class = "horizontalFlexer textinput margined">  
 <div class = "marginedText black">  
 Add to favorites  
 </div>  
   
 <button id="favI"class = "checkmark" data-selected="0" onclick="forceClose(); selectButton('favI')">  
 </button>  
 </div>  
 <center>  
 <div class = "button" onclick ="newIngridient()" >  
 Add to plate  
 </div>  
 </center>  
 <center>  
 <div class = "button" onclick = "saveMeal()">  
 Save now  
 </div>  
 </center>  
   
 </div>  
   
 </div>  
 <div class = "subBlock">  
 <div class = "verticalFlexer">  
 <center>  
 <div class = "title">  
 Plate  
 </div>  
 </center>  
 <div class = "horizontalFlexer textinput margined" style="margin-bottom:20px">  
 <div class = "marginedText black">  
 Title  
 </div>  
 <div class = "select long" id = "sl2">  
 <input id = "titlePlate" class = "select" onclick="manageClick('titlePlate', 'dropdownTitleDish2', 'notNull')" oninput = "showPrefs('dropdownTitleDish2', 'titlePlate', 'notNull')">  
  
  
 <img class = "buttonImg small titleselect" src = "assets/images/subinput.png">  
 </div>  
 </div>  
 <div class = "dropdown" style ="display:flex; flex-direction: column; align-items: flex-end; margin-right: 50px">  
 <div id="dropdownTitleDish2" data-opened="0" class="dropdown-content" style = "display:none; flex-direction: column">  
 </div>  
 </div>  
 <div class = "horizontalFlexer flexed">  
 <div id = mealstat class = "marginedText black" style="flex:1">  
 0/0/0  
 </div>  
 <div id = mealkkal class = "marginedText black"style="flex:1; text-align: center">  
 0kkal  
 </div>  
 <div id = mealprecent class = "marginedText black"style="flex:1; text-align: right">  
 0/0/0%  
 </div>  
 </div>  
 <div class = "horizontalFlexer textinput margined">  
 <div class = "marginedText black">  
 Add to favorites  
 </div>  
   
 <button id="favD"class = "checkmark" data-selected="0" onclick="forceClose();selectButton('favD')">  
 </button>  
 </div>  
 <center>  
 <div class = "button" onclick = "closeExcept('plate')">  
 Edit  
 </div>  
 </center>  
 <center>  
 <div class = "button" onclick = "saveFullPlate()">  
 Save now  
 </div>  
 </center>  
   
 </div>  
   
 </div>  
 <div style = "flex:1"></div>  
 <div class = "subBlock">  
 <div class = "verticalFlexer">  
 <div class = "title">Today results</div>  
 <div class = "horizontalFlexer flexed">  
 <div class = "marginedText black" style="flex:1">  
 Protein  
 </div>  
 <div id = "todayProt" class = "marginedText black"style="flex:1; text-align: center">  
  
 </div>  
 <div id = "todayProtPercent" class = "marginedText black"style="flex:1; text-align: right">  
  
 </div>  
 </div>  
 <div class = "horizontalFlexer flexed">  
 <div class = "marginedText black"style="flex:1">  
 Fat  
 </div>  
 <div id = "todayFat" class = "marginedText black"style="flex:1; text-align: center">  
  
 </div>  
 <div id = "todayFatPercent" class = "marginedText black"style="flex:1; text-align: right">  
  
 </div>  
 </div>  
 <div class = "horizontalFlexer flexed">  
 <div class = "marginedText black"style="flex:1">  
 Carbs  
 </div>  
 <div id = "todayCarbs" class = "marginedText black"style="flex:1; text-align: center">  
  
 </div>  
 <div id = "todayCarbsPercent" class = "marginedText black"style="flex:1; text-align: right">  
  
 </div>  
 </div>  
   
 </div>  
   
 </div>  
   
 </div>  
 <div id = "plate" style="display:none; margin-bottom: 10px !important;" class = "mainWindow" >  
 <div class = "subBlock" style = "flex:1">  
 <div class = "verticalFlexer spaceBetween center">  
 <div class = "title">  
 Plate  
 </div>  
 <div class = "horizontalFlexer textinput margined">  
 <div class = "marginedText black">  
 Title  
 </div>  
 <div class = "select long" id = "sl1">  
 <input id = "plateTitle2" class = "select" onclick="manageClick('plateTitle2', 'dropdownTitleDish3', 'notNull')" oninput = "showPrefs('dropdownTitleDish3', 'plateTitle2, 'notNull')">  
 <img class = "buttonImg small titleselect" src = "assets/images/subinput.png">  
 </div>  
 </div>  
 <div class = "dropdown" style ="display:flex; flex-direction: column; align-items: flex-end; margin-right: 50px">  
 <div id="dropdownTitleDish3" data-opened="0" class="dropdown-content" style = "display:none; flex-direction: column">  
 </div>  
 </div>  
 <div class = "realDiv"></div>  
 <div id = "plateBox" class = "scrollingBox">  
 </div>  
 <div class = "realDiv"></div><div class = "realDiv"></div>  
 <div class = "horizontalFlexer textinput">  
 <div class = "marginedText black">  
 Add to favorites  
 </div>  
 <button id="favP"class = "checkmark" data-selected="0" onclick="forceClose();selectButton('favP')">  
 </button>  
 </div>  
 <div class = "realDiv"></div>  
 <div class = "horizontalFlexer">  
 Taken  
 <input class = "select" id = "parts1" placeholder="1" type = "number" style = "text-align: right; margin-left: 25px; margin-right: 25px;border-radius: 5px; border: solid 2px black;">  
 of  
 <input class = "select" id = "parts2" placeholder="1" type = "number" style = "text-align: right; margin-left: 25px; margin-right: 25px;border-radius: 5px; border: solid 2px black;">  
 </div>  
 <div class = "realDiv"></div>  
 <div class = "button" onclick = "savePlate()">  
 Save now  
 </div>  
 </div>  
   
 </div>  
 </div>   
 <div id = "menu", class = "menu", data-opened = "0", style="display:none">  
 <div class = "verticalFlexer">  
 <div class = menuOption onclick = "closeExcept('account')">  
 Account  
 </div>  
 <div class = menuOption onclick = "closeExcept('mainWindow')">  
 Main page  
 </div>  
 <div class = menuOption onclick = "closeExcept('plate')">  
 Plate   
 </div>  
 <div class = menuOption onclick = "closeExcept('favorites')">  
 Favorites  
 </div>  
 <div class = menuOption onclick = "closeExcept('results')">  
 Results  
 </div>  
 <div class = menuOption onclick = "closeExcept('goals')">  
 Goals  
 </div>  
 </div>  
 </div>  
 <div id = "favorites", class = "mainWindow" style = "display:none">  
   
   
  
 <div class = "subBlock" style = "height: 100%;">  
 <div class = "title">  
 Favorites  
 </div>  
 <div id = "favBox" class = "scrollingBox" style = "height: auto;width: 100%; box-sizing: border-box;">  
 </div>  
 </div>  
   
