

**Колледж космического машиностроения и технологий**

**ОТЧЕТ**

**по производственной практике (преддипломной)**

Выполнил студент:

Денисов М.В.

группа П1-18

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

Преподаватель:

Бобкова Н. Ю.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(оценка)

Королев 2022

Оглавление

[1. Характеристика объекта практики 2](#_Toc104742979)

[**1.1.** **Краткая технико-экономическая характеристика предприятия** 2](#_Toc104742980)

[**1.1.1. Структура предприятия** 2](#_Toc104742981)

[**1.1.2. Основные задачи управления качества образования** 4](#_Toc104742982)

[**1.1.3. Характеристика аппаратного обеспечения** 7](#_Toc104742983)

[**1.2.** **Характеристика программного обеспечения** 8](#_Toc104742984)

[**1.3 Характеристика методов проектирования и разработки программного обеспечения** 9](#_Toc104742985)

[2. Теоретическая часть для ВКР 12](#_Toc104742986)

[**2.1.** **Анализ предметной области** 12](#_Toc104742987)

[**2.2.** **Анализ имеющихся программных решений** 12](#_Toc104742988)

[**Примеры сайтов** 12](#_Toc104742989)

[**2.3.** **Разработка бизнес-модели обработки данных** 15](#_Toc104742990)

[**2.4.** **Разработка алгоритм решения поставленной задачи** 17](#_Toc104742991)

[**2.6.** **Разработка структуры главного модуля** 19](#_Toc104742992)

[**2.7.** **Разработка структуры БД** 22](#_Toc104742993)

[3. Организационно-экономическая часть для ВКР 23](#_Toc104742994)

[**3.1.** **Раздел техники безопасности** 23](#_Toc104742995)

[Литература и интернет источники 29](#_Toc104742996)

# Характеристика объекта практики

## **Краткая технико-экономическая характеристика предприятия**

### **1.1.1. Структура предприятия**

Предприятие, на котором проходит преддипломная практика – ГБОУ ВО МО «Технологический университет».

Университет образован 16 июля 1998 года в форме некоммерческой организации с названием: Негосударственное образовательное учреждение «Королевская академия управления, экономики и социологии».

27 апреля 1999 года наименование было изменено на Негосударственное образовательное учреждение «Королевский институт управления, экономики и социологии».

Постановлением Главы города Королёва от 13 июня 2001 года №1009 Королевский институт управления, экономики и социологии был преобразован в муниципальное образовательное учреждение.

26 декабря 2005 года постановлением Правительства Московской области № 964/52 вуз перешёл в собственность Московской области и получил наименование: Государственное образовательное учреждение высшего профессионального образования Московской области «Королевский институт управления, экономики и социологии». Этим же постановлением учредителем вуза было назначено Министерство образования Московской области.

Постановлением Правительства Московской области от 25.09.2007 года № 725/32 Государственное образовательное учреждение высшего профессионального образования Московской области «Королевский институт управления, экономики и социологии» передано в ведомственное подчинение Министерства финансов Московской области.

В конце декабря 2011 года Приказом Федеральной службы по надзору в сфере образования и науки КИУ­ЭС признан прошедшим проце­дуру государственной аккреди­тации с установлением статуса «академия». 12 апреля 2012 года постановлением Правительства Московской области № 484/13 вуз переименован в Государственное бюджетное образовательное учреждение высшего профессионального образования Московской области «Финансово-технологическая академия».

11 июля 2012 года постановлением Правительства Московской области № 906/24 Академия была реорганизована в форме присоединения к ней государственного бюджетного образовательного учреждения среднего профессионального образования Московской области «Королевский государственный техникум технологии и дизайна одежды» и государственного бюджетного образовательного учреждения среднего профессионального образования Московской области «Королевский колледж космического машиностроения и технологии»

26 декабря 2013 года постановлением Правительства Московской области от 26.12.2013 № 1137/57 «О передаче Государственного бюджетного образовательного учреждения высшего профессионального образования Московской области «Финансово-технологическая академия» в ведомственное подчинение Министерству образования Московской области» Академия была передана в ведомственное подчинение Министерству образования Московской области.

20 января 2015 года постановлением Правительства Московской области от 20.01.2015 № 9/2 Академии присвоен статус «университета» и вуз переименован в Государственное бюджетное образовательное учреждение высшего образования Московской области «Технологический университет».

3 апреля 2015 года Технологическим университетом, Администрацией г.о. Королёв и ведущими предприятиями ракетно-космической отрасли РФ, работающими в наукограде Королёве, ряде городов Московской области и г. Москве, подписано соглашение о создании консорциума «Региональный научно-образовательный кластер «Северо-восток» с целью обеспечения потребностей наукоёмких высокотехнологичных предприятий квалифицированными кадрами.

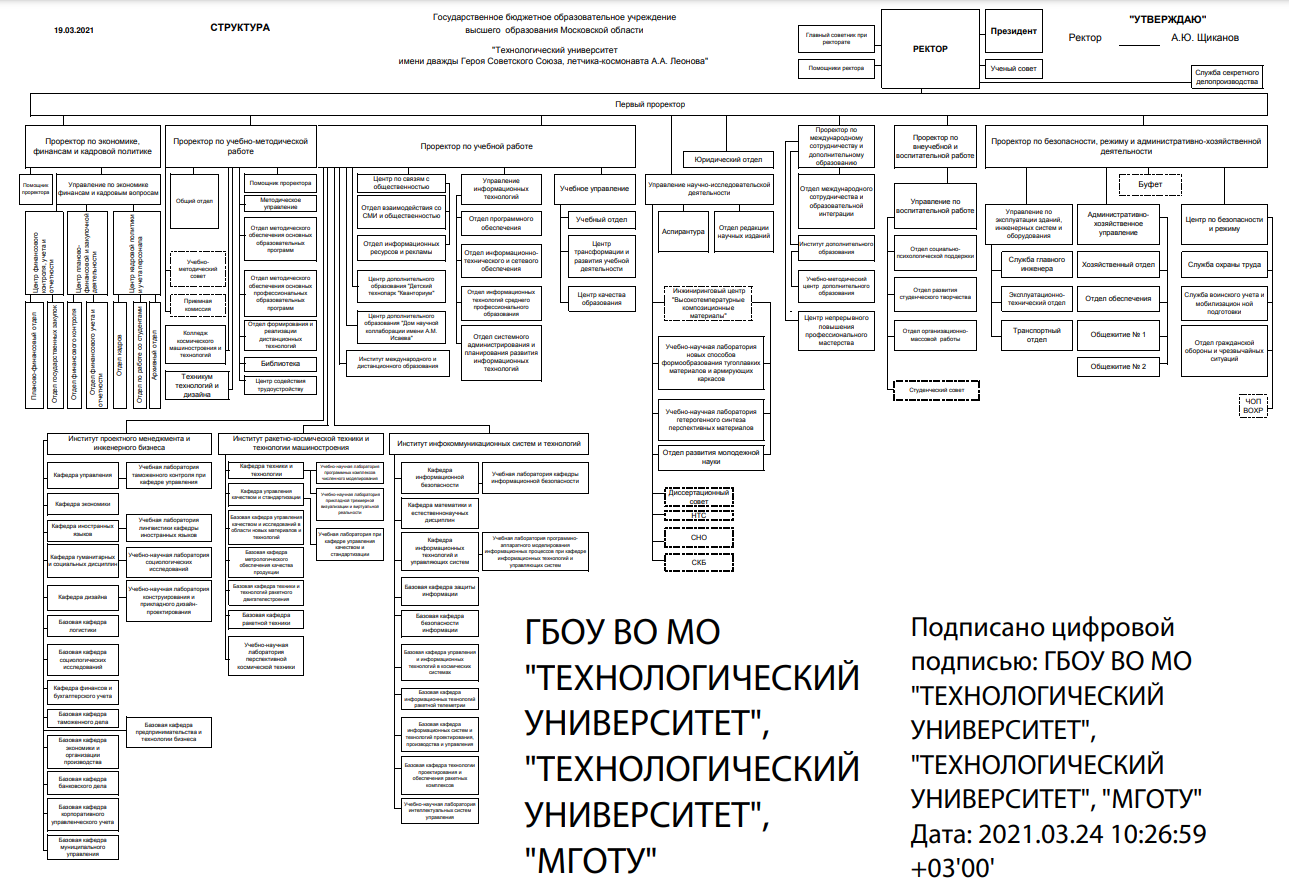


Рисунок 1. Структура МГОТУ

### **1.1.2. Основные задачи управления качества образования**

Основными задачами управления являются:

1. Формирование системы менеджмента качества Университета системы международных стандартов ISO 9001:2015; российских стандартов ИСО

Р 9001-2015, а также соблюдения требований федеральных государственных образовательных стандартов (далее – ФГОС).

1. Содействие реализации стратегии развития Университета и проведение политики в области качества; разработка предложений и реализация решений руководства Университета по повышению качества образования и конкурентоспособности образовательной организации (далее – ОО) в соответствии с требованиями современного общества и участников образовательного процесса.
2. Обеспечение эффективного функционирования системы менеджмента качества, своевременного планирования, координации, методической поддержки и контроля выполнения работ, осуществляемых всеми подразделениями Университета по обеспечению качества образования.
3. Организация работ по совершенствованию системы менеджмента качества (далее - СМК) образования в соответствии с современными тенденциями в области управления качеством.
4. Рациональная организация и эффективное проведение всех необходимых мероприятий по контролю управления качеством, принятия по их результатам корректирующих и предупреждающих действий, направленных на повышение качества образования
5. Систематическое проведение и анализ результатов внутренних проверок СМК с целью оценки ее эффективности и определения путей развития.
6. Организация мероприятий по подготовке и прохождению процедур лицензирования образовательной деятельности и государственной аккредитации.
7. Организация процедур по подготовке форм статистической отчетности № 1-Мониторинг и № СПО-Мониторинг по основным направлениям деятельности Университета.
8. Организация информационно-методического сопровождения системы контроля качества учебного процесса и подготовка предложений по вопросам обеспечения и совершенствования качества образовательных услуг.
9. Организация и участие в проведении внутренних аудитов и самообследования Университета, его структурных подразделений, образовательных процессов и подготовка отчетов руководству.
10. Координация деятельности уполномоченных по качеству от структурных подразделений Университета.
11. Организация обучения уполномоченных структурных подразделений по вопросам ведения делопроизводства по управлению качеством образовательных услуг.
12. Организационное и информационное сопровождение процесса содействия трудоустройству выпускников Университета.
13. Мониторинг и анализ потребностей регионального рынка труда, подготовка предложений по повышению конкурентоспособности выпускников Университета.

### **1.1.3. Характеристика аппаратного обеспечения**

Аппаратное обеспечение учебного управления состоит из:

1. Автоматизированного рабочего места (8)
   1. ЦПУ: Intel(R) Celeron(R) CPU E3500;
   2. Базовая тактовая частота: 2.7 ГГц;
   3. ОЗУ 2 ГБ;
   4. ПЗУ HDD объемом 500 ГБ;

**Структура сети**

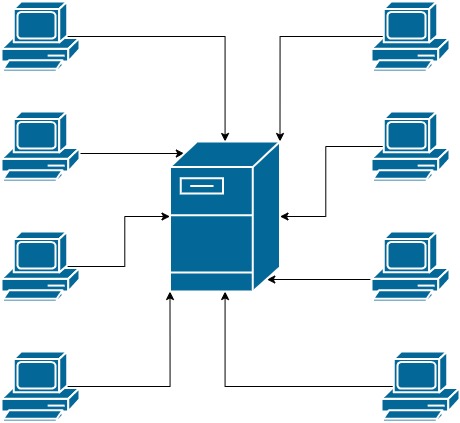


Рисунок 2. Схема сети отдела

## **Характеристика программного обеспечения**

Программное обеспечение центра качества образования состоит из перечня продуктов, представленных в таблице 1.

Таблица 1.

Программное обеспечение

|  |  |  |  |
| --- | --- | --- | --- |
| № | Наименование | Количество лицензий | Вебсайт продукта |
| 1 | MS Windows 7 | 8 | https://www.microsoft.com/ |
| 2 | MS Office 2013 | 8 | https://www.microsoft.com/ |
| 3 | 7-Zip | 8 | https://www.7-zip.org/ |
| 4 | Adobe Acrobat DC | 8 | https://www.adobe.com/ru/ |
| 5 | Adobe Bridge C56 | 8 | https://www.adobe.com/ru/ |
| 6 | Adobe ExtendScript Toolkit C56 | 8 | https://www.adobe.com/ru/ |
| 7 | Adobe Extension Manager C56 | 8 | https://www.adobe.com/ru/ |
| 8 | Adobe Photoshop C56 | 8 | https://www.adobe.com/ru/ |
| 9 | Яндекс.Браузер | 8 | https://www.google.ru/ |
| 10 | TeamViewer 14 | 8 | https://www.teamviewer.com/ru/ |
| 11 | Zoom | 8 | https://zoom.us/ |

## **1.3 Характеристика методов проектирования и разработки программного обеспечения**

В отделе формирования и реализации дистанционного обучения собственные разработки отсутствуют.

