

Московский государственный технический университет  
имени Н.Э. Баумана

---

Факультет «Информатика и системы управления»

Кафедра «Программное обеспечение ЭВМ и  
информационные технологии»

**Т.И. Вишневская, Т.Н. Романова**

# **МЕТОДОЛОГИЯ ПРОГРАММНОЙ ИНЖЕНЕРИИ**

Электронное учебное издание

*Методические указания к лабораторному практикуму*

*по дисциплине*

*«Методология программной инженерии»*

Москва

(С) 2017 МГТУ им. Н.Э. БАУМАНА

**УДК 004.41**

*Рецензент доцент, к.т.н. Олег Викторович Rogozin*

**Вишневская Т.И., Романова Т.Н.**

Методология программной инженерии: Методические указания к лабораторному практикуму. - М.: Изд-во МГТУ им. Н.Э. Баумана, 2017. –52с.

Даны рекомендации по выбору и использованию методологий и технологий программной инженерии для проектирования Web-портала в рамках выполнения лабораторного практикума по курсу «Методология программной инженерии»

Для студентов МГТУ имени Н.Э. Баумана, обучающихся по направлениям 231000 «Программная инженерия».

*Рекомендовано учебно-методической комиссией факультета «Информатика и системы управления» МГТУ им. Н.Э. Баумана*

Татьяна Ивановна Вишневская

Татьяна Николаевна Романова

**МЕТОДОЛОГИЯ ПРОГРАММНОЙ ИНЖЕНЕРИИ**

(С) 2017 МГТУ им. Н.Э. БАУМАНА

# ОГЛАВЛЕНИЕ

ОГЛАВЛЕНИЕ .....	3
ПРЕДИСЛОВИЕ .....	4
1. WEB-ПОРТАЛ КАК РАСПРЕДЕЛЕННАЯ СИСТЕМА.....	5
2. ТЕХНИЧЕСКОЕ ЗАДАНИЕ.....	6
Глоссарий.....	6
Введение .....	7
Краткое описание предметной области.....	7
Существующие аналоги.....	7
Описание системы .....	7
Основания для разработки .....	8
Назначение разработки.....	8
Требования к системе .....	8
Требования к функциональным характеристикам.....	9
Функциональные требования к portalу с точки зрения пользователя .....	9
Входные параметры системы .....	10
Выходные параметры системы.....	10
Требования к составу и параметрам технических средств .....	11
Требования к надежности .....	11
Требования к документации .....	11
Лабораторная работа 1.....	11
Вопросы для самоконтроля .....	11
3. СОЗДАНИЕ ОБЩЕЙ КАРТИНЫ ПОРТАЛА.....	12
Роль географического местоположения пользователей.....	12
Топология системы.....	13
Требования по реализации со стороны заказчика .....	14
Функциональные требования по подсистемам.....	15
Лабораторная работа 2.....	17
Вопросы для самоконтроля .....	17
Сценарий взаимодействия фронтенда и бекендов.....	17
Сценарий взаимодействия бекендов с СУБД.....	18
Лабораторная работа 3.....	20
Вопросы для самоконтроля .....	20
Пользовательский интерфейс .....	20
Лабораторная работа 4.....	22
Вопросы для самоконтроля .....	22
4. ПРОЕКТИРОВАНИЕ ПОРТАЛА.....	23
Концептуальный дизайн.....	23
Концептуальная модель системы в нотации IDEF0.....	23
Сценарии функционирования системы.....	24
Диаграммы прецедентов .....	25
Лабораторная работа 5.....	28
Вопросы для самоконтроля .....	28
Логический дизайн .....	29
Диаграмма классов .....	29
Диаграмма деятельности.....	33
Диаграммы последовательности действий .....	34
Диаграмма потоков данных .....	35
Лабораторная работа 6.....	35
Вопросы для самоконтроля .....	35

Высокоуровневый дизайн пользовательского интерфейса .....	36
<b>Лабораторная работа 7.....</b>	<b>40</b>
<b>Вопросы для самоконтроля .....</b>	<b>40</b>
Выбор системы развертывания компонентов распределенной системы.....	41
Выбор операционной системы .....	44
Выбор СУБД для профилей пользователей .....	44
Выбор СУБД для учета финансовой информации.....	45
Выбор языка разработки компонент портала .....	46
Выбор фреймворка для разработки портала .....	47
Обеспечение надежности портала.....	48
<b>Лабораторная работа 8.....</b>	<b>49</b>
<b>Вопросы для самоконтроля .....</b>	<b>49</b>
<b>ЗАКЛЮЧЕНИЕ .....</b>	<b>50</b>
<b>СПИСОК ЛИТЕРАТУРЫ .....</b>	<b>51</b>

## ПРЕДИСЛОВИЕ

Интернет – это колоссальное хранилище информации и средство поиска в нем необходимых данных. Пользователи сети осуществляют свои поисковые запросы с использованием различных интерактивных сервисов (интернет-сервисов), которые работают в рамках Web-сайтов. Веб-портал для пользователей — сайт в компьютерной сети, который предоставляет пользователю сервисы, работающие в рамках этого сайта. Веб-портал может состоять из нескольких сайтов, если они объединены под одним доменным именем.