 </div>  
 <div id = "account", class = "mainWindow" style = "display:none">  
  
 <div class = "button" onclick = "logoff()">Выйти</div>  
 </div>  
  
 <div id ="results" class = "mainWindow" style = "display:none">  
  
  
 <div class = "subBlock">  
 <div class="title">  
 Average result  
 </div>  
 <div class = "horizontalFlexer textinput margined">  
 <div class = "marginedText black">  
 Last week:  
 </div>  
 <div id = "weekdata" class = "marginedText black" style = "overflow: scroll">  
 ?/?/?  
 </div>  
 <div id = "weekdata2" class = "marginedText black" style = "overflow: scroll">  
 ?/?/?  
 </div>  
 </div>  
 <div class = "horizontalFlexer textinput margined">  
 <div class = "marginedText black">  
 Last month:  
 </div>  
 <div id = "monthdata" class = "marginedText black" style = "overflow: scroll">  
 ?/?/?  
 </div>  
 <div id = "monthdata2" class = "marginedText black" style = "overflow: scroll">  
 ?/?/?  
 </div>  
 </div>  
 </div>  
 <div class = "subBlock" style="flex:1;display: flex;flex-direction: column;">  
 <div class="title">  
 Results by date  
 </div>  
 <div id = "dateres" class = "marginedText black">  
 No date selected  
 </div>  
 <div class = "horizontalFlexer textinput margined">  
 <div class = "marginedText black">  
 Select date  
 </div>  
 <input id = "resultDate" type = "date" oninput="dateSelected()" style = "font-size: 50px">  
 </div>  
 <div id = "dateBox" class = "scrollingBox">  
 </div>  
 </div>  
 </div>  
 <div id ="goals" class = "mainWindow" style = "display:none">  
 <div class = "subBlock">  
 <div class = "verticalFlexer spaceBetween center">  
 <div class = "title">  
 Goals  
 </div>  
 <div class = "horizontalFlexer textinput margined">  
 <div class = "marginedText black padded">  
 Your goals:  
 </div>  
 <div class = "marginedText black padded" id = "goalsDiv">  
 No current goal  
 </div>  
 </div>  
 </div>  
 </div>  
 <div class = "subBlock">  
 <div class = "realDiv"></div><div class = "realDiv"></div>  
 <div class = "title">  
 Recount goals (Miffin St Geor)  
 </div>  
 <div class = "horizontalFlexer textinput margined">  
 <div class = "marginedText black padded">  
 Age (years)  
 </div>  
 <input id = "age" class = "int", placeholder = "18", type = "number" >  
 </div>  
 <div class = "horizontalFlexer textinput margined">  
 <div class = "marginedText black padded">  
 Height (cm)  
 </div>  
 <input id = "height" class = "int", placeholder = "175", type = "number" >  
 </div>  
 <div class = "horizontalFlexer textinput margined">  
 <div class = "marginedText black padded">  
 Mass (kg)  
 </div>  
 <input id = "mass" class = "int", placeholder = "70", type = "number" >  
 </div>  
 <div class = "horizontalFlexer textinput margined">  
 <div class = "marginedText black padded">  
 Sex  
 </div>  
 <select name="sex" id="sex">  
 <option value="male">Male</option>  
 <option value="female">Female</option>  
 </select>  
 </div>  
 <div class = "horizontalFlexer textinput margined">  
 <div class = "marginedText black padded">  
 Want to  
 </div>  
 <select name="desire" id="desire">  
 <option value="gain">Gain weight</option>  
 <option value="keep">Keep weight</option>  
 <option value="lose">Lose weight</option>  
 </select>  
 </div>  
 <div class = "horizontalFlexer textinput margined">  
 <div class = "marginedText black padded">  
 Activity level  
 </div>  
 <select name="activity" id="activity">  
 <option value="1.2">Minimum</option>  
 <option value="1.4">Small</option>  
 <option value="1.5">Moderate</option>  
 <option value="1.7">Hard</option>  
 <option value="1.9">Extreme</option>  
 </select>  
 </div>  
 <div class = "button" onclick = "recountGoals()">  
 Recount goals  
 </div>  
 </div>  
 <div class = "subBlock">  
 <div class = "marginedText black padded">  
 <br> – Minimum: None or almost no physique trainings, sitting work<div class="realDiv"></div>  
 <br> – Small: Active exercises 1-3 times a week for about 20-40 minutes or active walking till day<div class="realDiv"></div>  
 <br> – Moderate: Intense training 30-60 minutes 3-4 times a week<div class="realDiv"></div>  
 <br> – Hard: Heavy work or active sport 5-7 times a week<div class="realDiv"></div>  
 <br> – Extreme: More than 7 trainings a week or extremely hard work<div class="realDiv"></div>  
  
 </div>  
 </div>  
 </div>  
<script src="assets/js/index.js"></script>  
 </body>  
</html>

login.html

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <title>Журнал питания</title>  
 <link rel="stylesheet" href="assets/css/index.css" />  
 <link rel="icon" type="image/x-icon" href="assets/images/favicon.ico">  
</head>  
<body style="font-family: 'Lexend Deca'; display: flex; flex-direction: column; height: 100%;">  
 <center>  
 FoodJournal. Please login.  
 </center>  
 <div class = "subBlock" style = "height: 300px; display:flex; flex-direction:column; align-items: center; justify-content: space-evenly">  
 <input class = "select2" id = "usernameRegisterLine" placeholder = "Login">  
 <input class = "select2" id = "passwordRegisterLine" placeholder = "Password" type = "password">  
 <center>  
 <div class = "button" onclick = "register()" style ="padding: 5px 10px 5px 10px">Register</div>  
 </center>  
 </div>  
 <div class = "subBlock" style = "height: 300px; display:flex; flex-direction:column; align-items: center; justify-content: space-evenly">  
 <input class = "select2" id="usernameLoginLine" placeholder = "Login">  
 <input class = "select2" id="passwordLoginLine" placeholder = "Password" type = "password">  
 <center>  
 <div class = "button" onclick = "login()" style ="padding: 5px 10px 5px 10px">Login</div>  
 </center>  
 </div>  
  
 </div>  
 <div id = "msg">  
  
 </div>  
 <script src="assets/js/login.js"></script>  
</body>  
  
</html>

Index.css

@font-face {  
 font-family: "Lexend Deca";  
 src: url("../fonts/LexendDeca-Regular.ttf");  
 }  
  
body {  
 margin: 0px;  
 background-color: #BECBFF;  
 font-size: 50px;  
}  
div.mainWindow {  
 display: flex;  
 flex-direction: column;  
 width: 100%;  
 align-items: center;  
 flex:1;  
}  
div.header {  
 background-color: #5751eb;   
 height: 90px;  
 min-height: 90px;  
 width: 100vw;  
 max-width: 100%;  
 padding: 5px;  
 padding-left: 40px;  
 padding-right: 40px;  
 box-sizing: border-box;  
}  
div.headerDivisorManager {  
 display: flex;  
 flex-direction: row-reverse;  
 justify-content: space-between;  
 height: 100%;  
}  
div.helperButtonWrapper {  
 display: flex;  
 align-items: center;  
 justify-content: flex-end;  
 width: 100px;  
}  
div.accountWrapper {  
 display: flex;  
 align-items: center;  
}  
img.buttonImg {  
 width: 35px;  
 height: 35px;  
}  
img.buttonimg.small {  
 width: 15px;  
 height: 15px;  
}  
img.buttonimg.titleselect {  
 margin: 7px  
}  
div.marginedText {  
 margin-left: 10px;  
 margin-right: 10px;  
 font-size: 50px;  
 color: #D9D9D9;  
}  
div.marginedText.blue {  
 color: #0700D7  
}  
div.marginedText.black {  
 color: #000000  
}  
div.marginedText.black.padded {  
 padding-left: 10px;  
}  
div.selectbox {  
 width: 50%;  
}  
div.mainBlock {  
 display: flex;  
 flex-direction: row;  
 padding: 15px;  
 height: 100%;  
 justify-content: space-between;  
}  
div.subBlock {  
 margin-top: 25px;  
 width: 100%;  
 height: max-content;  
 padding: 10px;  
 padding-top: 5px;  
 background-color: #D9D9D9;  
 border: 2px solid #000000;  
 box-sizing: border-box;  
}  
div.subBlock.sub {  
 width: auto;  
 height: 45%;  
}  
div.verticalFlexer {  
 display: flex;  
 flex-direction: column;  
 height: 100%;  
}  
div.verticalFlexer.center {  
 align-items: center;  
}  
div.verticalFlexer.bloksHolder {  
 justify-content: space-between;  
 height: 95%;  
 width: 29vw;  
}  
div.verticalFlexer.spaceBetween {  
 justify-content: space-between;  
}  
div.verticalFlexer.spaceBetween.center {  
 justify-content: space-between;  
 align-items: center;  
}  
div.horizontalFlexer {  
 display: flex;  
 flex-direction: row;  
}  
div.horizontalFlexer.flexed {  
 flex: 1;  
 width: 100%;  
 justify-content: space-between;  
  