В качестве метода проектирования для разрабатываемого проекта было выбрано структурное проектирование. Такой метод необходимо четко структурировать и разбить программу на модули, также требуется разбить все задачи на более маленькие для удобства их решения. Плюсы такого метода в более эффективной разработке и лёгкости последующей поддержки проекта. Примером такого модуля могут быть следующие модули: курс, подборка с курсами, профиль.

В качестве метода разработки была выбрана итеративная модель разработки. При данном методе работы выполняются параллельно с непрерывным анализом полученных результатов и корректировкой последующих этапов работы. Данная модель позволяет постоянно корректировать систему без больших затрат. На рисунке 3 представлена 1-я итерация разработки модуля перехода страниц. На рисунке 2 представлены изменения, внесенные на 2-й итерации.

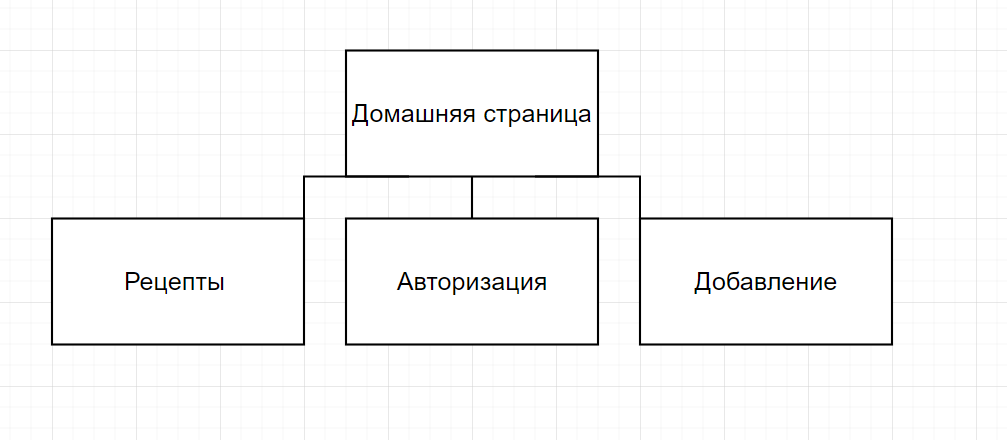


Рисунок 3. Модуль перехода на страницы

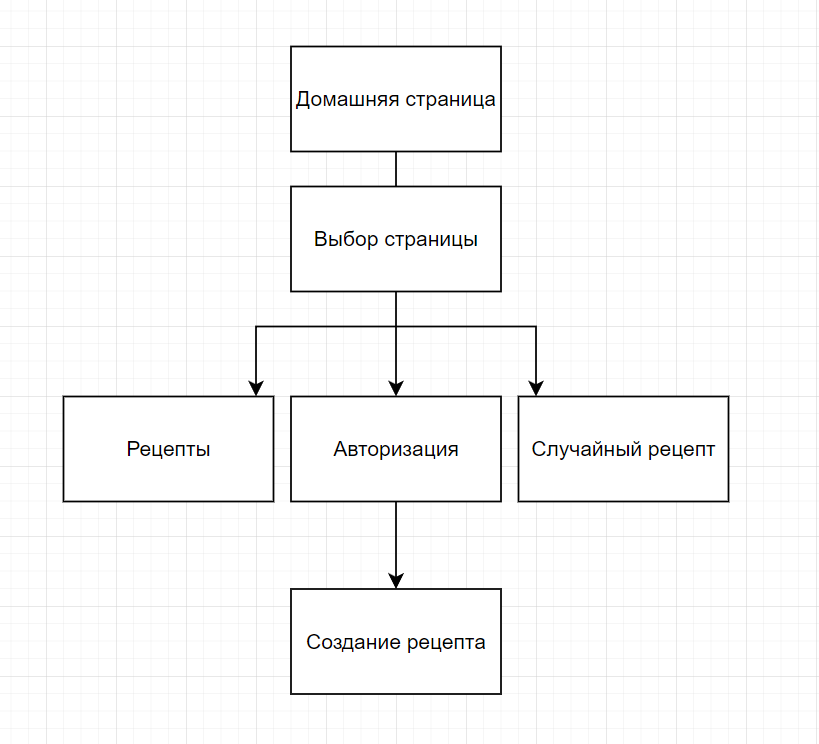
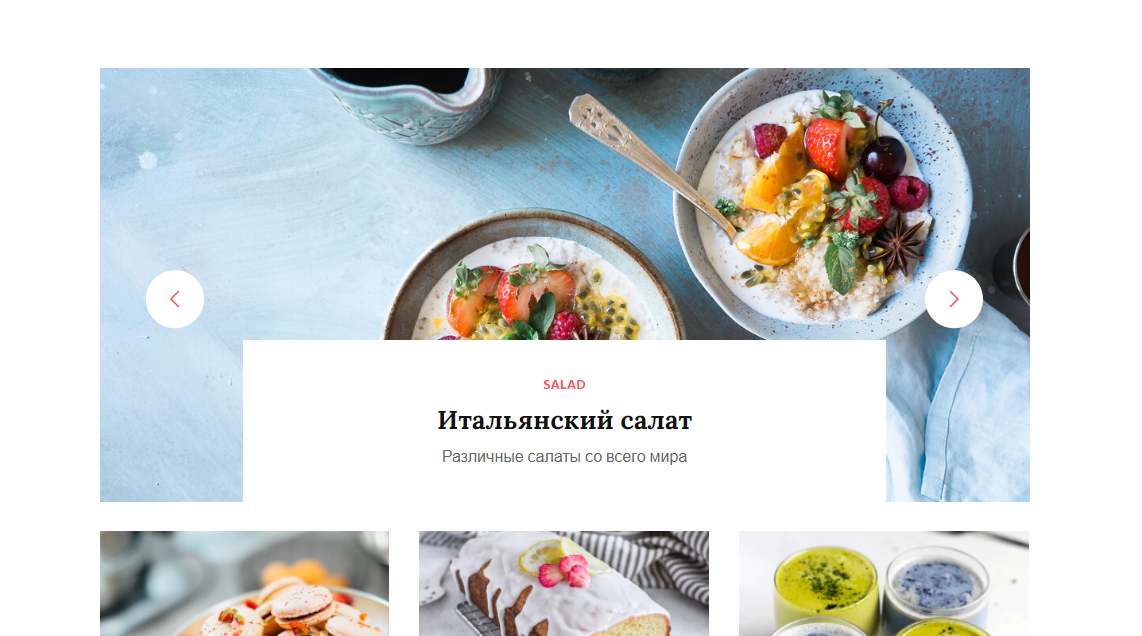
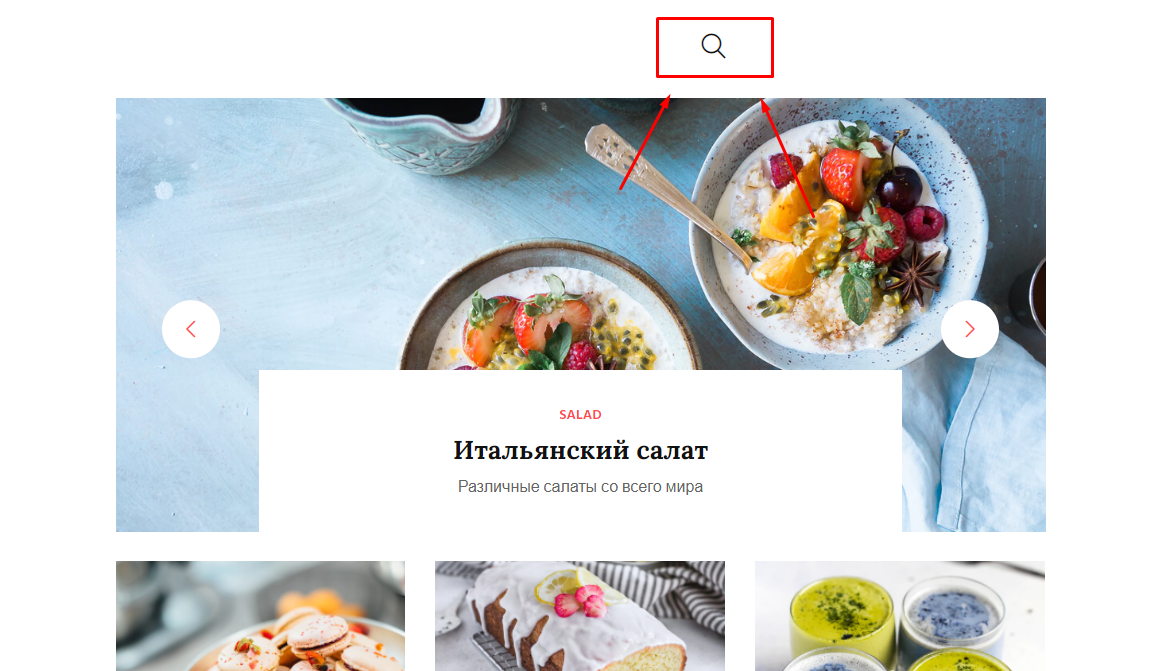


Рисунок 4. Модуль перехода на страницы после изменений

Рисунок 5. До добавления функционала (Главная страница)

Рисунок 6. После добавления функционала (Главная страница)

# Теоретическая часть для ВКР

## **Анализ предметной области**

## **Анализ имеющихся программных решений**

На данный момент, кулинарные сайты — это не редкость. Кулинарные сайты, схожи по функционалу, не в каждом из них есть уникальные решения.

Функционал кулинарных сайтов:

* Поиск рецептов
* Написание собственного рецепта
* Добавление рецепта в свои любимые

### **Примеры сайтов**

**Сайт: Еда.ру**

Этот сайт с достаточно хорошим дизайном, минималистично и понятно(рис1).

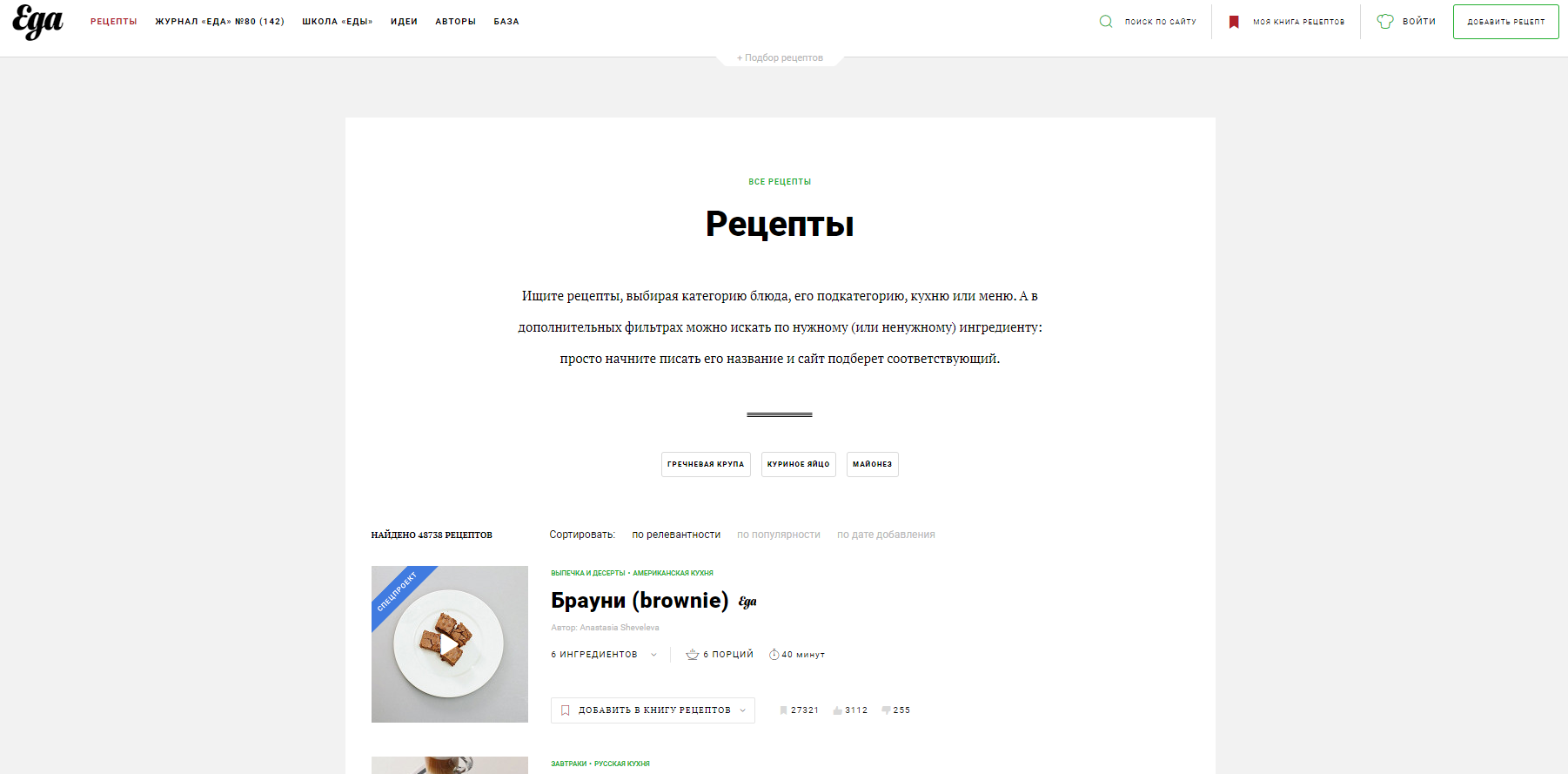


Рисунок 7. Еда.ру главная страница

Плюсы:

* Удобен в использовании
* Показано готовое блюдо

Минусы:

* Нет видео поэтапного приготовления

**Сайт: Готовим Дома**

Данный сайт, может похвастаться видео роликами поэтапного приготовления блюда. Интерфейс сайта, с первого взгляда, не приветлив.