В методических указаниях «Методология программной инженерии» даны рекомендации по выбору и использованию методологий и технологий программной инженерии для проектирования Web-портала при выполнении лабораторного практикума по курсу «Методология программной инженерии».

Практикум предназначен для студентов специальности «Программная инженерия», изучающих информационные технологии в рамках дисциплины «Методология программной инженерии».

**Авторы выражают благодарность студентам факультета «Информатика и системы управления» МГТУ им. Н.Э. Баумана Исаеву Дмитрию Сергеевичу и Ремень Ивану Валерьевичу за помощь в разработке программного проекта по предложенной методике.**

## 1. WEB-ПОРТАЛ КАК РАСПРЕДЕЛЕННАЯ СИСТЕМА

Web-портал это Web-сайт, представляющий набор сервисов по одной или нескольким тематикам [1]. Как правило, порталы используют сервис-ориентированную архитектуру (COA, или SOA) – подход к разработке, при котором необходимо разбивать систему на распределенные, независимые подсистемы (сервисы). Такой подход удобен, потому что облегчает командную разработку, позволяет использовать различные технологии и раздельно масштабировать сервисы. Следовательно, информационный Web-портал - это распределенная информационная система, основанная на современных сетевых коммуникационных технологиях. Примерами Web-порталов могут служить универсальные порталы, предоставляющие сервисы общего назначения, порталы тематической направленности, корпоративные порталы для сотрудников одного предприятия.

Цель лабораторного практикума – освоение методологий программной инженерии для разработки Web-портала. Для достижения данной цели необходимо:

- выбрать тематику портала;
- разработать общую картину Web-портала;
- выполнить проектирование портала.

В данной работе предложены рекомендации по выбору и использованию методологий и технологий программной инженерии для разработки Web-сайта «Портал для мастеров и клиентов в сфере бытовых услуг». Данный сайт должен включать в себя сервисы поиска клиентов для мастеров, сервисы поиска мастеров для клиентов, новостные сервисы и реализовывать распределенную систему [4], так как местоположение подсистем играет существенную роль с точки зрения функционирования сайта. Поэтому разрабатываемый веб-сайт именуется в данной работе порталом и является примером тематического портала. При выборе этапов описания инженерного цикла разработки Web-портала использовался подход, описанный в [3] и основанный на платформе Microsoft .NET.

## 2. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

### Глоссарий

Термин	Определение
Валидация данных	Проверка на корректность, полноту и непротиворечивость входных, выходных и обрабатываемых данных
WEB-интерфейс	Интерфейс пользователя, предоставляемой системой через Web-браузер. В разрабатываемой системе только один веб-интерфейс.
Медиана	Возможное значение признака, которое делит ранжированную совокупность (вариационный ряд выборки) на две равные части: 50 % «нижних» единиц ряда данных будут иметь значение признака не больше, чем медиана, а «верхние» 50 % — значения признака не меньше, чем медиана.
COA (SOA)	Сервис-ориентированная архитектура (Service Oriented Architecture), модульный подход к разработке программного обеспечения, основанный на использовании распределённых, слабо связанных заменяемых компонентов, оснащённых стандартизированными интерфейсами для взаимодействия по стандартизированным протоколам.
Профиль пользователя	Информация о пользователе портала, хранящаяся в портале, в частности, имя, фамилия, фотография и др.
Сессия	Сессия на сайте - серия запросов к portalу, сделанных одним пользователем в заданный промежуток времени, в данном документе - в течение 30 минут.
Фронтенд	Серверное приложение, принимающее запросы от пользователя портала. На каждый из типов запросов от пользователя (показать новости, оплатить работу мастера и др.) фронтенд определяет, как организовать выполнение запроса. Фронтенд принимает запросы от пользователя, анализирует их и в соответствии с заложенным алгоритмом выполняет запросы к бекендам.
Бекенд	Серверное приложение, выполняющее определенную задачу, например, взаимодействие с СУБД. Бекенды принимают запросы от фронтенда.
Сервер	Компьютер, выполняющий функции обслуживания пользователей при доступе к информационным ресурсам в вычислительных системах.
Шардинг	Разделение данных по диапазонам первичных ключей между несколькими базами данных.
Проект, портал, система	В данной работе термины «проект», «портал» и «система» взаимозаменяемы.

## ***Введение***

Данное техническое задание является модельным (одним из возможных заданий для лабораторных работ по Методологии программной инженерии) и составлено для разработки проекта «Портал для мастеров и клиентов в сфере бытовых услуг». Техническое задание выполнено на основе ГОСТ 19.201—78 «ЕСПД. Техническое задание. Требования к содержанию и оформлению» [5].

## ***Краткое описание предметной области***

Существует проблема поиска услуги в сфере бытовых услуг, подходящей по качеству и цене для клиента. Клиенты могут пользоваться услугами компаний, либо частных мастеров. Мастера могут осуществлять свою деятельность, как из дома, так и с выездом к клиенту домой.

Сервисы, предлагающие услуги по поиску частных мастеров, являются агрегаторами услуг. Такие сервисы часто выполняются в виде портала со списком частных мастеров и возможностями подбора нужного мастера.