 overflow: scroll  
}  
div.horizontalFlexer.textinput {  
 justify-content: space-between;  
 padding-left: 5%;  
 padding-right: 5%;  
 height: min-content;  
 width: 90%;  
}  
div.horizontalFlexer.textinput.margined {  
 margin-top: 30px;  
}  
div.button {  
 margin-top: 20px;  
 text-align: center;  
 background-color: #94AAFF;  
 font-size: 50px;  
 color: #0700D7;  
 border-radius: 15px;  
 border: 2px solid #5651EA;  
 width: 90%;  
 padding: 10 0 10 0;  
 -webkit-border-radius: 15px;  
 -moz-border-radius: 15px;  
 -ms-border-radius: 15px;  
 -o-border-radius: 15px;  
}  
div.button:hover {  
 background-color: #acbcfc;  
 color: #0700D7;  
 transition: color 0.3s;  
}  
div.select {  
 background-color: #D9D9D9;  
 font-family: "Lexend Deca";  
 border: solid 1px #000000;  
 border-radius: 7px;  
 -webkit-border-radius: 7px;  
 -moz-border-radius: 7px;  
 -ms-border-radius: 7px;  
 -o-border-radius: 7px;  
 display: flex;  
 flex-direction: row;  
 justify-content: space-between;  
 align-items: center;  
}  
  
div.select.long {  
 width: 70%;  
 height: 70px;  
}  
div.select.short {  
 width: 40%;  
 height: 30px;  
}  
input.select {  
 height: 100%;  
 width: 100%;  
 font-size: 50px;  
 background-color: #D9D9D9;  
 font-family: "Lexend Deca";  
 border: solid 0px;  
 border-radius: 7px;  
 -webkit-border-radius: 7px;  
 -moz-border-radius: 7px;  
 -ms-border-radius: 7px;  
 -o-border-radius: 7px;  
}  
input.select2 {  
 font-size: 50px;  
 background-color: #D9D9D9;  
 font-family: "Lexend Deca";  
 border: solid 0px;  
 border-radius: 7px;  
 -webkit-border-radius: 7px;  
 -moz-border-radius: 7px;  
 -ms-border-radius: 7px;  
 -o-border-radius: 7px;  
 outline: black 1px solid;  
}  
input.select:focus {  
 outline-width: 0;  
}  
div.select.long:focus {  
 outline-width: 1;  
}  
button.checkmark {  
 height: 60px;  
 width: 60px;  
 padding: 0px;  
 background-color: #D9D9D9;  
}  
button.checkmark:hover {  
 outline: auto;  
 background-attachment: none;  
}  
div.scrollingBox {  
 padding: 5px;  
 flex: 1;  
 width: 95%;  
 height: 100%;  
 overflow-y: scroll;  
 border-radius: 10px;  
 border: 2px solid black;  
 -webkit-border-radius: 10px;  
 -moz-border-radius: 10px;  
 -ms-border-radius: 10px;  
 -o-border-radius: 10px;  
}  
::-webkit-scrollbar {  
 width: 0px;  
}  
div.realDiv {  
 height: 10px;  
 width: 10px;  
}  
select {  
 background-color: #D9D9D9;  
 font-family: "Lexend Deca";  
 height: fit-content;  
 font-size: 50px;  
 padding-right: 5px;  
 border-radius: 5px;  
 -webkit-border-radius: 5px;  
 -moz-border-radius: 5px;  
 -ms-border-radius: 5px;  
 -o-border-radius: 5px;  
 border: 2px solid black;  
}  
input.int {  
 text-align: end;  
 padding-right: 10px !important;  
 height: 70px;  
 width: 50%;  
 font-size: 50px;  
 background-color: #D9D9D9;  
 font-family: "Lexend Deca";  
 border: solid 0px;  
 border-radius: 7px;  
 -webkit-border-radius: 7px;  
 -moz-border-radius: 7px;  
 -ms-border-radius: 7px;  
 -o-border-radius: 7px;  
 border: 1px solid black;  
}  
input.int::-webkit-outer-spin-button,  
input.int::-webkit-inner-spin-button {  
 -webkit-appearance: none;  
 margin: 0;  
}  
  
*/\* Firefox \*/*input.int[type=number] {  
 -moz-appearance: textfield;  
}  
div.title {  
 text-align: center;  
 margin: 10px;  
 font-size: 50px;  
 color: #D9D9D9;  
 color: #0700D7  
}  
div.menu {  
 position: fixed;  
 padding-top: 60px;  
 top: 90px;  
 left: 0px;  
 width: 100%;  
 height: 100%;  
 background-color: #D9D9D9;  
}  
div.menuOption {  
 margin-top: 5px;  
 box-sizing: border-box;  
 padding-right: 20px !important;  
 text-align: end;  
 align-content: center;  
 width: 100%;  
 background-color:#94AAFF;  
 height: 100px;  
}  
div.menuOption:hover {  
 color:#566294;  
}  
.dropdown {  
 position: relative;  
 display: inline-block;  
 width: 100%;  
}  
.dropdown-content {  
overflow: scroll;  
max-height: 700px;  
width: 100%  
  
 position: inherit;  
 flex-direction: column;  
 display: none;  
 position: absolute;  
 background-color: #f1f1f1;  
 min-width: 160px;  
 box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);  
 z-index: 1;  
}  
.show {display:block;}

Index.js

let loggedIn = false;  
let timeOffset = new Date().getTimezoneOffset()/(-60)  
let craftingMeal = []  
let menuState = false;  
let userIncomes = [];  
let userGoals = {protein:-1, fat:-1, carbs:-1};  
  
modalWindows = ["account", "mainWindow", "menu", "plate", "favorites", "results", "goals"];  
let lastIngid = 0;  
let fav\_id = 0  
  
  
async function sendRegisterRequest(login, password) {  
 const response = await fetch("/accounts", {  
 method: "POST",  
 headers: {  
 'Content-Type': 'application/json'  
 },  
 body: JSON.stringify({  
 login: login,  
 pass: password  
 })  
 });  
 if (response.status === 200)  
 return;  
 throw response.status;  
  