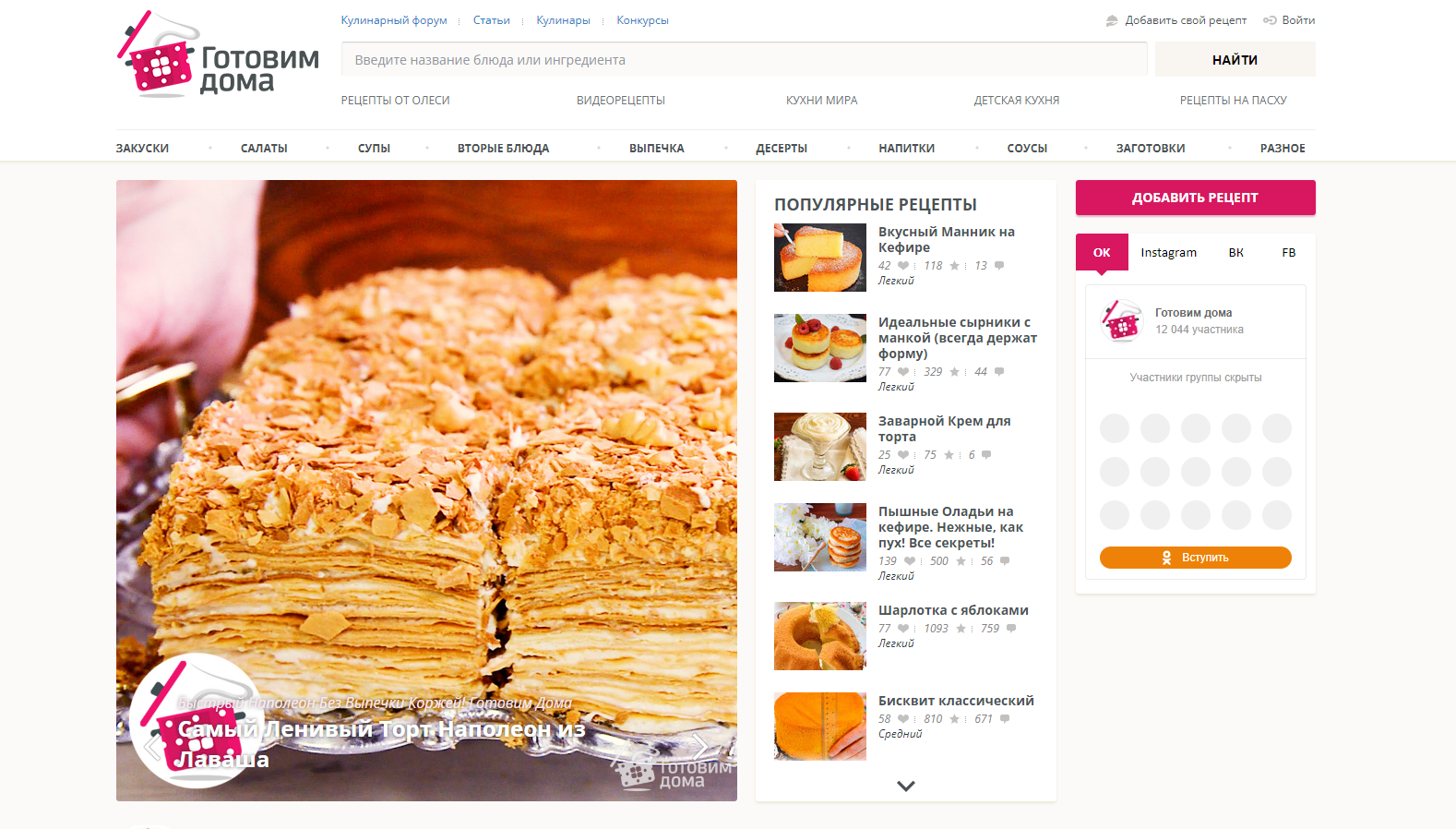


Рисунок 8. Готовим дома главная страница

Плюсы:

* Видео с поэтапной готовкой блюда
* Показано готовое блюдо
* Можно оценивать

Минусы:

* Интерфейс не понятен

**Сайт: Повар.ру**

Данный сайт ничем не отличается от других платформ, он вобрал в себя все функции, которые существуют на других платформах.

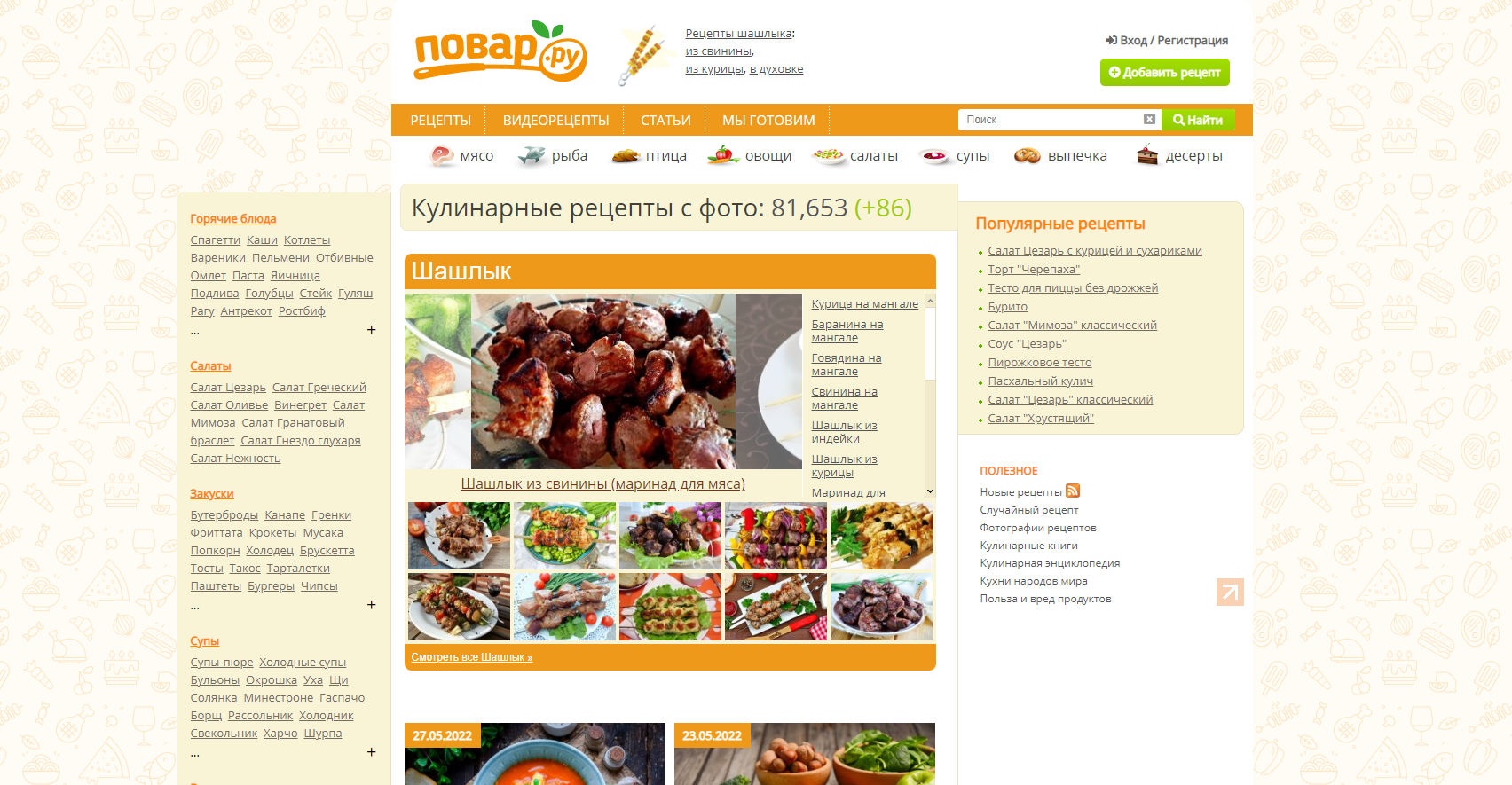


Рисунок 8. Повар.ру главная страница

Плюсы:

* Удобен в использовании
* Показано готовое блюдо
* Можно оценивать
* Видео поэтапного приготовления блюда
* Можно провести анализ этих разработок и оценить целесообразность разработки собственного аналога. (оценки по 10-бальной шкале)

Таблица 1

|  |  |  |  |
| --- | --- | --- | --- |
|  | Дизайн | Скорость работы сайта | Функционал |
| Повар.ру | 8 | 7 | 10 |
| Еда.ру | 10 | 8 | 8 |
| Готовим Дома | 6 | 7 | 8 |
| СоосMaster | 10 | 5 | 10 |