Данное техническое задание определяет требования к разработке веб-портала для подбора мастеров в сфере бытовых услуг.

## ***Существующие аналоги***

Среди аналогов можно отметить порталы [servisMos.ru](http://servisMos.ru) и [MasterSPb.ru](http://MasterSPb.ru). Данный проект должен иметь следующие преимущества перед существующими аналогами:

- поиск мастеров не только в Москве и Санкт-Петербурге, но и в других городах России;
- поиск, как частных мастеров, так и мастеров из салонов;
- высокая скорость загрузки страниц портала.

## ***Описание системы***

Проект должен представлять собой портал для соединения клиентов и мастеров в сфере бытовых услуг. Каждый мастер регистрируется на портале и указывает информацию о себе: имя, фамилия, географическое положение, фотография, описание и виды оказываемых услуг. На основе этой информации клиенты, посещающие портал, производят поиск подходящего мастера. На рисунке 1 отображена схема предметной области.

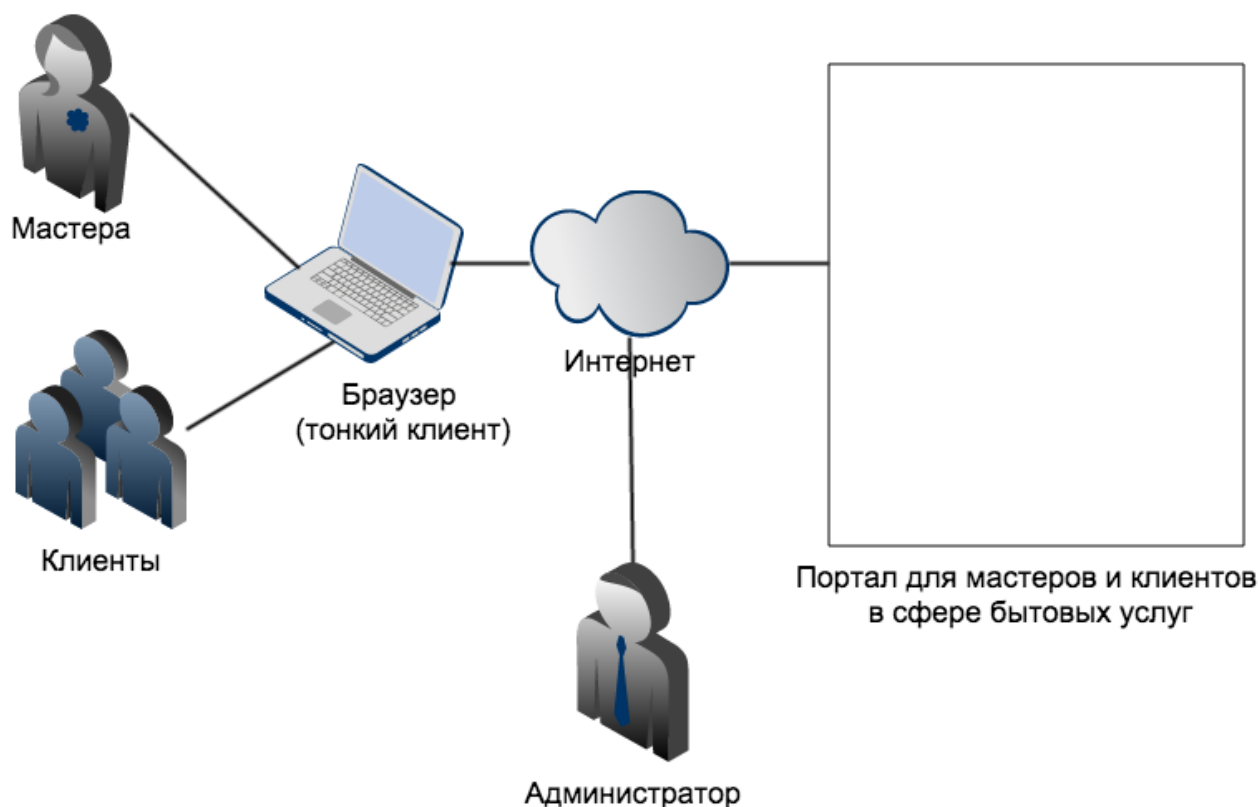


Рис. 1. Схема предметной области.

### ***Основания для разработки***

Разработка ведется в рамках выполнения лабораторных работ по курсу Методология программной инженерии на кафедре «Программное обеспечение ЭВМ и информационные технологии» факультета «Информатика и системы управления» МГТУ им. Н. Э. Баумана.

### ***Назначение разработки***

Главное назначение разрабатываемого портала – соединение клиентов и мастеров. Портал должен учитывать все пожелания к нему со стороны клиента и со стороны мастера. Клиент подбирает себе мастера по интересующей его услуге (парикмахерские услуги, ремонт бытовой техники и т.п.) на основе определенного набора параметров. Мастер публикует на данном портале свои услуги и надеется на привлечение клиентов с целью извлечения прибыли.