}  
function register() {  
 let login = document.getElementById("usernameRegisterLine").value;  
 let password = document.getElementById("passwordRegisterLine").value;  
 sendRegisterRequest(login, password).then(() => registerSuccessful(login), (status) => registerFailed(login, status));  
}  
async function login() {  
 let username = document.getElementById("usernameLoginLine").value;  
 let password = document.getElementById("passwordLoginLine").value;  
 const response = await fetch("/login", {  
 method: "POST",  
 headers: {  
 'Content-Type': 'application/json'  
 },  
 body: JSON.stringify({  
 login: username,  
 pass: password})  
 })  
 if (response.status === 200) {  
 document.getElementById("messagesWindow").innerHTML = response.status;  
 }  
 else {  
 document.getElementById("messagesWindow").innerHTML = response.status;  
 }  
}  
async function logoff() {  
 const response = await fetch("/login", {  
 method: "DELETE",  
 })  
 if (response.status === 200) {  
 window.location.href = "/static/login.html";  
 }  
 window.location.href = "/static/login.html";  
}  
  
  
function registerSuccessful(login) {  
 document.getElementById("messagesWindow").innerHTML = login;  
}  
function registerFailed(login, status) {  
 document.getElementById("messagesWindow").innerHTML = status;  
}  
  
  
async function postMeal(meal) {  
 craftingMeal = [];  
 const mealString = JSON.stringify(meal)  
 const response = await fetch("/incomes", {  
 method: "POST",  
 headers: {  
 'Content-Type': 'application/json'  
 },  
 body: JSON.stringify({  
 json: mealString  
 })  
 })  
   
 userIncomes.push({id:parseInt(await response.text()), login:document.getElementById("usernameSign").innerHTML, json: mealString})  
}  
async function saveMeal() {  
 let meal = {};  
  
 meal.title = document.getElementById("titi").value  
 if (meal.title == "") {  
 meal.title = "No Title";  
 };  
 meal.protein = document.getElementById("proi").value;  
  
 if (meal.protein == "") {  
 meal.protein = "0";  
 };  
 meal.fat = document.getElementById("fati").value;  
  
 if (meal.fat == "") {  
 meal.fat = "0";  
 };  
 meal.carbs = document.getElementById("cari").value;  
  
 if (meal.carbs == "") {  
 meal.carbs = "0";  
 };  
   
 let quantity = document.getElementById("amni").value  
  
 if(document.getElementById("selm").value == "gram") {  
 if (quantity == "") {quantity = 100}  
 quantity /= 100;  
 let amnt = (parseFloat(meal.protein) + parseFloat(meal.fat) + parseFloat(meal.carbs));  
 if (amnt > 100) {  
 alert("Invalid value. Too much macronutrients");  
 return;  
 }  
 } else {  
 if (quantity == "") {quantity = 1}  
 }  
 document.getElementById("amni").value = "";  
 document.getElementById("titi").value = "";document.getElementById("proi").value = "";document.getElementById("fati").value= "";document.getElementById("cari").value = "";  
  
 meal.date = new Date().toISOString().split(".")[0];  
 meal.type = document.getElementById("selm").value;  
 meal.protein = meal.protein\*quantity;  
 meal.fat = meal.fat\*quantity;;  
 meal.carbs = meal.carbs\*quantity;  
 if (document.getElementById("favI").dataset.selected == "1") {  
 if (meal.title == "") {  
 document.getElementById("messagesWindow").innerHTML = "Missing title";  
 return;  
  
 } else {  
 postPreference(meal);  
 }  
 }  
  
 postMeal(meal)  
 location.reload();  
}  
  
async function saveFullPlate() {  
 let meal = getCraftedMeal();  
 craftingMeal = [];  
 recountMeal();  
 meal.title = document.getElementById("titlePlate").value;  
 meal.date = new Date().toISOString().split(".")[0];  
 document.getElementById("titlePlate").value = "";  
 if (meal.title == "")  
 meal.title = "No Title";  
 postMeal(meal);  
 if (document.getElementById("favD").dataset.selected == "1") {  
 postPreference(meal);  
 location.reload();  
 }  
}  
async function savePlate() {  
  
 let meal = getCraftedMeal();  
 meal.protein = meal.protein/getPart();  
 meal.fat = meal.fat/getPart();  
 meal.carbs = meal.carbs/getPart();  
 meal.title = document.getElementById("plateTitle2").value;  
 meal.date = new Date().toISOString().split(".")[0];  
 if (meal.title == "")  
 meal.title = "No Title";  
 postMeal(meal);  
 if (document.getElementById("favP").dataset.selected == "1") {  
 postPreference(meal);  
 }  
 location.reload();  
}  
  
function getCraftedMeal() {  
 return countMeals(craftingMeal);  
}  
  
function countMeals(meals) {  
 let finalMeal = {};  
 meals.forEach(meal => {  
 let mealKeys = Object.keys(meal);  
 mealKeys.forEach(key => {  
 if (finalMeal[key]) {  
 finalMeal[key] += parseFloat(meal[key]);  
 } else {  
 finalMeal[key] = parseFloat(meal[key]);  
 }  
 });  
 });  
 if (finalMeal.protein == undefined)  
 finalMeal.protein = 0;  
 if (finalMeal.fat == undefined)  
 finalMeal.fat = 0;  
 if (finalMeal.carbs == undefined)  
 finalMeal.carbs = 0;  
 return finalMeal;  
}  
  
async function postPreference(preference) {  
 const preferenceString = JSON.stringify(preference);  
 const response = await fetch("/preferences", {  
 method: "POST",  
 headers: {  
 'Content-Type': 'application/json'  
 },  
 body: JSON.stringify({  
 json: preferenceString  
 })  
 })  
 document.getElementById("messagesWindow").innerHTML = response.status;  
}  
  
async function getPrefs() {  
 const response = await fetch("/preferences", {  
 method: "GET",  
 headers: {  
 'Content-Type': 'application/json'  
 },  
 });  
 if (response.status === 200) {  
 preferences = await response.json();  
 return preferences  
 }  
 else throw response.status;  
}  
let preferences;  
getPrefs().then(() => {rebuildFavs()});  
  
function rebuildFavs() {  
 fav = document.getElementById("favBox")  
 fav.innerHTML = "";  
 preferences.forEach(pref => {  
 preference = JSON.parse(pref.json);  
 fav\_id++;  
 elem = document.createElement("div");  
 elem.style = "border: solid 2px black; border-radius: 5px; margin-bottom: 10px;"  
 elem.id = "FAVBOX\_" + pref.id;  
 elem.className = "plateElem";  
 elem.innerHTML = `  
 <div style='display:flex; flex-direction: row; justify-content: space-between; padding-top:10px; padding-right:10px'>  
 <div style='display:flex; flex-direction: column; padding: 20px 30px 20px 30pxж'>  
 <div style='display:flex; flex-direction: row; justify-content: space-between; padding-right: 10px;'>  
 <span> `+preference.title+`:</span>  
 </div>  
 <span>`+preference.protein+`/`+preference.fat+`/`+preference.carbs+` - ` + (preference.protein\*4+preference.fat\*9+preference.carbs\*4)+`kcal</span>  
 </div>  
 <img class = "buttonImg", style = "width: 60px; height:60px" src = "assets/images/delete.png" onclick = "deleteFavorites('` + pref.id + `')">  
 </div>`;  
 fav.appendChild(elem);  
 })  
}  
  
async function getGoals() {  
 const response = await fetch("/goals", {  
 method: "GET",  
 headers: {  
 'Content-Type': 'application/json'  
 },  
 });  
 if (response.status === 200) {  
 goals = await response.json();  
 return goals  
 } else if (response.status === 204) return null  
 else throw response.status;  
}  
  
  
  
  
  