## **Разработка бизнес-модели обработки данных**

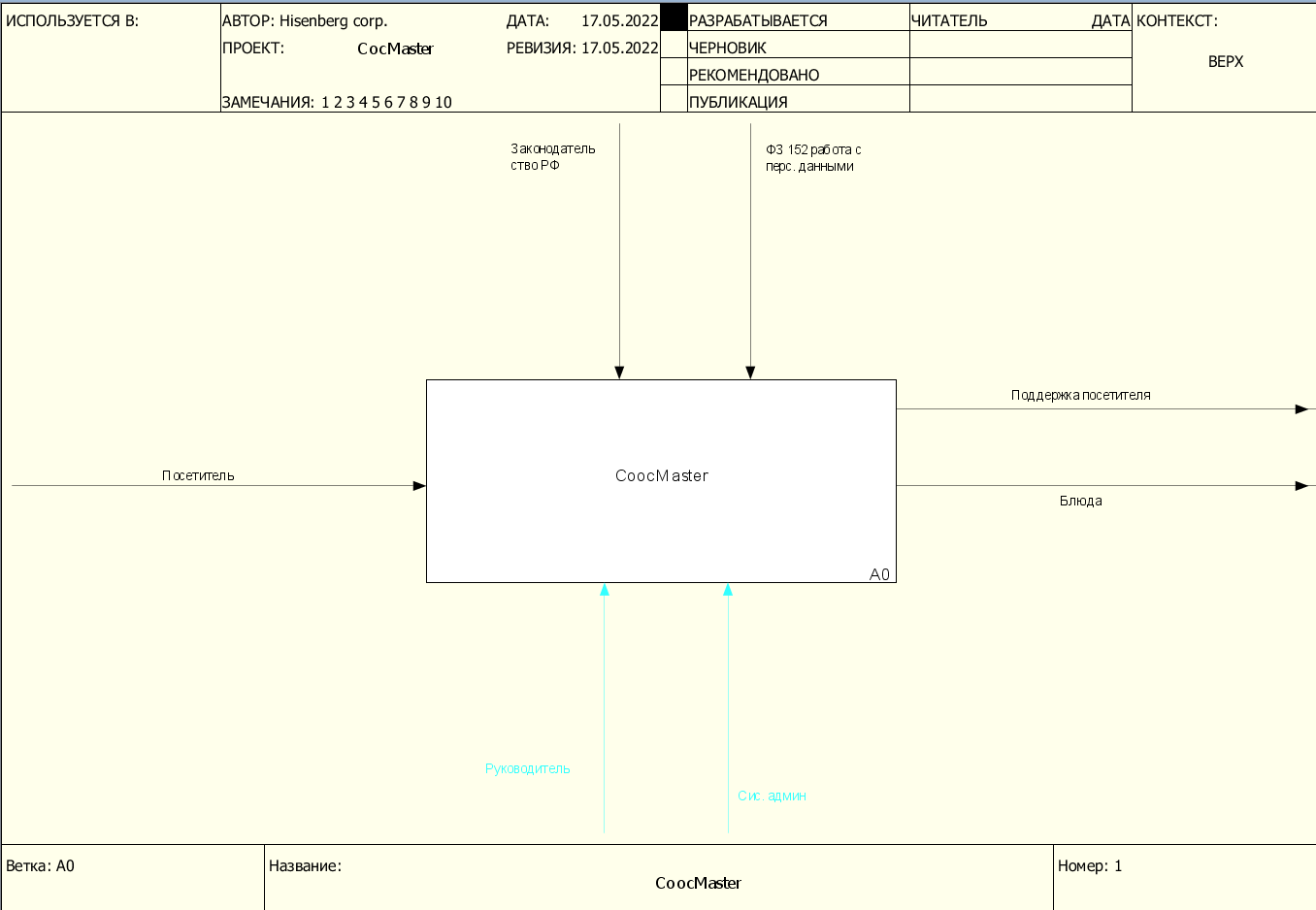


Рисунок 9. Первый уровень IDEF

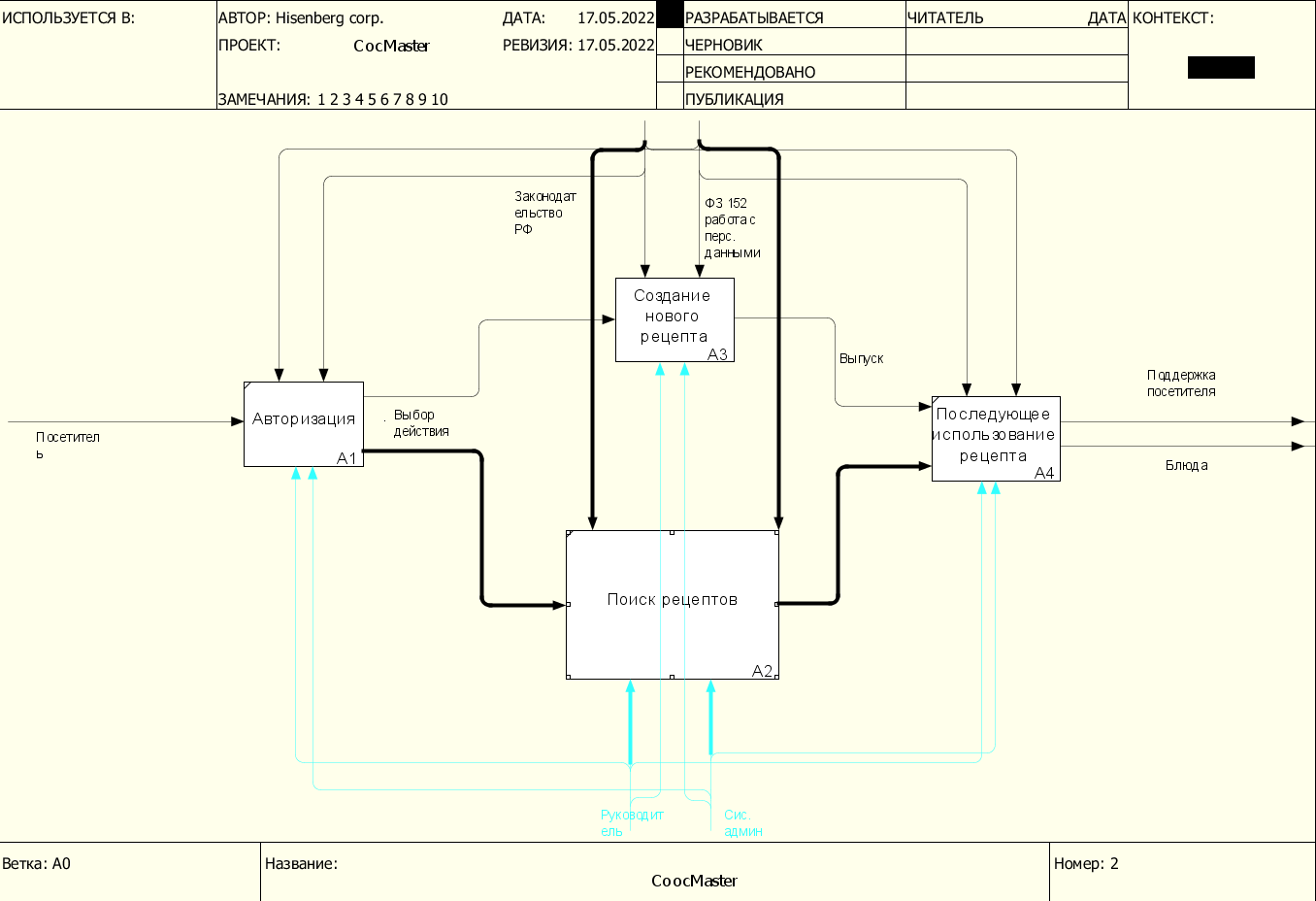


Рисунок 10. Второй уровень IDEF

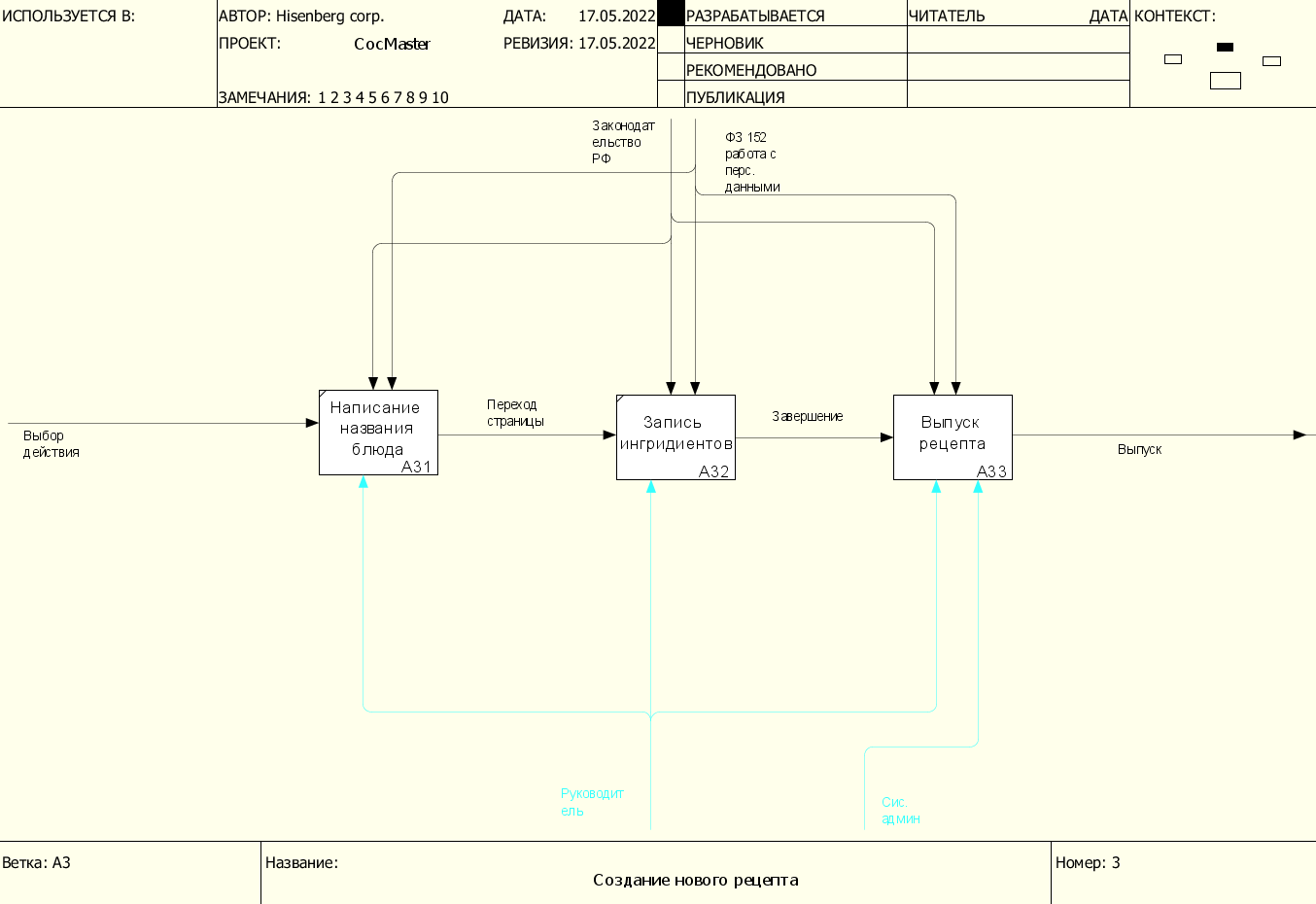


Рисунок 11. Третий уровень IDEF

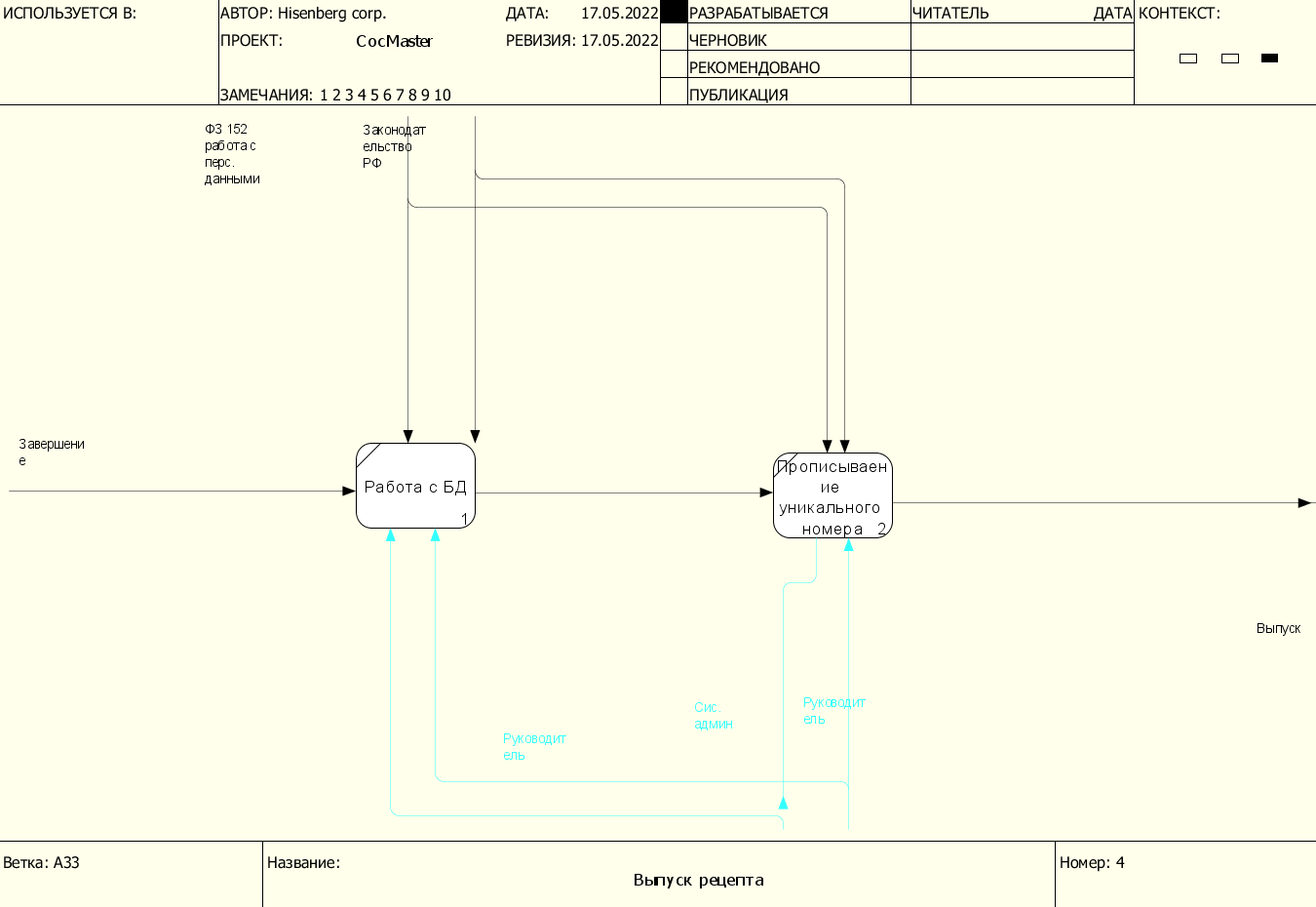


Рисунок 12. Уровень DFD

## **Разработка алгоритм решения поставленной задачи**

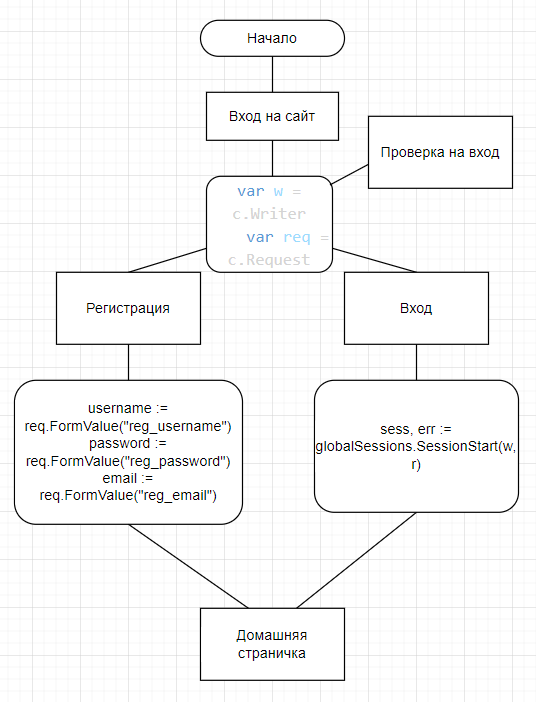


Рисунок 13. Алгоритм авторизации на сайте

* 1. **Обоснование выбора инструментов**

Для разработки Back-end модуля в основном используются язык программирования Golang.