### ***Требования к системе***

1. Разрабатываемое программное обеспечение должно обеспечивать функционирование системы в режиме 24/7/365 со среднегодовым временем доступности не менее



99.9%. Допустимое время, в течение которого система не доступна, за год должна составлять  $24 \cdot 365 \cdot 0.001 = 8.76$  часа.

2. Время восстановления системы после сбоя не должно превышать 15 минут.
3. Система должна поддерживать возможность «горячего» переконфигурирования системы. Необходимо поддерживать возможность добавления нового узла во время работы системы без рестарта.

### ***Требования к функциональным характеристикам***

1. По результатам работы модуля сбора статистики медиана времени отклика системы на запросы пользователя на получение информации не должна превышать 3 секунд без учета латентности географического расположения узла [3].
2. По результатам работы модуля сбора статистики медиана времени отклика системы на запросы, добавляющие или изменяющие информацию на портале не должна превышать 7 секунд без учета латентности географического расположения узла.
3. Портал должен обеспечивать возможность запуска в современных браузерах: не менее 85% пользователей Интернета должны иметь возможность пользоваться порталом без какой-либо деградации функционала.

### ***Функциональные требования к portalу с точки зрения пользователя***

Портал должен обеспечивать реализацию следующих функций:

1. Система должна обеспечивать регистрацию пользователей с валидацией вводимых данных.
2. Система должна обеспечивать аутентификацию пользователей.
3. Система должна обеспечивать разделение пользователей на три роли:
  - мастер;
  - клиент
  - администратор.
4. Система должна предоставлять **мастеру** следующие функции:
  - просмотр информации о клиентах, которые интересовались им;
  - управление своим расписанием и сервисом онлайн-бронирования;
  - оплата за клиентов, пришедших к мастеру через портал.
5. Система должна предоставлять **администратору** следующие функции:
  - неограниченные полномочия по изменению контента на портале;
  - возможность конфигурирования узлов системы:  
настройки, удаления, добавления узлов;

- возможность «горячего» конфигурирования узлов (без рестарта).
6. Система должна предоставлять **клиенту** следующие функции:
- просмотр информации о мастерах: имя, фотография, описание профиля мастера, фотографии работ мастера.
  - поиск мастеров по заданным параметрам:
    - географическое положение – задается клиентом на карте;
    - вид оказываемой услуги.

### ***Входные параметры системы***

#### **Мастер**

- Фотографии работ мастера в формате JPEG, размером не более 1МБ и разрешением не более 1000х1000 пикселей.
- Фотография мастера в формате JPEG, размером не более 300кб и разрешением не более 500х500 пикселей.
- Имя и фамилия, максимальная длина каждого из них 512 символов.
- Географическое положение мастера в виде текста на естественном языке, описывающего расположение мастера. Максимальная длина текста – 1024 символа. Например, «г. Киров, ул. Мира, дом 73/2».
- Текстовое описание профиля мастера, в которое мастера будут включать описание своих услуг, их стоимость и т.п. Максимальная длина текста – 8192 символа.

#### **Клиент**

- Фотография клиента в формате JPEG, размером не более 300кб и разрешением не более 500х500 пикселей.
- Имя, максимальная длина 512 символов.

### ***Выходные параметры системы***

Выходными параметрами системы являются веб-страницы. Они должны содержать следующую информацию:

- карту с указанием географического местоположения мастеров по заданному запросу;
- детальную информацию о выбранном на карте мастере (фото, адрес, перечень услуг, стоимость услуг).

### ***Требования к составу и параметрам технических средств***

Все серверные приложения должны потреблять суммарно не более 2 Гб оперативной памяти и работать на сервере с процессором Intel i7 2.4 GHz и жестким диском SATA 100 IOPS 100 Гб.

### ***Требования к надежности***

Система должна работать в соответствии с данным техническим заданием без рестарта. Необходимо использовать «зеркалируемые серверы» для всех подсистем, которые будут держать нагрузку в случае сбоя до тех пор, пока основной сервер не восстановится.

### ***Требования к документации***

Исполнитель должен подготовить и передать Заказчику следующие документы:

- руководство по развертыванию Системы;
- руководство администратора Системы;
- руководство для мастера по использованию Системы;
- руководство для клиента по использованию Системы.

## ***Лабораторная работа 1***

**Цель работы:** выбрать тематику Web-портала и написать техническое задание для его разработки.

**Задачи и порядок выполнения работы:** использовать ГОСТ 19.201—78 «ЕСПД. Техническое задание. Требования к содержанию и оформлению» [5] и методику, изложенную при написании технического задания модельного Web-портала.

**Форма отчета по лабораторной работе** – это текст технического задания для разработки Web-портала выбранной тематики.

### ***Вопросы для самоконтроля***

1. Укажите отличие функциональных характеристик от функциональных требований к portalу.
2. Сформулируйте роль описания входных и выходных данных для разработки portalа.

### 3. СОЗДАНИЕ ОБЩЕЙ КАРТИНЫ ПОРТАЛА

Цель создания документа Общая картины портала – выработать единое понимание проекта всеми его участниками (как разработчиками, так и заказчиками).