  
  
function showPrefs(elemento, sorter, target) {  
   
 element = document.getElementById(elemento);  
 element.innerHTML = "";  
 let prefType = "";  
 if (document.getElementById(sorter) != null)  
 prefType = document.getElementById(sorter).value;  
 preferences.forEach(preference => {  
 let pref = JSON.parse(preference.json)  
 if (pref.title.includes(prefType)) {  
 elem = document.createElement("div")  
 elem.className = "dropdownElement";  
 elem.style ="border: solid 1px black; width:100%"  
 let kcal = (pref.protein\*4+pref.fat\*9+pref.carbs\*4).toFixed(2);  
 elem.innerHTML = pref.title + ": " + pref.protein + "/" + pref.fat + "/" + pref.carbs + "<br>" + kcal + "kcal";  
 if (target != null)  
 elem.setAttribute( "onClick", "craftingMeal = []; selectPlatePreference('"+ JSON.stringify(pref) +"'); manageClick('null', '"+ elemento + "', 'notnull')" );  
 else elem.setAttribute( "onClick", "selectDishPreference('"+ JSON.stringify(pref) +"');manageClick('null', '"+ elemento + "', 'null')" );  
 element.appendChild(elem);  
 }  
 })  
 database.forEach(preference => {  
 if (preference.protein == undefined) preference.protein = "0";  
 if (preference.fat == undefined) preference.fat = "0";  
 if (preference.carbs == undefined) preference.carbs = "0";  
 let pref = preference;  
 if (pref.title.includes(prefType)) {  
 elem = document.createElement("div")  
 elem.className = "dropdownElement";  
 elem.style ="border: solid 1px black;"  
 let kcal = (pref.protein\*4+pref.fat\*9+pref.carbs\*4).toFixed(2);  
 pref.protein = "" + pref.protein;  
 pref.fat = "" + pref.fat;  
 pref.carbs = "" + pref.carbs;  
 elem.innerHTML = pref.title + ": " + pref.protein + "/" + pref.fat + "/" + pref.carbs + "<br>" + kcal + "kcal";  
 if (target != null)  
 elem.setAttribute( "onClick", "craftingMeal = []; selectPlatePreference('"+ JSON.stringify(pref) +"'); manageClick('null', '"+ elemento + "', 'notnull')" );  
 else elem.setAttribute( "onClick", "selectDishPreference('"+ JSON.stringify(pref) +"');manageClick('null', '"+ elemento + "', 'null')" );  
 element.appendChild(elem);  
 }  
 })  
 element.style = "display: flex;";  
}  
function selectDishPreference(string) {  
   
 dish = JSON.parse(string);  
 if (dish.type == "gram") {  
 document.getElementById("selm").value = "gram";  
 } else {  
 document.getElementById("selm").value = "serv"  
 }  
 document.getElementById("titi").value = dish.title;  
 document.getElementById("proi").value = dish.protein;  
 document.getElementById("fati").value = dish.fat;  
 document.getElementById("cari").value = dish.carbs;  
}  
  
function selectPlatePreference(string) {  
   
 document.getElementById("plateBox").innerHTML = "";  
  
 dish = JSON.parse(string);  
 id = lastIngid;  
 protein = dish.protein;  
 fat = dish.fat;  
 carbs = dish.carbs;  
 title = dish.title;  
 addIngridient(protein, fat, carbs, title, id);  
 addToPlate(protein, fat, carbs, title, id);  
 lastIngid++;  
 document.getElementById("titlePlate").value = title;  
 document.getElementById("plateTitle2").value = title;  
}  
  
  
  
async function getIncomes() {  
 const response = await fetch("/incomes", {  
 method: "GET",  
 })  
 if (response.status === 200) {  
 return response.json();  
 }  
 else if (response.status === 204)  
 return [];  
 else {  
 document.getElementById("messagesWindow").innerHTML = response.status;  
 }  
}  
  
function countGoals() {  
 massS = document.getElementById("mass").value;  
 if (parseFloat(massS) <= 10 || parseFloat(massS) >= 500) {  
 alert("invalid value of mass")  
 return;  
 }  
 ageS = document.getElementById("age").value;  
 if (parseFloat(ageS) <= 13 || parseFloat(ageS) >= 99) {  
 alert("invalid value age")  
 return;  
 }  
 heiS = document.getElementById("height").value;  
 if (parseFloat(heiS) <= 80 || parseFloat(heiS) >= 250) {  
 alert("invalid value of height")  
 return;  
 }  
 if (massS == "") massS = 70;  
 if (ageS == "") ageS = 18;  
 if (heiS == "") heiS = 175;  
 mass = parseFloat(massS);  
 age = parseFloat(ageS);  
 height = parseFloat(heiS);  
 sex = document.getElementById("sex").value;  
 activity = parseFloat(document.getElementById("activity").value);  
 if (sex == "male") {  
 return (10\*mass + 6.25\*height - 5\*age + 5)\*activity;  
 } else {  
 return (10\*mass + 6.25\*height - 5\*age - 161)\*activity;  
 }  
}  
  
function goalsM() {  
 getIncomes().then((incomesunparsed) => {  
   
 userIncomes = incomesunparsed;  
  
 dateStart = new Date();  
 dateEnd = new Date();  
 dateStart.setHours(0,0,0,0);  
 dateEnd.setHours(23,59,59,59);  
 let todayIncomes = [];  
 todayIncomes = selectIncomesByDate(dateStart.toISOString(), dateEnd.toISOString());  
  
 let countProt = 0;  
 let countFat = 0;  
 let countCarbs = 0;  
 todayIncomes.forEach(income => {  
 if (income.protein == "") income.protein = "0";  
 if (income.fat == "") income.fat = "0";  
 if (income.carbs == "") income.carbs = "0";  
 countProt += parseFloat(income.protein);  
 countFat += parseFloat(income.fat);  
 countCarbs += parseFloat(income.carbs);  
  
 })  
  
 dateStart = new Date();  
 dateStart.setDate(dateStart.getDate() - 8);  
 dateEnd = new Date();  
 dateEnd.setDate(dateEnd.getDate() - 1);  
 let weeklyIncomes = [];  
 weeklyIncomes = selectIncomesByDate(dateStart.toISOString(), dateEnd.toISOString());  
  
 let countProtw = 0;  
 let countFatw = 0;  
 let countCarbsw = 0;  
 weeklyIncomes.forEach(income => {  
 if (income.protein == "") income.protein = "0";  
 if (income.fat == "") income.fat = "0";  
 if (income.carbs == "") income.carbs = "0";  
 countProtw += parseFloat(income.protein);  
 countFatw += parseFloat(income.fat);  
 countCarbsw += parseFloat(income.carbs);  
 })  
  
 dateStart = new Date();  
 dateStart.setDate(dateStart.getDate() - 31);  
 dateEnd = new Date();  
 dateEnd.setDate(dateEnd.getDate() - 1);  
 let monthlyIncomes = [];  
 monthlyIncomes = selectIncomesByDate(dateStart.toISOString(), dateEnd.toISOString());  
  
 let countProtm = 0;  
 let countFatm = 0;  
 let countCarbsm = 0;  
 weeklyIncomes.forEach(income => {  
 countProtm += parseFloat(income.protein);  
 countFatm += parseFloat(income.fat);  
 countCarbsm += parseFloat(income.carbs);  
 })  
  
  
  
  
 document.getElementById("todayProt").innerHTML = countProt + "g";  
 document.getElementById("todayFat").innerHTML = countFat + "g";  
 document.getElementById("todayCarbs").innerHTML = countCarbs + "g";  
  
 document.getElementById("weekdata").innerHTML = (countProtw/7).toFixed(1) + "/"+ (countFatw/7).toFixed(1) + "/" + (countCarbsw/7).toFixed(1);  
 document.getElementById("monthdata").innerHTML = (countProtm/30).toFixed(1) + "/"+ (countFatm/30).toFixed(1) + "/" + (countCarbsm/30).toFixed(1);  
  
 getGoals().then((goals) => rebuildGoals(goals));  
  
 function rebuildGoals(goals) {  
   
 if (goals == null) {  
 document.getElementById("todayProtPercent").innerHTML = "No goal"  
 document.getElementById("todayFatPercent").innerHTML = "No goal"  
 document.getElementById("todayCarbsPercent").innerHTML = "No goal"  
 document.getElementById("goalsDiv").innerHTML = "No goal"  
 } else {  
 goal = JSON.parse(goals.json)  
 userGoals = goal;  
 document.getElementById("todayProtPercent").innerHTML = ((countProt/goal.protein)\*100).toFixed(1) + "%"  
 document.getElementById("todayFatPercent").innerHTML = ((countFat/goal.fat)\*100).toFixed(1) + "%"  
 document.getElementById("todayCarbsPercent").innerHTML = ((countCarbs/goal.carbs)\*100).toFixed(1) + "%"  
 document.getElementById("goalsDiv").innerHTML = "p " + goal.protein.toFixed(1) + "|" + "f " + goal.fat.toFixed(1) + "|" + "c " + goal.carbs.toFixed(1) + "<br>" + (goal.protein\*4 + goal.fat\*9 + goal.carbs\*4).toFixed(1) + " kkcal";  
  
 document.getElementById("weekdata2").innerHTML = ((countProtw/(goal.protein\*7))\*100).toFixed(1) + "/"+ ((countFatw/(goal.fat\*7))\*100).toFixed(1) + "/" + ((countCarbsw/(goal.carbs\*7))\*100).toFixed(1) + "%";  
 document.getElementById("monthdata2").innerHTML = ((countProtm/(goal.protein\*30))\*100).toFixed(1) + "/"+ ((countFatm/(goal.fat\*30))\*100).toFixed(1) + "/" + ((countCarbsm/(goal.carbs\*30))\*100).toFixed(1) + "%";  
  