Back-end модуль должен работать с Базой Данных, поэтому необходимо выбрать подходящею СУБД. Выбиралась СУБД из 3-х следующих: MySQL, PostgreSQL, MS SQL.

Таблица 2. Сервер

|  |  |  |  |
| --- | --- | --- | --- |
| Критерий \ Сервер | MSSQL | MySQL | PostgreSQL |
| Опыт работы с сервером | - | + | - |

После выбора Golang, предстал выбор перед Фреймворками Gin и Echo. Был проведён анализ и выбран лучший из них

Таблица 3. Фреймворки

|  |  |  |  |
| --- | --- | --- | --- |
|  | Скорость разработки | Простота кода | Современность |
| Gin | + | + | + |
| Echo | + | - | + |

Таблица 4. IDE для Golang

|  |  |  |
| --- | --- | --- |
|  | VS Code | Atom |
| Возможность скачивать аддоны для облегчения работы | + | - |
| Удобство при работе с несколькими модулями | + | + |
| Возможность просмотреть каталог библиотек | + | - |

## **Разработка структуры главного модуля**

**Создание прототипа интерфейса**

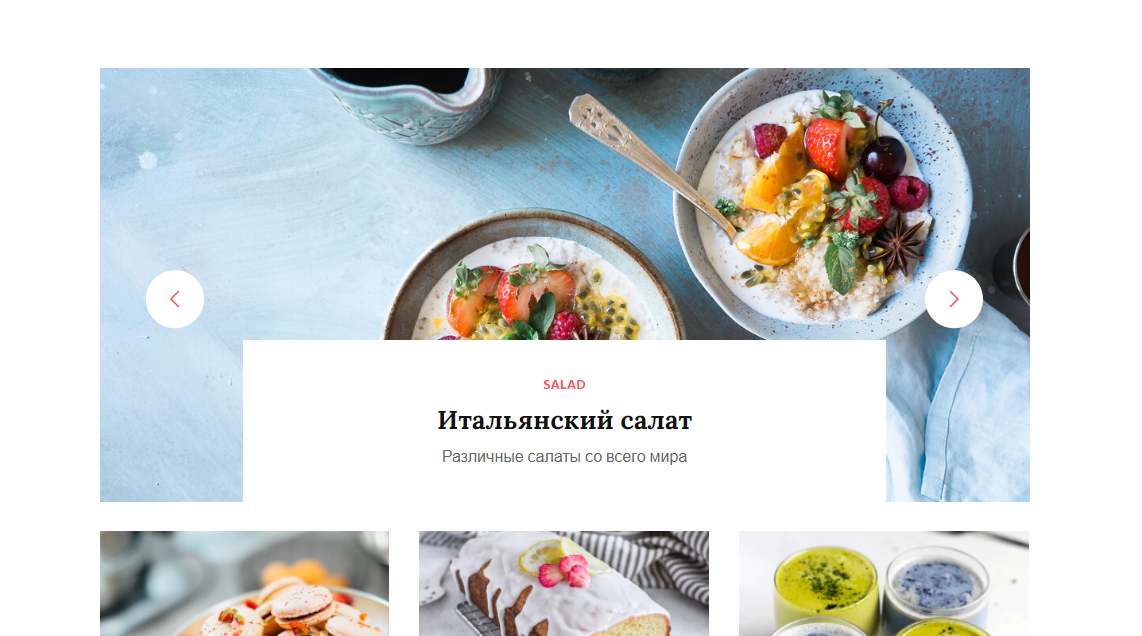


Рисунок 14. Прототип интерфейса – Часть 1

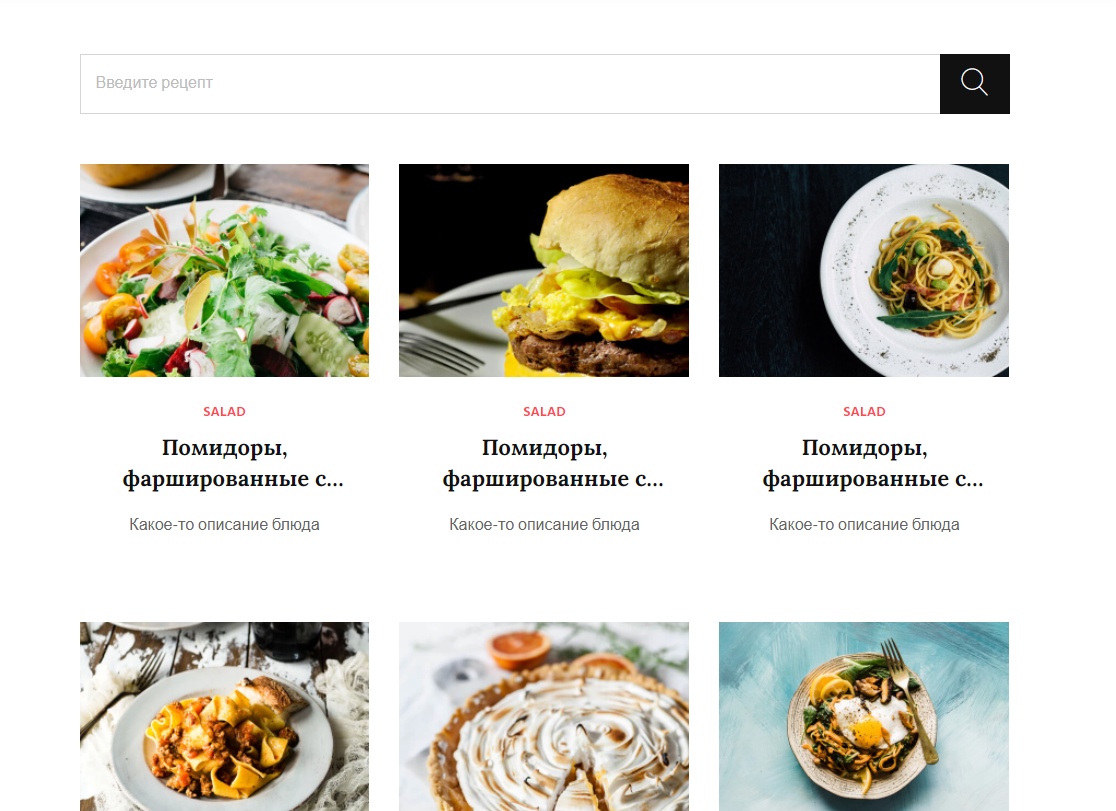


Рисунок 15. Прототип интерфейса – Часть 2

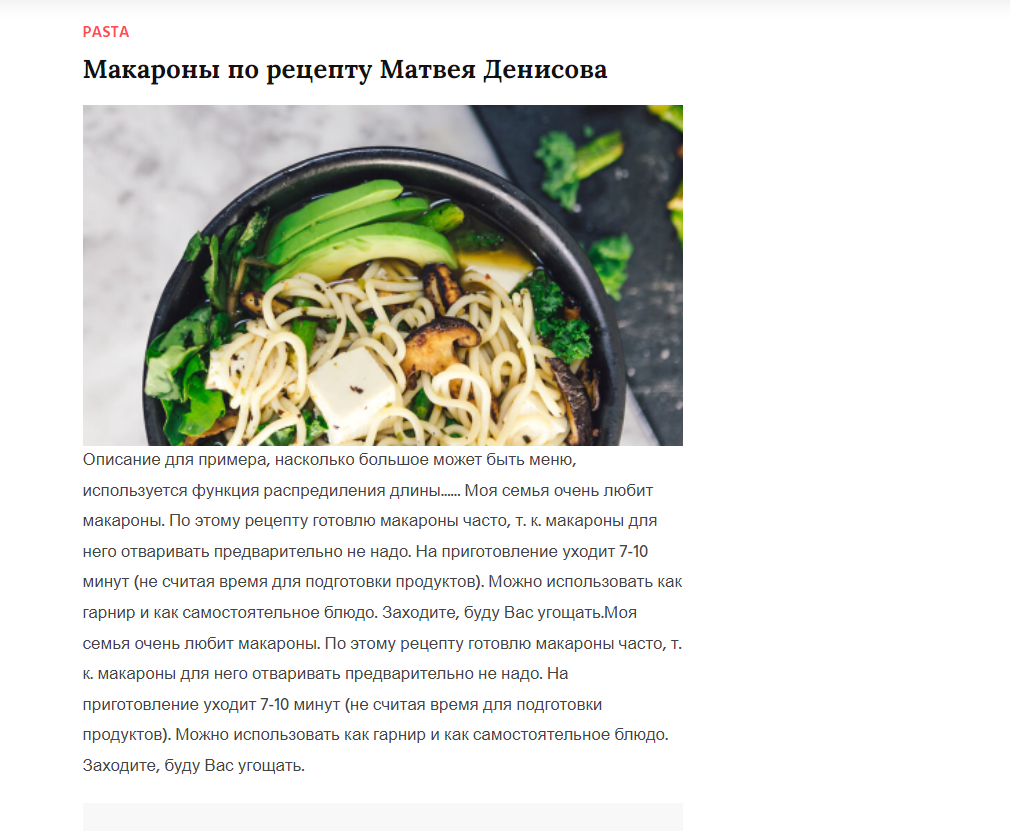


Рисунок 16. Прототип интерфейса – Часть 3

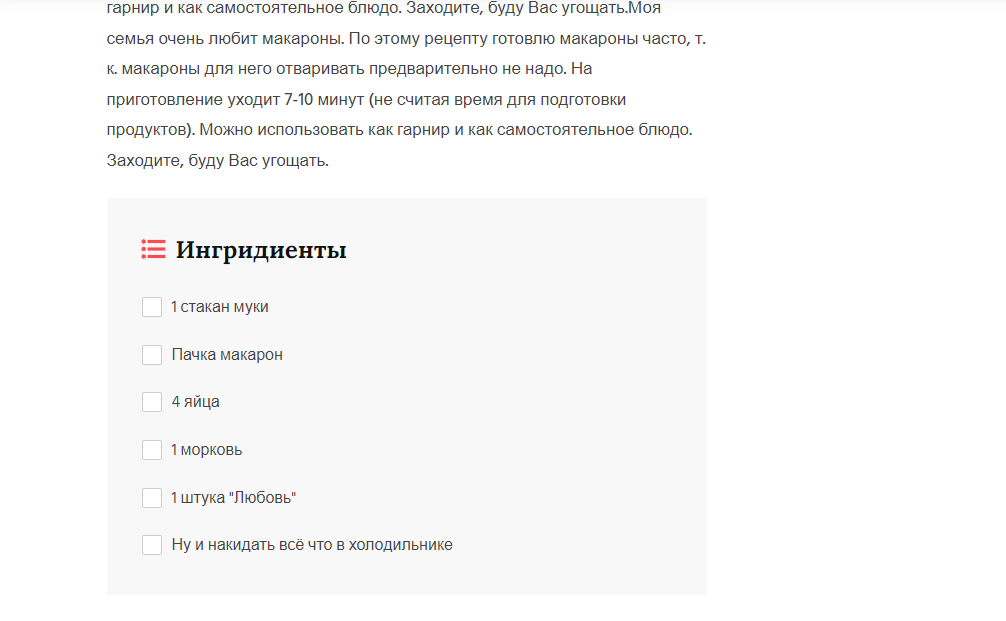


Рисунок 17. Прототип интерфейса – Часть 4

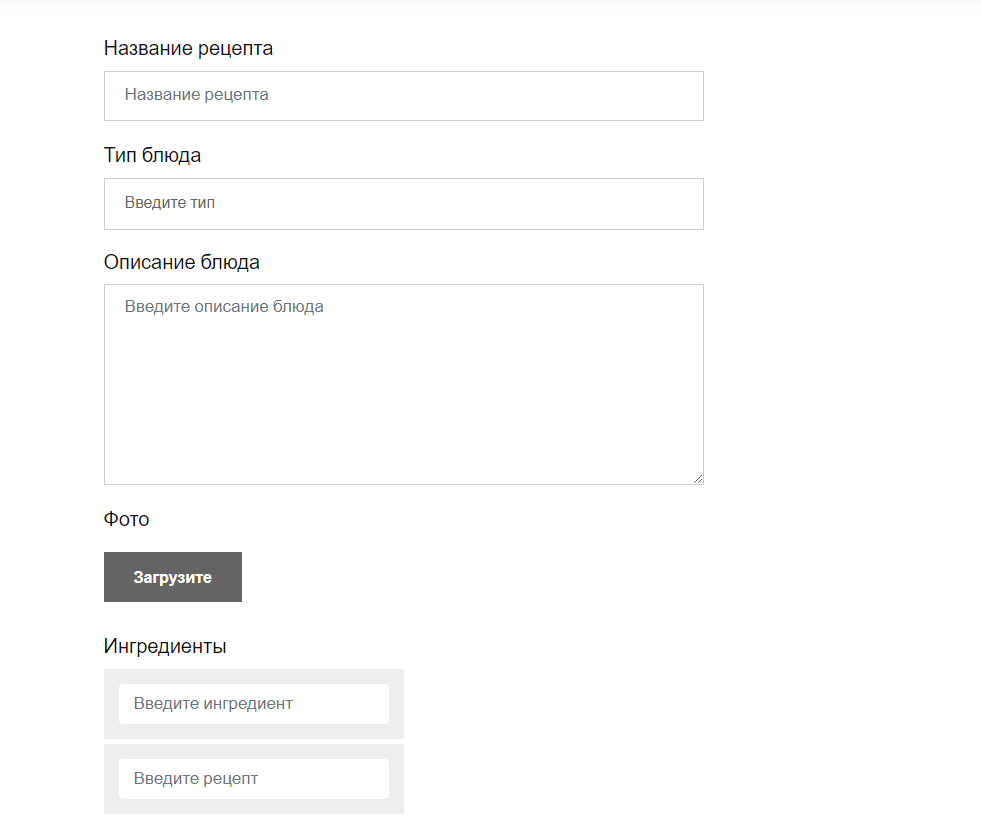


Рисунок 18. Прототип интерфейса – Часть 5

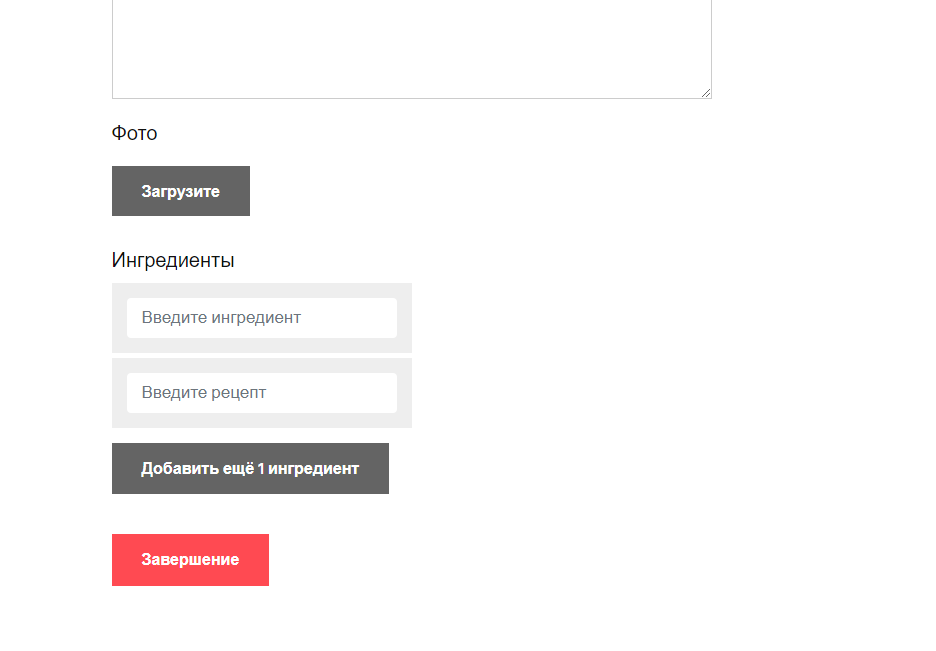


Рисунок 19. Прототип интерфейса – Часть 6

## **Разработка структуры БД**

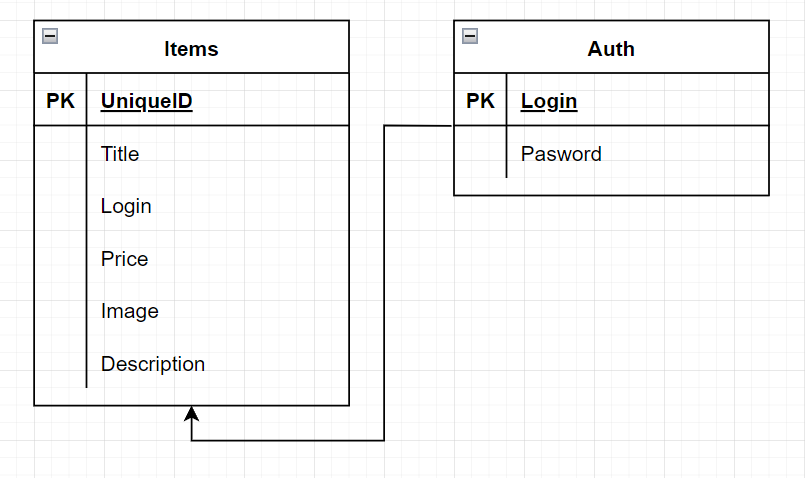


Рисунок 20. ER- Модель БД

# Организационно-экономическая часть для ВКР

## **Раздел техники безопасности**

**Техника безопасности при работе с компьютером**

Правильный выбор рабочего места – дисплей (монитор) является источником электромагнитного излучения. Рекомендуется устанавливать защитный экран для снижения воздействия электромагнитного излучения от задней части другого дисплея. Недопустимо устраивать рабочие места близко одно от другого. Размещать компьютер необходимо вдали от отопительных приборов и исключать попадания на него прямых солнечных лучей. Недопустимо работать напротив боковой или задней части другого дисплея, если расстояние до него - менее 2 м.

1. Системный блок – помещается на надежную поверхность (крепкий стол, массивная подставка/тумба) – так, чтобы исключать даже случайное его сотрясение;
2. Дисплей необходимо устанавливать на такой высоте, чтобы центр экрана был на 15-20 см ниже уровня глаз. Расстояние от глаз до экрана – не менее 50 см;
3. Клавиатура располагается на расстоянии 15-30 см от края столешницы или на специальной выдвижной доске.

Не рекомендуется установка компьютера (и клавиатуры) на поверхность, которая создает блики и легко «собирает» статическое электричество (оргстекло, полированная и покрытая лаком доска).

Необходимо следить, чтобы бумаги, какие-либо предметы не закрывали

вентиляционные отверстия работающих аппаратов.

1. Кресло – ширина и глубина сиденья не менее 40см.; спинка: высота опорной поверхности 30±2 см; ширина не менее 38 см.; подлокотники: длина не менее 25 см; ширина 5-7 см., высота над сиденьем 23+3 см.;
2. Стол – размеры рабочей поверхности (столешницы): длина – 80-120 см; ширина – 80-100 см.; высота (расстояние от пола до рабочей поверхности) 68-85 см; оптимальная высота 72,5 см.;
3. Подставки – для рук: опорная планка для запястья («подзапястник») - плоская или изогнутая пластина из мягкого материала; помещается перед клавиатурой. Для ног: ширина не менее 30 см; длина (глубина) не менее 40 см.
4. Площадь одного рабочего места с компьютером - не менее 6 м2;
5. Освещение должно быть естественным и искусственным.

Рекомендуется работать в помещении, где окна выходят на север или северо-восток. Местное освещение не должно создавать блики на поверхности экрана дисплея. Недопустим яркий нерассеянный верхний свет (с потолка). Сдерживать поток избыточного света от окон следует с помощью жалюзи (или тканевых штор);