#### *Роль географического местоположения пользователей*

Разрабатываемый портал нацелен на жителей всех регионов России, поэтому необходимо учитывать географическое положение пользователей. Если серверы проекта располагаются в Москве, то пользователи из других удаленных населенных пунктов могут испытывать трудности с загрузкой страниц портала. Это будет происходить из-за латентности сети, то есть время отклика увеличивается на латентное время. В среднем, веб-сайт требует от браузера около 100 запросов к серверу для полной загрузки. Если каждый из запросов замедляется на время латентности сети (например, латентность сети от Москвы до Германии составляет около 40 миллисекунд), то загрузка портала значительно замедляется в глазах пользователей.

Для решения данной проблемы следует использовать Content Delivery Network (CDN) – сеть доставки контента. Инфраструктура CDN позволяет уменьшить время загрузки статических ресурсов сайта за счет более близкого географического расположения серверов провайдера CDN к пользователям. Обычно поставщики услуг CDN размещают свои серверы в крупных городах по всей России. Таким образом, пользователь из Хабаровска получит статический контент (который составляет большую часть запросов на сайт со стороны браузера) не от сервера из Москвы, а от сервера из ближайшего города. При разработке Портала для мастеров и клиентов в сфере бытовых услуг можно воспользоваться услугами Российского оператора CDN [6].

## Топология системы

На Рис. 2 изображен один из возможных вариантов топологии разрабатываемой распределенной системы.

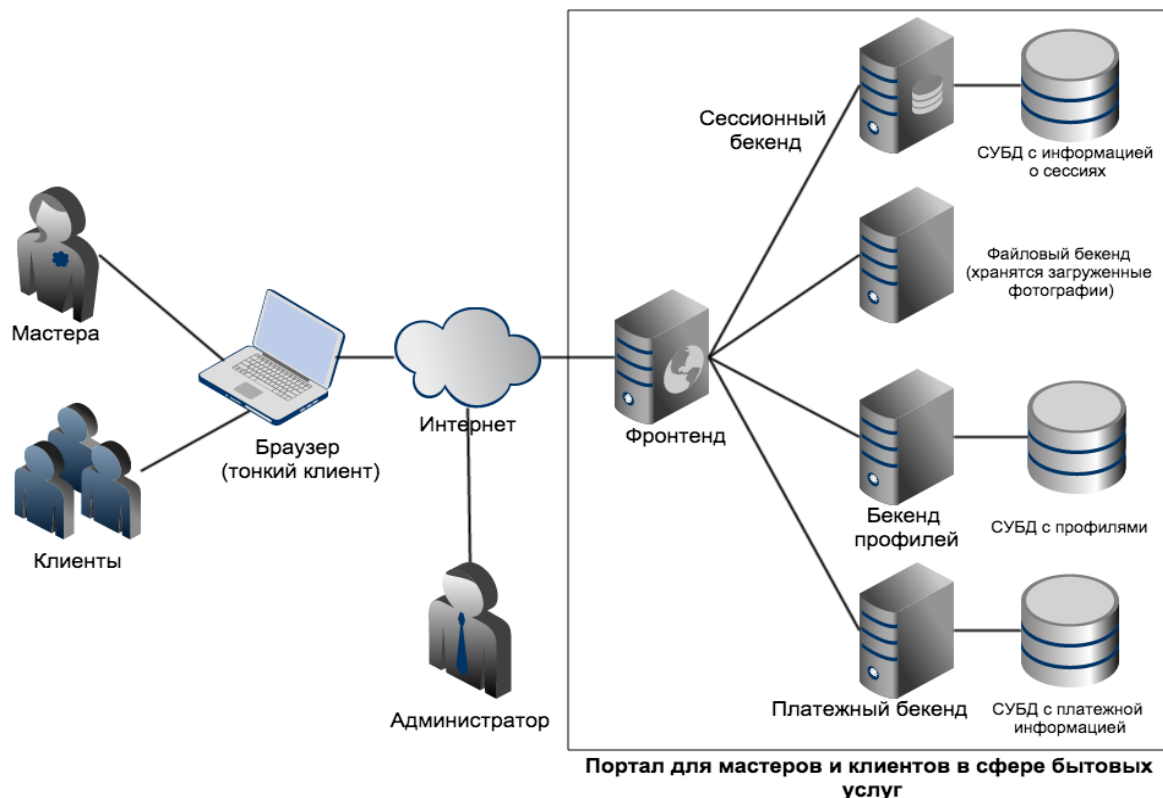


Рис. 2. Топология системы.

Система будет состоять из фронтенда и четырех бекендов, что наиболее целесообразно для реализации ее основного назначения.

- **Сессионный бекенд** отвечает за сессию пользователей портала и реализует следующие функции:
  - регистрация пользователя (клиента или мастера);
  - аутентификация (проверка сессии) пользователя;
  - авторизация пользователя (вход, или «логин»);
  - выход из сессии («логаут»).
- **Файловый бекенд** отвечает за хранение данных в файлах, которые содержат фотографии пользователей и фотографии работ мастеров.
- **Бекенд профилей** реализует следующие функции:
  - получение списка мастеров с условиями фильтрации;
  - получение, изменение, удаление конкретного профиля;
  - добавление нового профиля.