  
 }  
 }  
  
  
  
 });  
}  
  
  
async function addToDate(protein, fat, carbs, title, id) {  
 if ((title == "") || (title == undefined))  
 title = "Without name"  
 elem = document.createElement("div");  
 elem.style = "border: solid 2px black; border-radius: 5px; margin-bottom: 10px;"  
 elem.id = "DATE\_" + id;  
 elem.className = "plateElem";  
 elem.innerHTML = `  
 <div style='display:flex; flex-direction: row; justify-content: space-between; padding-top:10px; padding-right:10px'>  
 <div style='display:flex; flex-direction: column; padding: 20px 30px 20px 30pxж'>  
 <div style='display:flex; flex-direction: row; justify-content: space-between; padding-right: 10px;'>  
 <span> `+title+`:</span>  
  
 </div>  
 <span>`+protein+`/`+fat+`/`+carbs+` - ` + (protein\*4+fat\*9+carbs\*4)+`kcal</span>  
 </div>  
 <img class = "buttonImg", style = "width: 60px; height:60px" src = "assets/images/delete.png" onclick = "deleteFromDate('` + id + `')">  
 </div>`;  
 document.getElementById("dateBox").appendChild(elem);  
}  
async function addToDatev2(protein, fat, carbs, title, id) {  
 if ((title == "") || (title == undefined))  
 title = "Without name"  
 elem = document.createElement("div");  
 elem.style = "border: solid 2px black; border-radius: 5px; margin-bottom: 10px;"  
 elem.id = "DATE\_" + id;  
 elem.className = "plateElem";  
 elem.innerHTML = `  
 <div style='display:flex; flex-direction: row; justify-content: space-between; padding-top:10px; padding-right:10px'>  
 <div style='display:flex; flex-direction: column; padding: 20px 30px 20px 30pxж'>  
 <div style='display:flex; flex-direction: row; justify-content: space-between; padding-right: 10px;'>  
 <span> `+title+`:</span>  
  
 </div>  
 <span>`+protein+`/`+fat+`/`+carbs+` - ` + (protein\*4+fat\*9+carbs\*4)+`kcal</span>  
 </div>  
 </div>`;  
 document.getElementById("dateBox").appendChild(elem);  
}  
  
  
  
function dateSelected() {  
   
 document.getElementById("dateBox").innerHTML = "";  
 let date = document.getElementById("resultDate").value;  
 todayDate = new Date();  
 dateStart = new Date(date);  
 dateEnd = new Date(date);  
 dateStart.setHours(0,0,0,0);  
 dateEnd.setHours(23,59,59,59);  
 let dateIncomes = [];  
 dateIncomes = selectIncomesByDate(dateStart.toISOString(), dateEnd.toISOString());  
 let totalMeal = {protein:0, fat:0, carbs:0}  
 if(dateStart.setHours(0,0,0,0) == todayDate.setHours(0,0,0,0)) {  
 dateIncomes.forEach(realIncome => {  
 totalMeal.protein += parseFloat(realIncome.protein),  
 totalMeal.fat += parseFloat(realIncome.fat),  
 totalMeal.carbs += parseFloat(realIncome.carbs),  
 addToDate(realIncome.protein, realIncome.fat, realIncome.carbs, realIncome.title, realIncome.id)  
 })  
 } else {  
 dateIncomes.forEach(realIncome => {  
 totalMeal.protein += parseFloat(realIncome.protein),  
 totalMeal.fat += parseFloat(realIncome.fat),  
 totalMeal.carbs += parseFloat(realIncome.carbs),  
 addToDatev2(realIncome.protein, realIncome.fat, realIncome.carbs, realIncome.title, realIncome.id)  
 })  
 }  
 document.getElementById("dateres").innerHTML = totalMeal.protein.toFixed(1) + "/" + totalMeal.fat.toFixed(1) + "/" + totalMeal.carbs.toFixed(1) + " " + (totalMeal.protein\*4 + totalMeal.fat\*9 + totalMeal.carbs\*4).toFixed(1) + "kcal";  
}  
  
async function recountGoals() {  
 let kcal = countGoals();  
 let protein;  
 let fat;  
 let carbs;  
 let mass = document.getElementById("mass").value;  
 if (mass == "") mass = '70';  
  
 let desire = document.getElementById("desire").value;  
 if (desire == "gain") {  
 kcal += parseFloat(mass)\*2;  
 fat = (kcal\*0.25)/9;  
 protein = parseFloat(mass)\*1.8;  
 carbs = (kcal - fat\*9 - protein\*4)/4  
 } else if (desire == "keep") {  
 fat = (kcal\*0.3)/9;  
 protein = parseFloat(mass)\*1;  
 carbs = (kcal - fat\*9 - protein\*4)/4  
 } else {  
 kcal -= parseFloat(mass)\*2;  
 fat = (kcal\*0.2)/9;  
 protein = parseFloat(mass)\*1.2;  
 carbs = (kcal - fat\*9 - protein\*4)/4  
 }  
  
 const goals = {protein:protein, fat:fat, carbs:carbs}  
 const goalString = JSON.stringify(goals);  
 const response = await fetch("/goals", {  
 method: "POST",  
 headers: {  
 'Content-Type': 'application/json'  
 },  
 body: JSON.stringify({  
 json: goalString  
 })  
 });  
 goalsM();  
}  
  
  
goalsM();  
function selectIncomesByDate(dateStart, dateEnd) {  
 let res = [];  
 userIncomes.forEach(income => {  
 let realIncome = JSON.parse(income.json)  
 if ((realIncome.date >= dateStart) && (realIncome.date <= dateEnd)) {  
 realIncome.id = income.id;  
 res.push(realIncome);  
  
 }  
 })  
 return res;  
}  
  
*/////////SOMEWHAT GOOD UPPER THAN THIS  
/////////SHIT UNDER THIS*async function createMeal() {  
 let meal = getCraftedMeal()  
 isNow = document.getElementById("nowMealButton").checked;  
 let date;  
 if (isNow) {  
 date = new Date().toISOString().split(".")[0];  
 } else {  
 date = new Date(document.getElementById("dateMealLine").value);  
 }  
 await saveMeal(meal, date)  
}  
  
  
async function getResultByDate(date) {  
   
 affixValue = parseInt(document.getElementById("hourBorder").value);  
 *//2024-03-04* let dateStart = new Date(date);  
 let dateEnd = new Date(date);  
 dateStart.setHours(0+affixValue,0,0,0);  
 dateEnd.setHours(23+affixValue,59,59,59);  
 meals = await getMeals(dateStart.toISOString().split(".")[0], dateEnd.toISOString().split(".")[0])  
 result = countMeals(meals); *// <- Result object* document.getElementById("messagesWindow").innerHTML = result.totalProtein + "/" + result.totalFat + "/" + result.totalCarbs + "/" + result.calories + "/" + result.fatPercent;  
}  
  
async function todayResult() {  
   
 today = new Date();  
 affixValue = parseInt(document.getElementById("hourBorder").value);  
 if (today.getHours() < affixValue)  
 today.setDate(today.getDate()-1);  
   
 let month = String(today.getMonth() + 1);  
 if (month.length == 1) {  
 month = "0" + month;  
 }  
 let date = String(today.getDate());  
 if (date.length == 1) {  
 date = "0" + date;  
 }  
 dateToday = today.getFullYear() + "-" + month + "-" + date;  
   