1. Чистота обязательна при работе за компьютером. Влажную уборку помещения следует проводить ежедневно. Недопустима запыленность воздуха, пола, рабочей поверхности стола и техники. Помещение должно быть оборудовано системами вентиляции, кондиционирования и отопления. Запрещается работа на компьютере в подвальных помещениях.
2. Температура воздуха - от 21 до 25°С (в холодное время года); от 23 до 25˚С (в теплое время года);
3. Влажность воздуха (относительная) - от 40 до 60%. Недопустимы резкие перепады температуры и влажность воздуха более 75%;
4. Ионизация воздуха – образующиеся в помещении положительно заряженные ионы очень вредны для здоровья, вызывают быстрое утомление, головную боль, учащение пульса и дыхания (из-за недостаточного поступления кислорода в кровь). Специальные устройства - аэроионизаторы - нормализуют аэроионный режим, увеличивая концентрацию легких отрицательно заряженных ионов (воздух становится живительным, как в горах или на море). Перед включением аэроионизатора следует очистить воздух от пыли и аэрозолей. В больших помещениях используют аэроионизаторы типа люстры Чижевского, в небольшой комнате (15-20 м2) достаточно установить переносной аэроионизатор (некоторые модифицированные аппараты позволяют и очищать, и ионизировать воздух).

Необходимо в начале работы включать общее питание, периферийные устройства, системный блок, в конце работы наоборот – выключать системный блок, периферийные устройства, общее питание.

Не обязательно выключать компьютер на время небольших перерывов в работе.

Перед подсоединением/отсоединением устройств ввода-вывода требуется полностью отключать эту технику и компьютер от электросети.

При появлении запаха гари или при обнаружении повреждения изоляции, обрыва провода следует немедленно отключить устройства (лучше – общее электропитание) и сообщить о неисправности руководителю.

Прикасаться к задней панели работающего системного блока (процессора) запрещается.

Недопустимо попадание влаги на системный блок, дисплей, клавиатуру и другие устройства (лучше не ставить рядом чашку с чаем или кофе, стакан с соком).

При интенсивной работе резко возрастает напряженность электрического поля на клавиатуре и «мыши». От трения рук о них через 0,5- 1 час работы электростатический потенциал достигает 10-20 кВ/м, что оказывает на организм вредное воздействие. Работать с «мышью» нужно на специальном коврике. Необходимо несколько раз в течение рабочей смены (лучше ежечасно) смачивать руки водой, чтобы снимать электростатические заряды, а по завершении работы вымыть с мылом руки и лицо, ежедневно протирать влажной салфеткой экран дисплея, клавиатуру, «мышь» - убирать статическое электричество, регулярно проводить влажную антистатическую обработку помещения.

Правильная рабочая поза позволяет избегать перенапряжения мышц, способствует лучшему кровотоку и дыханию. Следует сидеть прямо (не сутулясь) и опираться спиной о спинку кресла. Прогибать спину в поясничном отделе нужно не назад, а, наоборот, немного вперед. Недопустимо работать, развалившись в кресле. Такая поза вызывает быстрое утомление, снижение работоспособности.

Чтобы не травмировать позвоночник, важно:

1. Избегать резких движений;
2. Поднимаясь/садясь, держать голову и торс прямо.

Необходимо найти такое положение головы, при котором меньше напрягаются мышцы шеи. Рекомендуемый угол наклона головы - до 20°. В этом случае значительно снижается нагрузка на шейные позвонки и на глаза. Во время работы за компьютером необходимо расслабить руки, держать предплечья параллельно полу, на подлокотниках кресла, кисти рук - на уровне локтей или немного ниже, запястья - на опорной планке. Тогда пальцы получают наибольшую свободу передвижения.

Не следует высоко поднимать запястья и выгибать кисти - это может стать причиной боли в руках и онемения пальцев. Можно надевать легкие перчатки без пальцев, если стынут руки.