В профиле хранится следующая информация о **клиенте**:

- имя, фамилия;
- пол;
- адрес;
- фотография.

В профиле хранится следующая информация о **мастере**:

- имя, фамилия;
- пол;
- адрес;
- фотография мастера;
- описание предоставляемых услуг и их стоимость;
- фотографии работ данного мастера.

• **Платежный бекенд** реализует функции работы с платежами для мастеров:

- проведение платежа от клиента к portalу;
- получение истории проведенных платежей клиента;
- получение истории проведенных платежей для мастера.

**Фронтенд** принимает запросы от пользователей по протоколу HTTP и анализирует их. На основе проведенного анализа фронтенд выполняет запросы к бекендам, агрегирует ответы бекендов и отправляет ответ пользователю. Фронтенд также включает в себя модуль сбора статистики. Модуль сбора статистики считает и визуализирует время загрузки страницы.

### ***Требования по реализации со стороны заказчика***

1. Все бекенды и фронтенд должны быть запущены изолированно друг от друга.
2. Требуется использовать СОА (сервис-ориентированную архитектуру).
3. Разработка серверных приложений может осуществляться с использованием разных фреймворков для разных бекендов, поскольку бекенды должны быть слабо связаны друг с другом по требованию сервисно-ориентированного подхода к разработке.
4. Необходимо реализовать один веб-интерфейс для фронтенда. Интерфейс должен быть доступен через тонкий клиент – браузер.
5. Необходимо реализовать интерфейс для администратора с использованием командной строки (консоли), позволяющий осуществлять конфигурацию узлов системы, добавлять и удалять узлы.

6. Серверы бекендов недоступны пользователю, это реализуется их расположением во внутренней сети.
7. Валидация входных данных должна производиться и на стороне пользователя с помощью JavaScript скриптов, и на стороне фронтенда. Бекенды не должны валидировать входные данные, так как пользователь не может к ним обращаться напрямую, бекенды должны получать уже отфильтрованные входные данные от фронтенда.

## **Функциональные требования по подсистемам**

1. **Фронтенд**– это серверное приложение при разработке которого необходимо учитывать следующие факторы:
  - фронтенд должен принимать запросы по протоколу HTTP и формировать ответы пользователям портала в формате HTML;
  - в зависимости от типа запроса фронтенд должен отправлять последовательные запросы в соответствующие бекенды;
  - запросы к бекендам необходимо осуществлять по протоколу HTTP. Данные необходимо передавать в формате JSON. Данный текстовый формат обмена данными удобен для чтения;
  - требуется использование «обратного прокси» Nginx последней стабильной версии. Использовать его для возврата пользователям статического содержимого (изображения, таблиц стилей, JavaScript файлов) [7];
  - целесообразно использовать библиотеку Bootstrap v3 для создания HTML-верстки [8].
2. **Сессионный бекенд** – это серверное приложение, которое должно отвечать следующим требованиям по разработке.
  - Сессионный бекенд должен принимать и возвращать данные в формате JSON по протоколу HTTP.
  - Выполнять авторизацию пользователей, проверять и удалять сессию, а также регистрировать пользователей.
  - Поддерживать авторизацию и регистрацию пользователей через социальные сети Вконтакте и Facebook.
  - База данных, содержащая информацию о сессиях, должна находиться на этом же сервере. Доступ к СУБД должен осуществляться по протоколу TCP.
  - Резервирование базы данных должно производиться по расписанию.
  - Необходимо предусмотреть разработку скрипта для автоматического создания копий базы данных.

- Необходимо применить «шардинг» базы данных – разделение данных по диапазонам первичных ключей между несколькими базами данных для обеспечения хранения данных на нескольких серверах [12].

3. **Файловый бекенд** должен быть серверным приложением, которое:

- должно принимать и отвечать на запросы в формате JSON по протоколу HTTP;
- приложение должно уметь обрабатывать запросы на загрузку файлов в бекенд и получение файлов из бекенда.

Для файлового бекенда необходимо разработать функциональные тесты.

4. **Бекенд профилей** должен быть серверным приложением, которое:

- должно принимать и отвечать на запросы в формате JSON по протоколу HTTP;
- обрабатывать запросы на создание, удаление, редактирование и получение профиля пользователя по ключу – идентификатору пользователя;
- обрабатывать запросы на получение списка профилей в соответствии с условиями фильтрации профилей. Профиль представляет собой иерархический набор данных.
- При разработке базы данных, содержащей информацию о профилях, требуется учитывать следующие требования:
  - доступ к СУБД должен осуществляться по протоколу TCP;
  - необходимо разработать скрипт для автоматического создания резервной копий базы данных по расписанию;
  - необходимо применить репликацию базы данных для синхронизации содержимого нескольких копий данных распределенной системы;
  - первичным ключом является идентификатор пользователя;
  - необходимо применить «шардинг» базы данных – разделение данных по диапазонам первичных ключей между несколькими базами данных.