  
  
 getResultByDate(dateToday)  
}  
async function checkAuth() {  
 const response = await fetch("/auth", {  
 method: "GET",  
 })  
 if (response.redirected) {  
 window.location.href = response.url;  
 } else if (response.status === 200) {  
 loggedIn = true;  
 document.getElementById("usernameSign").innerHTML = (await response.text());  
 } else {  
 window.location.href = "/static/login.html";  
 }  
}  
async function getDayDivisor() {  
 const response = await fetch("/daydivisors", {  
 method: "GET",  
 })  
 if (response.status === 200) {  
 return (await response.json()).daydivisor;  
 }  
 else {  
 document.getElementById("messagesWindow").innerHTML = response.status;  
 }  
}  
async function setDayDivisor(value) {  
 const response = await fetch("/daydivisors", {  
 method: "PUT",  
 body: JSON.stringify({  
 daydivisor: value  
 }),  
 })  
 if (response.status === 200) {  
 return  
 }  
 else {  
 document.getElementById("messagesWindow").innerHTML = response.status;  
 }  
}  
function sleep(ms) {  
 return new Promise(resolve => setTimeout(resolve, ms));  
}  
async function getDesires() {  
 const response = await fetch("/desires", {  
 method: "GET",  
 })  
 if (response.status === 200) {  
 result = await response.json();  
 reqpProtein = parseFloat(result.reqprotein);  
 reqFat = parseFloat(result.reqfat);  
 reqCarbs = parseFloat(result.reqcarbs);  
 reqCalories = parseFloat(result.reqcalories);  
 return {reqProtein, reqFat, reqCarbs, reqCalories};  
 }  
 else {  
 document.getElementById("messagesWindow").innerHTML = response.status;  
 }  
}  
async function setDesires(protein, fat, carbs, calories) {  
 const response = await fetch("/desires", {  
 method: "PUT",  
 body: JSON.stringify({  
 reqprotein: protein,  
 reqfat: fat,  
 reqcarbs: carbs,  
 reqcalories: calories  
 }),  
 })  
 if (response.status === 200) {  
 return  
 }  
 else {  
 document.getElementById("messagesWindow").innerHTML = response.status;  
 }  
}  
async function addIngridient(protein, fat, carbs, title, id) {  
 craftingMeal.push({protein, fat, carbs, title, id})  
 recountMeal();  
}  
async function deleteIngridient(id) {  
 craftingMeal = craftingMeal.filter(obj => obj.id !== id);  
 recountMeal();  
}  
async function deleteIncome(id) {  
 const response = await fetch("/incomes/" + id, {  
 method: "DELETE",  
 })  
 if (response.status === 200) {  
 return  
 }  
 else {  
 document.getElementById("messagesWindow").innerHTML = response.status;  
 }  
}  
async function newIngridient() {  
   
 let protein = document.getElementById("proi").value;  
 if (protein == "") protein = "0";  
  
 let fat = document.getElementById("fati").value  
 if (fat == "") fat = "0";  
  
 let carbs = document.getElementById("cari").value  
 if (carbs == "") carbs = "0";  
  
 let quantity = document.getElementById("amni").value  
  
 let title = document.getElementById("titi").value;  
  
 if(document.getElementById("selm").value == "gram") {  
 type = 1;  
 if (quantity == "") {quantity = 100}  
 if ((parseFloat(protein) + parseFloat(fat) + parseFloat(carbs)) > 100) {  
 alert("Invalid value. Too much macronutrients");  
 return;  
 }  
 quantity /= 100;  
 } else {  
 if (quantity == "") {quantity = 1}  
 type = 0  
 }  
 protein \*= quantity;  
 fat \*= quantity;  
 carbs \*= quantity;  
 id = lastIngid;  
 addIngridient(protein, fat, carbs, title, id);  
 addToPlate(protein, fat, carbs, title, id);  
 lastIngid++;  
 if (document.getElementById("favI").dataset.selected == "1") {  
 addFav(protein, fat, carbs, title)  
 }  
 document.getElementById("cari").value = "";  
 document.getElementById("proi").value = "";  
 document.getElementById("fati").value = "";  
document.getElementById("amni").value = "";  
 document.getElementById("titi").value = "";  
}  
async function addToPlate(protein, fat, carbs, title, id) {  
 if ((title == "") || (title == undefined))  
 title = "Without name"  
 elem = document.createElement("div");  
 elem.style = "border: solid 2px black; border-radius: 5px; margin-bottom: 10px;"  
 elem.id = id;  
 elem.className = "plateElem";  
 elem.innerHTML = `  
 <div style='display:flex; flex-direction: row; justify-content: space-between; padding-top:10px; padding-right:10px'>  
 <div style='display:flex; flex-direction: column; padding: 20px 30px 20px 30pxж'>  
 <div style='display:flex; flex-direction: row; justify-content: space-between; padding-right: 10px;'>  
 <span> `+title+`:</span>   
   
 </div>  
 <span>`+protein+`/`+fat+`/`+carbs+` - ` + (protein\*4+fat\*9+carbs\*4)+`kcal</span>  
 </div>  
 <img class = "buttonImg", style = "width: 60px; height:60px" src = "assets/images/delete.png" onclick = "deleteFromPlate(` + id + `)">  
 </div>`;  
 document.getElementById("plateBox").appendChild(elem);  
   
   
   
}  
  
function recountMeal() {  
 result = getCraftedMeal();  
 prot = result.protein.toFixed(2);  
 fat = result.fat.toFixed(2);  
 carbs = result.carbs.toFixed(2);  
 document.getElementById("mealstat").innerHTML = prot + "/" + fat + "/" + carbs;  
 document.getElementById("mealkkal").innerHTML = (prot\*4+fat\*9+carbs\*4).toFixed(1) + "kcal";  
 if (userGoals.protein != "-1")  
 document.getElementById("mealprecent").innerHTML = ((prot/userGoals.protein)\*100).toFixed(1) + "/" + ((fat/userGoals.fat)\*100).toFixed(1) + "/" + ((carbs/userGoals.carbs)\*100).toFixed(1) + "%"  
}  
checkAuth();  
function changeMealType() {  
 if(document.getElementById("mealType").value == "gramm") {  
 document.getElementById("mealQuantity").placeholder = "Грамм"  
 } else {  
 document.getElementById("mealQuantity").placeholder = "Штук"  
 }  
  