Колени должны располагаться на уровне бедер или немного ниже. При таком положении ног не возникает напряжение мышц.

Нельзя скрещивать ноги, класть ногу на ногу - это нарушает циркуляцию крови из-за сдавливания сосудов. Лучше держать обе стопы на подставке или на полу. Необходимо сохранять прямой угол (90°) в области локтевых, тазобедренных, коленных и голеностопных суставов.

1. Дышать ритмично, свободно, глубоко, чтобы обеспечить кислородом все части тела;
2. Держать в расслабленном состоянии плечи и руки - в руках не будет напряжения, если плечи опущены;
3. Чаще моргать и смотреть вдаль. Моргание способствует не только увлажнению и очищению поверхности глаз, но и расслаблению лицевых, лобных мышц (без сдвигания бровей). Малая подвижность и длительное напряжение глазных мышц могут стать причиной нарушения аккомодации.

При ощущении усталости какой-либо части тела необходимо сделать глубокий вдох и сильно напрячь уставшую часть тела, после чего задержать дыхание на 3-5 секунды и на выдохе расслабиться; можно повторить.

При ощущении усталости глаз нужно в течение 2-3 мин окинуть взглядом комнату, устремлять взгляд на разные предметы, смотреть вдаль (в окно).

Если резко возникло общее утомление, появилось дрожание изображения на экране дисплея (покачивание, подергивание, рябь), следует сообщить об этом руководителю.

1. Негативное влияние на зрение – зрительная система человека приспособлена для восприятия объектов в отраженном свете (картин природы, рисунков, печатных текстов и т.д.), а не для работы с дисплеем. Изображение на дисплее принципиально отличается от привычного глазу объектов наблюдения – оно светится, состоит из дискретных точек; оно мерцает, т.е. эти точки с определенной частотой зажигаются и гаснут; цветное компьютерное изображение не соответствует естественным цветам. При работе за компьютером часами у глаз не бывает необходимых фаз расслабления, глаза напрягаются, их работоспособность снижается.
2. Микротравма – это постепенный износ организма в результате ежедневных нагрузок. Большинство нарушений в организме происходит из-за накапливающихся микротравм.
3. Заболевания, вызванные повторяющимися нагрузками: повторяющиеся действия, приводят к накоплению продуктов распада в мышцах. Эти продукты и вызывают болезненные ощущения
4. Неудобное сидячее положение, что приводит к неправильной осанке, сутулости.
5. Умственное перенапряжение и эмоциональные перегрузки.
6. Монотонность труда.

# Литература и интернет источники

1. <https://habr.com/ru/post/145381/> - основы реляционной алгебры
2. <https://i.voenmeh.ru/kafi5/Kam.loc/BaseDate/Lekcii6_RelAlgebra.html> - лекция по реляционной алгебре
3. https://app.diagrams.net/- сайт создания диаграмм
4. https://www.mysql.com/ – скачивание MySQL
5. https://go.dev/ - скачивание Golang
6. https://code.visualstudio.com/download – скачивание Visual Studio Code

Приложение 1. Листинг 1. Routes.go

package main

import (

"database/sql"

"fmt"

"log"

"net/http"

\_ "github.com/lib/pq"

"github.com/plimble/ace"

//"github.com/astaxie/beego/session"

"github.com/unrolled/render"

"golang.org/x/crypto/bcrypt"

)

func PostLogin(c \*ace.C) {

var w = c.Writer

var req = c.Request

sess, \_ := globalSessions.SessionStart(w, req)

defer sess.SessionRelease(w)

sessname := sess.Get("email")

cartsess := sess.Get("cart")

var cart []string

fmt.Println(sessname)

fmt.Println(cartsess)

var password\_in\_database string

var email string

badstr := "Authorization Failed"

username, password := req.FormValue("inputUsername"), req.FormValue("inputPassword")

err := db.QueryRow("SELECT email, password FROM users WHERE username=$1", username).Scan(&email, &password\_in\_database)

if err == sql.ErrNoRows {

BadPage(c, badstr)

} else if err != nil {

log.Print(err)

BadPage(c, badstr)

}

err = bcrypt.CompareHashAndPassword([]byte(password\_in\_database), []byte(password))

if err == bcrypt.ErrMismatchedHashAndPassword {

BadPage(c, badstr)

} else if err != nil {

log.Print(err)

BadPage(c, badstr)

}

if err == nil {

err2 := sess.Delete("email")

err2 = sess.Set("email", email)

err2 = sess.Set("cart", cart)

log.Println(req)

if err2 == nil {

c.Redirect("/")

}

}

}

func PostSignup(c \*ace.C) {

var w = c.Writer

var req = c.Request

username := req.FormValue("reg\_username")

password := req.FormValue("reg\_password")

email := req.FormValue("reg\_email")

hashedPassword, err := bcrypt.GenerateFromPassword([]byte(password), 10)

if err != nil {

panic(err)

}

\_, err = db.Exec("INSERT INTO users (username, password, email) VALUES ($1, $2, $3)", username, string(hashedPassword), email)

if err != nil {

log.Print(err)

}

http.Redirect(w, req, "/login", 302)

}

func Logout(c \*ace.C) {

var w = c.Writer

var req = c.Request

sess, \_ := globalSessions.SessionStart(w, req)

globalSessions.SessionDestroy(w, req)

err := sess.Delete("email")

if err == nil {

http.Redirect(w, req, "/", 302)

}

}

func ShowItemsHome(w http.ResponseWriter, r \*http.Request) {

// Loop through rows using only one struct

items := []Item{}

err := db.Select(&items, "SELECT \* FROM items ORDER BY id ASC")

if err != nil {

log.Println("Is this the problem?")

log.Println(err)

return

}

log.Println("%#v\n", items)

render := render.New(render.Options{

IndentJSON: true,

})

render.HTML(w, http.StatusOK, "home", map[string]interface{}{

"item0": items[0],

"item1": items[1],

"item2": items[2],

})

}

func ShowItemsPages(c \*ace.C, i int) {

var w = c.Writer

// Loop through rows using only one struct

items := []Item{}

err := db.Select(&items, "SELECT \* FROM items ORDER BY id ASC")

if err != nil {

log.Println("Is this the problem?")

log.Println(err)

return

}

max := len(items)

factor := i - 1

one, two, three := factor\*3, (factor\*3)+1, (factor\*3)+2

log.Println("%d\n %d\n %d\n", one, two, three)

log.Println(max)

render := render.New(render.Options{

IndentJSON: true,

})

if one < max {

if max%3 == 0 {

render.HTML(w, http.StatusOK, "home", map[string]interface{}{

"item0": items[one],

"item1": items[two],

"item2": items[three],

})

} else if max%3 == 2 {

render.HTML(w, http.StatusOK, "home", map[string]interface{}{

"item0": items[one],

"item1": items[two],

})

} else if max%3 == 1 {

render.HTML(w, http.StatusOK, "home", map[string]interface{}{

"item0": items[one],

})

}

} else {

c.Redirect("/")

}

}

func ShowItemid(c \*ace.C, s string) {

var w = c.Writer

var r = c.Request

fmt.Println(s)

itemid := Item{}

err := db.Get(&itemid, "SELECT \* FROM items WHERE id=$1", s)

if err != nil {

log.Print("This must be the problem")

}

if itemid.Id == 0 {

http.Redirect(w, r, "/", 302)

}

render := render.New(render.Options{

IndentJSON: true,

})

render.HTML(c.Writer, http.StatusOK, "item", &itemid)

}

Приложение 1. Листинг 2. search.go

package main

import (

"log"

"net/http"

"strings"

\_ "github.com/lib/pq"

"github.com/plimble/ace"

"github.com/unrolled/render"

)

func Substring(s string, substring string) bool {

lowers := strings.ToLower(s)

lowersubstring := strings.ToLower(substring)

compare := strings.Contains(lowers, lowersubstring)

tf := false

if compare {

tf = true

}

return tf

}

func Compare(arr []Item, query string) (bool, []Item) {

var tf bool

results := []Item{}

var sub bool

var str bool

for \_, elem := range arr {

sub = Substring(elem.Title, query)

str = strings.EqualFold(elem.Title, query)

log.Println(sub, str)

if sub || str {

tf = true

results = append(results, elem)

}

}

return tf, results

}

func Search(c \*ace.C, i int) {

r := c.Request

usersearch := r.FormValue("usersearch")

itemsearch := []Item{}

err := db.Select(&itemsearch, "SELECT \* FROM items ORDER BY id ASC")

if err != nil {

log.Println("Is this the problem?")

log.Println(err)

return

}

compared, results := Compare(itemsearch, usersearch)

log.Println("%#v\n", results)

max := len(results)

factor := i - 1

one, two, three := factor\*3, (factor\*3)+1, (factor\*3)+2

if compared && max < 4 {

render := render.New(render.Options{})

if max == 3 {

render.HTML(c.Writer, http.StatusOK, "results", map[string]interface{}{

"result0": results[one],

"result1": results[two],

"result2": results[three],

})

} else if max == 2 {

render.HTML(c.Writer, http.StatusOK, "results", map[string]interface{}{

"result0": results[one],

"result1": results[two],

})

} else if max == 1 {

render.HTML(c.Writer, http.StatusOK, "results", map[string]interface{}{

"result0": results[one],

})

}

} else {

c.Redirect("/noresults")

}

}

Приложение 1. Листинг 3. rendering.go

package main

import (

"fmt"

"io/ioutil"

"log"

"net/http"

"path/filepath"

"strings"

"github.com/plimble/ace"

"github.com/unrolled/render"

)

func SimplePage(w http.ResponseWriter, req \*http.Request, template string) {

r := render.New(render.Options{})

r.HTML(w, http.StatusOK, template, nil)

}

func SimpleAuthenticatedPage(c \*ace.C, template string, data interface{}) {

w := c.Writer

r := c.Request

sess, err := globalSessions.SessionStart(w, r)

defer sess.SessionRelease(c.Writer)

sessname := sess.Get("email")

log.Println(sessname)

if err != nil {

log.Println(err)

} else {

r := render.New(render.Options{})

r.HTML(c.Writer, http.StatusOK, template, data)

}

}

func fileLoadHandler(c \*ace.C) {

url := c.Param("cssfile")

str := strings.TrimLeft(url, ":")

w := c.Writer

baseDir, \_ := filepath.Abs("/Users/asdasd/golang/src/webshop/public/css/")

fileName := "/" + str

file, err := ioutil.ReadFile(fmt.Sprintf("%s%s", baseDir, fileName))

w.Header().Set("Content-Type", "text/css")

fmt.Fprint(w, string(file))

if err != nil {

fmt.Println("Error reading file: ")

fmt.Println(err)

}

}

Приложение 1. Листинг 4. pages.go

package main

import (

"fmt"

"log"

"net/http"

"strings"

"github.com/plimble/ace"

"github.com/unrolled/render"

)

func Home(c \*ace.C) {

var w = c.Writer

var r = c.Request

sess, err := globalSessions.SessionStart(w, r)

defer sess.SessionRelease(w)

err = sess.Delete("itemid")

log.Println(err)

ShowItemsHome(w, r)

}

func Login(c \*ace.C) {

var w = c.Writer

var r = c.Request

sess, err := globalSessions.SessionStart(w, r)

defer sess.SessionRelease(w)

err = sess.Delete("itemid")

log.Println(err)

SimplePage(w, r, "try")

}

func Authfail(c \*ace.C) {

log.Print("Authorization Failed")

}

func UserPage(c \*ace.C) {

SimpleAuthenticatedPage(c, "user", nil)

}

func Itemid(c \*ace.C) {

req := c.Request

w := c.Writer

url := c.Param("id")

str := strings.Trim(url, ":")

fmt.Println(str)

sess, err := globalSessions.SessionStart(w, req)

defer sess.SessionRelease(w)

sessname := sess.Get("itemid")

log.Println(sessname)

if err == nil {

err = sess.Set("itemid", str)

ShowItemid(c, str)

}

}

func Signup(c \*ace.C) {

var w = c.Writer

var r = c.Request

sess, err := globalSessions.SessionStart(w, r)

defer sess.SessionRelease(w)

err = sess.Delete("itemid")

log.Println(err)

SimplePage(w, r, "signup")

}

func Pagination(c \*ace.C) {

var w = c.Writer

var r = c.Request

sess, err := globalSessions.SessionStart(w, r)

defer sess.SessionRelease(w)

err = sess.Delete("itemid")

url := c.Param("pg")

str := strings.Trim(url, ":")

var b int

if \_, err = fmt.Sscanf(str, "%5d", &b); err == nil {

log.Println(err)

ShowItemsPages(c, b)