5. **Платежный бекенд** должен быть серверным приложением, которое:

- принимает и отвечает на запросы в формате JSON по протоколу HTTP;
- обрабатывает запросы на оплату пользователем услуг портала, на изменение и получение счета пользователя;
- сохраняет информацию об изменениях в журнале транзакций для ручного восстановления счетов пользователей в случае возникновения непредвиденных ошибок;
- разработка базы данных, содержащей платежную информацию, требует учета следующих требований:
  - доступ к СУБД должен осуществляться по протоколу TCP;



- необходимо разработать скрипт для автоматического создания резервной копий базы данных по расписанию;
- необходимо применяться репликация базы данных;
- необходимо применить «шардинг» базы данных – разделение данных по диапазонам первичных ключей между несколькими базами данных.

## **Лабораторная работа 2**

**Цель работы:** описать топологию системы и основные функциональные требования по подсистемам для Web-портала.

**Задачи и порядок выполнения работы:** использовать методiku, изложенную при написании общей картины модельного Web-портала.

**Форма отчета по лабораторной работе** – это описание в произвольной форме общей картины Web-портала, выбранной тематики.

## **Вопросы для самоконтроля**

1. Обоснуйте необходимость учитывать географическое положение пользователей для разработки топологии Web-портала.
2. (Сформулируйте роль описания функциональных требований по подсистемам для распределенной системы, которой представлен Web-портал.)
2. Почему необходимо описывать в проекте функциональные требования к подсистемам и указывать их взаимосвязи?
3. Что такое «шардинг» базы данных и когда применяется такая фрагментация БД?

## **Сценарий взаимодействия фронтенда и бекендов**

Рассмотрим, как взаимодействуют фронтенд и бекенды на примере выполнения запроса от пользователя на получение списка мастеров, находящихся в Москве.

1. На фронтенд приходит данный запрос пользователя.
2. Фронтенд анализирует заголовки запроса пользователя и, если пользователь был авторизован ранее, получает из них идентификатор пользователя – число. Далее выполняется запрос к сессионному бекенду. При этом фронтенд отправляет запрос к случайно выбранному сессионному бекенду из числа всех сессионных бекендов для проверки авторизации пользователя, отправляя заголовки HTTP запроса на сессионный бекенд. На запрос устанавливает максимальное время ожидания ответа (таймаут). Если запрос не успевает выполниться за данное время, то фронтенд по-

вторно отправляет данный запрос, но уже на другой сессионный бекенд. В случае ошибки и с этим бекендом, фронтенд отправляет запрос на следующий бекенд. Так будет продолжаться до тех пор, пока один сессионных бекендов не ответит успешно на запрос, либо пока не будет перебраны все сессионные бекенды. Если не удалось выполнить успешный запрос ни в один из сессионных бекендов, пользователю возвращается ошибка.

3. Выполняется запрос к бекенду профилей. В этот запрос включается условие географической принадлежности мастеров городу Москва. Фронтенд отправляет запрос к случайно выбранному бекенду профилей по первичному ключу – идентификатору пользователя. На запрос устанавливает максимальное время ожидания ответа (таймаут). Если запрос не успевает выполниться за данное время, то фронтенд повторно отправляет данный запрос, но уже на другой бекенд профилей. В случае ошибки и с этим бекендом, фронтенд отправляет запрос на следующий бекенд. Так будет продолжаться до тех пор, пока один из бекендов не ответит успешно на запрос, либо пока не будут опрошены все бекенды профилей. Если запрос к бекенду профилей не был выполнен успешно, пользователю возвращается ошибка.
4. Если ошибки не произошло, то производится генерация HTML содержимого страницы ответа пользователю с использованием данных, полученных от бекенда профилей и сессионного бекенда.

## **Сценарий взаимодействия бекендов с СУБД**

Рассмотрим, как происходит взаимодействие бекендов с СУБД на примере бекенда профилей и запроса на получение списка профилей в городе Москва.

1. Бекенд профилей получает запрос от фронтенда и преобразовывает его в запрос к СУБД.
2. Бекенд выполняет шардинг идентификатора пользователя, передаваемого в запросе к бекенду. Для этого берется остаток от деления идентификатора на число серверов, на которых располагается база данных под управлением СУБД. Данный остаток определяет номер группы серверов, обслуживающей данного пользователя. В данной группе как минимум два сервера: *ведущий и ведомый* (так как применяется master-slave репликация).
3. Бекенд начинает по очереди выполнять запросы к серверам из выбранной группы, выполняя запросы к серверам, на которых расположены ведомые СУБД в первую очередь, так данный запрос не требует изменения данных. Запросы на изменение данных

могут обрабатываться только *ведущим сервером*. Если хотя бы один сервер с СУБД успешно обработал запрос, то генерируется ответ с использованием полученных данных. Если ни один из серверов не смог успешно выполнить запрос, то пользователю возвращается ошибка.

Популярные СУБД, такие как MongoDB, PostgreSQL, реализуют алгоритмы обработки распределенных транзакций, так как запросы на изменение данных к ним должны выполняться сразу на нескольких серверах. Данные СУБД используют такие методы обработки распределенных транзакций, как **двухфазная блокировка транзакций (2PC или двухфазный коммит)** [4] и другие методы. Для обеспечения надежности СУБД используют журналы транзакций.