}  
  
async function selectPreferenceOption(protein, fat, carbs, title) {  
   
 document.getElementById("titleMealLine").value = title;  
 document.getElementById("proteinMealLine").value = protein;  
 document.getElementById("fatMealLine").value = fat;  
 document.getElementById("carbsMealLine").value = carbs;  
}  
async function changeMenuState(force = false) {  
 if (menuState || force) {  
 document.getElementById("menuHider").style.display = "none";  
 menuState = false  
 }  
 else {  
 document.getElementById("menuHider").style.display = "flex";  
 menuState = true  
 }  
}  
*//window.addEventListener("click", function(e){  
 // if(e.target == document.getElementById("preferenceDropdownMenu"))  
 // changeMenuState();  
 //else changeMenuState(true);  
//})*function select1(s) {  
 document.getElementById(s).style.outline = "solid thin #000000";  
}  
function unfocus1(s){  
 document.getElementById(s).style.outline = "0";  
}  
function getPart() {  
 val1 = document.getElementById("parts1").value;  
 if (val1 == "") val1 = "1";  
 val2 = document.getElementById("parts2").value;  
 if (val2 == "") val2 = "1";  
 return parseFloat(val1)/parseFloat(val2);  
}  
  
async function unfocusandclose(s, e, type){  
 if (s == "null") {  
 document.getElementById(e).style.display = "none";  
 return;  
 }  
 document.getElementById(e).style.display = "none";  
 document.getElementById(s).blur()  
 document.getElementById(s).style.outline = "0";  
  
}  
function focusandopen(s, e, type){  
 showPrefs(e, s, type);  
 document.getElementById(s).style.outline = "solid thin #000000";  
 document.getElementById(e).style.display = "flex";  
}  
  
function manageClick(s, e, type) {  
   
 let elem = document.getElementById(e);  
 if (elem.dataset.opened == "1") {  
 unfocusandclose(s, e, type);  
 elem.dataset.opened = "0";  
 } else {  
 focusandopen(s, e, type);  
 elem.dataset.opened = "1";  
 }  
}  
function forceClose() {  
 e1 = document.getElementById("dropdownTitleDish");  
 e2 = document.getElementById("dropdownTitleDish2");  
 e3 = document.getElementById("dropdownTitleDish3");  
 e1.dataset.opened = "0";  
 e2.dataset.opened = "0";  
 e3.dataset.opened = "0";  
 e1.style.display = "none";  
 e2.style.display = "none";  
 e3.style.display = "none";  
}  
function selectButton(obj) {  
 if (document.getElementById(obj).dataset.selected == "1") {  
 document.getElementById(obj).style.background = "#D9D9D9";  
 document.getElementById(obj).dataset.selected = "0";  
 } else {  
 document.getElementById(obj).style.background = "url(assets/images/selected.png)";  
 document.getElementById(obj).dataset.selected = "1"  
 }  
}  
function showMenu() {  
 forceClose();  
 if (document.getElementById("menu").dataset.opened == "0") {  
 document.getElementById("menu").style.display = "block"  
 document.getElementById("menu").dataset.opened = "1"  
 } else {  
 document.getElementById("menu").style.display = "none"  
 document.getElementById("menu").dataset.opened = "0"  
 }  
}  
function changedMeasurement() {  
 if (document.getElementById("selm").value == "serv") {  
 document.getElementById("meastext").innerHTML= " - Servings";  
 document.getElementById("amni").placeholder = "1";  
 } else {  
 document.getElementById("meastext").innerHTML= " - Grams";  
 document.getElementById("amni").placeholder = "100";  
 }  
}  
function closeExcept(openWindow) {  
 modalWindows.forEach(window => {   
 if (window != openWindow)  
 document.getElementById(window).style.display = "none";   
 else document.getElementById(window).style.display = "flex";  
 }  
 );  
}  
function deleteFromDate(id) {  
 document.getElementById("DATE\_" + id).remove();  
 deleteIncome(id);  
}  
  
function deleteFromPlate(id) {  
 document.getElementById(id).remove();  
 deleteIngridient(id);  
}  
async function deleteFavorites(id) {  
 document.getElementById("FAVBOX\_" + id).remove();  
 const response = await fetch("/preferences/" + id, {  
 method: "DELETE",  
 headers: {  
 'Content-Type': 'application/json'  
 },  
 })  
  
 if (response.status === 200) {  
 getPrefs().then(() => {rebuildFavs()});  
 return;  
 }  
  
 throw response.status;  
  
}  
function addFav(protein, fat, carbs, title, id) {  
 elem = document.createElement("div");  
 elem.style = "border: solid 2px black; border-radius: 5px; margin-bottom: 10px;"  
 elem.id = id;  
 elem.className = "plateElem";  
 elem.innerHTML = `  
 <div style='display:flex; flex-direction: row; justify-content: space-between; padding-top:10px; padding-right:10px'>  
 <div style='display:flex; flex-direction: column; padding: 20px 30px 20px 30pxж'>  
 <div style='display:flex; flex-direction: row; justify-content: space-between; padding-right: 10px;'>  
 <span> `+title+`:</span>   
   
 </div>  
 <span>`+protein+`/`+fat+`/`+carbs+` - ` + (protein\*4+fat\*9+carbs\*4)+`kcal</span>  
 </div>  
 <img class = "buttonImg", style = "width: 60px; height:60px" src = "assets/images/delete.png" onclick = "deleteFromFav(FAV\_` + title + `)">  
 </div>`;  
 document.getElementById("favBox").appendChild(elem);  
}  
function deleteFromFav(title) {  
 document.getElementById("FAV\_" + title).remove();  
}let database = [];  
getDb().then(db => {  
 database = db  
  
});  
  
  
async function getDb() {  
 const response = await fetch("/database", {  
 method: "GET",  
 headers: {  
 'Content-Type': 'application/json'  
 }  
 });  
 if (response.status === 200)  
 return response.json();  
 else throw response.status;  
  
}

login.js

async function sendRegisterRequest(login, password) {  
 const response = await fetch("/accounts", {  
 method: "POST",  
 headers: {  
 'Content-Type': 'application/json'  
 },  
 body: JSON.stringify({  
 login: login,  
 pass: password  
 })  
 });  
 if (response.status === 200)  
 return;  
 throw response.status;  
  
}  
function register() {  
 let login = document.getElementById("usernameRegisterLine").value;  
 let password = document.getElementById("passwordRegisterLine").value;  
 sendRegisterRequest(login, password).then(() => registerSuccessful(login), (status) => registerFailed(login, status));  
}  
async function login() {  
 let username = document.getElementById("usernameLoginLine").value;  
 let password = document.getElementById("passwordLoginLine").value;  
 const response = await fetch("/login", {  
 method: "POST",  
 headers: {  
 'Content-Type': 'application/json'  
 },  
 body: JSON.stringify({  
 login: username,  
 pass: password})  
 })  
 if (response.status === 200) {  
 window.location.href = "/static/index.html";  
 } else if (response.status == 401)  
 document.getElementById("msg").innerHTML = "Wrong login and/or a password";  
 else {  
 document.getElementById("messagesWindow").innerHTML = response.status;  
 }  
}  
async function logoff() {  
 const response = await fetch("/login", {  
 method: "DELETE",  
 })  
 if (response.status === 200) {  
 document.getElementById("msg").innerHTML = response.status;  
 }  
 else {  
 document.getElementById("msg").innerHTML = response.status;  
 }  
}  
  
  
async function registerSuccessful(login) {  
 const response = await fetch("/login", {  
 method: "POST",  
 headers: {  
 'Content-Type': 'application/json'  
 },  
 body: JSON.stringify({  
 login: login,  
 pass: document.getElementById("passwordRegisterLine").value})  
 })  
 if (response.status === 200) {  
 window.location.href = "/static/index.html";  
 } else if (response.status == 401)  
 document.getElementById("msg").innerHTML = "Wrong login and/or a password";  
 window.location.href = "/static/index.html";  
}  
function registerFailed(login, status) {  
 if (status == 409)  
 document.getElementById("msg").innerHTML = "User already exists. Try another login";  
}  
async function checkAuth() {  
 const response = await fetch("/auth", {  
 method: "GET",  
 })  
 if (response.redirected)  
 return;  
 else if (response.status === 200) {  
 window.location.href = "/static/index.html";  
 }  
}  
checkAuth();