}

}

func DisplaySearch(c \*ace.C) {

var w = c.Writer

var r = c.Request

sess, err := globalSessions.SessionStart(w, r)

defer sess.SessionRelease(w)

err = sess.Delete("itemid")

url := c.Param("pg")

str := strings.Trim(url, ":")

var b int

if \_, err = fmt.Sscanf(str, "%5d", &b); err == nil {

Search(c, b)

}

}

func Noresults(c \*ace.C) {

var w = c.Writer

var r = c.Request

sess, err := globalSessions.SessionStart(w, r)

defer sess.SessionRelease(w)

err = sess.Delete("itemid")

log.Println(err)

log.Println("login probz")

SimplePage(w, r, "results")

}

func CartPage(c \*ace.C) {

Cart(c)

}

func BuyPage(c \*ace.C) {

Buy(c)

}

func BadPage(c \*ace.C, str string) {

var w = c.Writer

render := render.New(render.Options{})

render.HTML(w, http.StatusOK, "badpage", str)

}

Приложение 1. Листинг 5. profile.go

package main

import (

"fmt"

"log"

"net/http"

"strconv"

"strings"

"github.com/drone/go.stripe"

"github.com/plimble/ace"

"github.com/unrolled/render"

)

func AddToCart(c \*ace.C) {

var w = c.Writer

var r = c.Request

sess, err := globalSessions.SessionStart(w, r)

defer sess.SessionRelease(w)

user := sess.Get("email")

item := sess.Get("itemid")

cart := sess.Get("cart")

if cartstr, ok := cart.([]string); ok {

if itemstr, ok2 := item.(string); ok2 {

cartstr = append(cartstr, itemstr)

}

if err == nil && user != nil {

err = sess.Set("cart", cartstr)

if err == nil {

c.Redirect("/cart")

} else {

badstr := "You need to be signed in to add to cart"

BadPage(c, badstr)

}

}

} else {

badstr := "You need to be signed in to add to cart"

BadPage(c, badstr)

}

}

func DeleteFromCart(c \*ace.C) {

w := c.Writer

r := c.Request

sess, err := globalSessions.SessionStart(w, r)

defer sess.SessionRelease(w)

log.Println(sess)

deleteid := r.FormValue("ajax\_post\_data")

log.Println(deleteid)

cartcookie := sess.Get("cart")

log.Println(cartcookie)

if a, ok := cartcookie.([]string); ok {

log.Println(a)

var i int

if \_, err := fmt.Sscanf(deleteid, "%5d", &i); err == nil {

aints := Stringtoint(a)

for \_, elem := range aints {

if elem == i {

elempos := aints.pos(elem)

a[elempos] = a[len(a)-1]

a = a[:len(a)-1]

}

}

}

log.Println(a)

cart := GetCart(a)

log.Println(cart)

err = sess.Set("cart", a)

RenderCart(c, cart)

log.Println(err)

}

}

func Cart(c \*ace.C) {

w := c.Writer

r := c.Request

sess, err := globalSessions.SessionStart(w, r)

defer sess.SessionRelease(w)

user := sess.Get("email")

cart := sess.Get("cart")

itemsincart := GetCart(cart)

log.Println(itemsincart)

max := len(itemsincart)

if max > 5 {

if cartstr, ok := cart.([]string); ok {

newcart := cartstr[0:4]

sess.Delete("cart")

sess.Set("cart", newcart)

}

}

if user != nil && err == nil {

RenderCart(c, itemsincart)

} else {

badstr := "Sorry, you must be signed in to access this page"

BadPage(c, badstr)

}

}

func GetCart(cart interface{}) []Item {

var cartitems []Item

if cartstring, ok := cart.([]string); ok {

for \_, elem := range cartstring {

cartitem := Item{}

err := db.Get(&cartitem, "SELECT \* FROM items WHERE id=$1", elem)

if err != nil {

log.Print("This must be the problem")

}

cartitems = append(cartitems, cartitem)

}

}

return cartitems

}

func CartTotal(cart []Item) float64 {

total := 0.0

for \_, elem := range cart {

price := strings.Trim(elem.Price, "$")

if n, err := strconv.ParseFloat(price, 64); err == nil {

total += n

}

}

return total

}

func RenderCart(c \*ace.C, itemsincart []Item) {

max := len(itemsincart)

render := render.New(render.Options{})

if max == 5 {

total := CartTotal(itemsincart)

render.HTML(c.Writer, http.StatusOK, "cart", map[string]interface{}{

"total": total,

"item0": itemsincart[0],

"item1": itemsincart[1],

"item2": itemsincart[2],

"item3": itemsincart[3],

"item4": itemsincart[4],

})

} else if max == 4 {

total := CartTotal(itemsincart)

render.HTML(c.Writer, http.StatusOK, "cart", map[string]interface{}{

"total": total,

"item0": itemsincart[0],

"item1": itemsincart[1],

"item2": itemsincart[2],

"item3": itemsincart[3],

})

} else if max == 3 {

total := CartTotal(itemsincart)

render.HTML(c.Writer, http.StatusOK, "cart", map[string]interface{}{

"total": total,

"item0": itemsincart[0],

"item1": itemsincart[1],

"item2": itemsincart[2],

})

} else if max == 2 {

total := CartTotal(itemsincart)

render.HTML(c.Writer, http.StatusOK, "cart", map[string]interface{}{

"total": total,

"item0": itemsincart[0],

"item1": itemsincart[1],

})

} else if max == 1 {

total := CartTotal(itemsincart)

render.HTML(c.Writer, http.StatusOK, "cart", map[string]interface{}{

"total": total,

"item0": itemsincart[0],

})

} else if max == 0 {

SimplePage(c.Writer, c.Request, "cart")

} else if max > 5 {

total := CartTotal(itemsincart)

maxitems := true

render.HTML(c.Writer, http.StatusOK, "cart", map[string]interface{}{

"total": total,

"max": maxitems,

"item0": itemsincart[0],

"item1": itemsincart[1],

"item2": itemsincart[2],

"item3": itemsincart[3],

"item4": itemsincart[4],

})

}

}

type intSlice []int

func Stringtoint(strs []string) intSlice {

var ints intSlice

for \_, elem := range strs {

var i int

if \_, err := fmt.Sscanf(elem, "%5d", &i); err == nil {

ints = append(ints, i)

}

}

return ints

}

func (slice intSlice) pos(value int) int {

for p, v := range slice {

if v == value {

return p

}

}

return -1

}

func Checkout(c \*ace.C) {

r := c.Request

w := c.Writer

stripe.SetKey("sk\_test\_s7zyOcXwo4E9YLlBXpL4Ie44")

r.ParseMultipartForm(5)

sess, err := globalSessions.SessionStart(w, r)

defer sess.SessionRelease(w)

user := sess.Get("email")

cartcookie := sess.Get("cart")

badstr := "There was an error with your purchase."

cart := GetCart(cartcookie)

total := CartTotal(cart)

totalcent := total \* 100

log.Println(totalcent)

totalcentint64 := int64(totalcent)

name := r.PostFormValue("nameoncard")

cardnum := r.PostFormValue("cardnumber")

expmonth := r.PostFormValue("expirationmonth")

expyear := r.PostFormValue("expirationyear")

if user != nil && err == nil {

err = sess.Delete("cart")

expmonthnum, \_ := strconv.ParseInt(expmonth, 0, 64)

expyearnum, \_ := strconv.ParseInt(expyear, 0, 64)

expyearnumint := int(expyearnum)

expmonthnumint := int(expmonthnum)

log.Println(totalcentint64)

params := stripe.ChargeParams{

Desc: "Cart",

Amount: totalcentint64,

Currency: "usd",

Card: &stripe.CardParams{

Name: name,

Number: cardnum,

ExpYear: expyearnumint,

ExpMonth: expmonthnumint,

},

}

log.Println(params)

charge, err := stripe.Charges.Create(&params)

log.Println(charge)

log.Println(err)

if err == nil {

render := render.New(render.Options{})

render.HTML(c.Writer, http.StatusOK, "checkout", nil)

}

} else {

BadPage(c, badstr)

}

}

func Buy(c \*ace.C) {

var w = c.Writer

var r = c.Request

sess, err := globalSessions.SessionStart(w, r)

defer sess.SessionRelease(w)

user := sess.Get("email")

cart := sess.Get("cart")

itemsincart := GetCart(cart)

total := CartTotal(itemsincart)

badstr := "you must be signed in to access this page"

if user != nil && err == nil {

render := render.New(render.Options{})

render.HTML(c.Writer, http.StatusOK, "buy", total)

} else {

BadPage(c, badstr)

}

}

Приложение 1. Листинг 6. actions.go

package main

import (

"database/sql"

"fmt"

"log"

"net/http"

\_ "github.com/lib/pq"

"github.com/plimble/ace"

//"github.com/astaxie/beego/session"

"github.com/unrolled/render"

"golang.org/x/crypto/bcrypt"

)

func PostLogin(c \*ace.C) {

var w = c.Writer

var req = c.Request

sess, \_ := globalSessions.SessionStart(w, req)

defer sess.SessionRelease(w)

sessname := sess.Get("email")

cartsess := sess.Get("cart")

var cart []string

fmt.Println(sessname)

fmt.Println(cartsess)

var password\_in\_database string

var email string

badstr := "Authorization Failed"

username, password := req.FormValue("inputUsername"), req.FormValue("inputPassword")

err := db.QueryRow("SELECT email, password FROM users WHERE username=$1", username).Scan(&email, &password\_in\_database)

if err == sql.ErrNoRows {

BadPage(c, badstr)

} else if err != nil {

log.Print(err)

BadPage(c, badstr)

}

err = bcrypt.CompareHashAndPassword([]byte(password\_in\_database), []byte(password))

if err == bcrypt.ErrMismatchedHashAndPassword {

BadPage(c, badstr)

} else if err != nil {

log.Print(err)

BadPage(c, badstr)

}

if err == nil {

err2 := sess.Delete("email")

err2 = sess.Set("email", email)

err2 = sess.Set("cart", cart)

log.Println(req)

if err2 == nil {

c.Redirect("/")

}

}

}

func PostSignup(c \*ace.C) {

var w = c.Writer

var req = c.Request

username := req.FormValue("reg\_username")

password := req.FormValue("reg\_password")

email := req.FormValue("reg\_email")

hashedPassword, err := bcrypt.GenerateFromPassword([]byte(password), 10)

if err != nil {

panic(err)

}

\_, err = db.Exec("INSERT INTO users (username, password, email) VALUES ($1, $2, $3)", username, string(hashedPassword), email)

if err != nil {

log.Print(err)

}

http.Redirect(w, req, "/login", 302)

}

func Logout(c \*ace.C) {

var w = c.Writer

var req = c.Request

sess, \_ := globalSessions.SessionStart(w, req)

globalSessions.SessionDestroy(w, req)

err := sess.Delete("email")

if err == nil {

http.Redirect(w, req, "/", 302)

}

}

func ShowItemsHome(w http.ResponseWriter, r \*http.Request) {

items := []Item{}

err := db.Select(&items, "SELECT \* FROM items ORDER BY id ASC")

if err != nil {

log.Println("Is this the problem?")

log.Println(err)

return

}

log.Println("%#v\n", items)

render := render.New(render.Options{

IndentJSON: true,

})

render.HTML(w, http.StatusOK, "home", map[string]interface{}{

"item0": items[0],

"item1": items[1],

"item2": items[2],

})

}

func ShowItemsPages(c \*ace.C, i int) {

var w = c.Writer

items := []Item{}

err := db.Select(&items, "SELECT \* FROM items ORDER BY id ASC")

if err != nil {

log.Println("Is this the problem?")

log.Println(err)

return

}

max := len(items)

factor := i - 1

one, two, three := factor\*3, (factor\*3)+1, (factor\*3)+2

log.Println("%d\n %d\n %d\n", one, two, three)

log.Println(max)

render := render.New(render.Options{

IndentJSON: true,

})

if one < max {

if max%3 == 0 {

render.HTML(w, http.StatusOK, "home", map[string]interface{}{

"item0": items[one],

"item1": items[two],

"item2": items[three],

})

} else if max%3 == 2 {

render.HTML(w, http.StatusOK, "home", map[string]interface{}{

"item0": items[one],

"item1": items[two],

})

} else if max%3 == 1 {

render.HTML(w, http.StatusOK, "home", map[string]interface{}{

"item0": items[one],

})

}

} else {

c.Redirect("/")

}

}

func ShowItemid(c \*ace.C, s string) {

var w = c.Writer

var r = c.Request

fmt.Println(s)

itemid := Item{}

err := db.Get(&itemid, "SELECT \* FROM items WHERE id=$1", s)

if err != nil {

log.Print("This must be the problem")

}

if itemid.Id == 0 {

http.Redirect(w, r, "/", 302)

}

render := render.New(render.Options{

IndentJSON: true,

})

render.HTML(c.Writer, http.StatusOK, "item", &itemid)

}