Спецификация на **распределенные глобальные транзакции (XA – транзакции)** была разработана комитетом *X/Open Group*. XA-транзакции управляются с помощью координатора XA-транзакций. XA-транзакция считается успешно завершенной, только если успешно завершены все локальные транзакции XA-ресурсов (то есть успешны все локальные коммиты). Если коммит в каком-либо из локальных ресурсов неуспешен, то происходит откат всей глобальной транзакции. Для реализации такого поведения используется **протокол двухфазного подтверждения (протокол двухфазного коммита, Two-phase Commit Protocol, 2PC)**.

**Двухфазная фиксация транзакций (2PC, двухфазный коммит)** позволяет PostgreSQL участвовать в распределенных транзакциях. Двухфазный коммит реализуется в два этапа [19]:

1. **Фаза подготовки (PREPARE TRANSACTION)**. Координатор (обычно, иницирующая база данных) посылает сообщение участникам распределенной транзакции о подготовке к транзакции (prepare message). Это сообщение также содержит уникальный номер транзакции TID. Когда участники получают это сообщение, они проверяют, могут ли они зафиксировать транзакцию, и сообщают об этом координатору. При этом транзакция выполняется, но не фиксируется, и ее состояние записывается на диск.
2. **Фаза фиксации (COMMIT PREPARED)**. После того, как координатор получил все ответы, он решает зафиксировать или прервать начатую транзакцию в соответствии с правилом глобальной фиксации. Если он решает, что все хорошо, то он посылает участникам сообщение о фиксации транзакции, в противном случае все участники получают приказ прервать транзакцию.

В результате, все узлы распределенной системы либо фиксируют транзакцию, либо откатывают ее, независимо от сетевых проблем или отказа какого-либо узла. В СУБД PostgreSQL протокол 2PC реализован так, что транзакции сначала подготавливаются (PREPARE TRANSACTION) и «записываются» на диск и впоследствии они могут быть зафиксированы (COMMIT PREPARED) или отменены (ROLLBACK PREPARED) даже после перезагрузки системы.

### **Лабораторная работа 3**

**Цель работы:** описать основные сценарии функционирования Web-портала.

**Задачи и порядок выполнения работы:** использовать методику, изложенную при написании технического задания модельного Web-портала.

**Форма отчета по лабораторной работе** – это текст в произвольной форме с детальным описанием сценариев функционирования всех подсистем портала.

### **Вопросы для самоконтроля**

1. Каким образом учитывать в сценариях функционирования портала географическое размещение его пользователей?
2. Сформулируйте роль двухфазной блокировки транзакций для функционирования Web-портала.
3. В чем достоинства и недостатки двухфазного коммита?

### **Пользовательский интерфейс**

Для реализации серверного приложения фронтенда используется схема «модель – вид – элемент управления» (Model – View – Controller, MVC). Пользовательский интерфейс представляет собой “вид” из этой схемы. Такое разделение позволяет осуществлять разработку пользовательского интерфейса отдельной командой разработчиков, которые специализируются на этом, так как для реализации интерфейса не требуется умений программировать бекенд или фронтенд: разработка интерфейса обычно осуществляется с помощью языка разметки HTML и языка шаблонов используемого фреймворка.

Пользовательский интерфейс в разрабатываемой системе должен обладать следующими характеристиками:

- **Адаптивность к размеру экрана устройства пользователя** – пользовательский интерфейс «подстраивается» под всевозможные размеры экранов устройств: мобильных телефонов, планшетов, ноутбуков и т.д. В зависимости от размера экрана меняется

размер элементов управления и структура некоторых элементов. Например, на маленьких экранах верхнее меню сайта «сворачивается» в кнопку, по нажатию на которую оно раскрывается на весь экран. Если этого не сделать, то меню будет занимать слишком много места на странице.

- **Кроссбраузерность** – способность интерфейса работать практически в любом браузере любой версии. Это сложная задача, так как браузеры часто используют разные наименования одинаковых элементов управления. Поэтому пользовательскому интерфейсу приходится в зависимости от типа браузера пользователя выбирать нужное имя элемента. Обычно, это некое имя с префиксом в виде названия браузера. Например, для свойства `border-radius` (радиус скругления границы элемента) существует 5 разных именований в зависимости от типа и версии браузера:
  - `–moz-border-radius`
  - `–ms-border-radius`
  - `–o-border-radius`
  - `–webkit-border-radius`
  - `border-radius`
- **«Плоский» дизайн** – это один из новых трендов в мире дизайна, суть которого заключается в отказе от объемных элементов (теней элементов, объемных кнопок и т.д.) и в замене их плоскими аналогами. Существует большое количество фреймворков для создания плоского дизайна.
- **Расширяемость** – возможность легко расширять и модифицировать пользовательский интерфейс. Это свойство обеспечивается за счет модульности исходного кода пользовательского интерфейса (разбиение на файлы, общие блоки, наследование от общего шаблона).
- **Доступность в 3 клика** – любая страница портала доступна максимум в 3 клика с главной страницы портала. Это стандартное правило при проектировании пользовательского интерфейса, оно позволяет сильно упростить доступ пользователей к контенту на портале и приводит к хорошей структуре страниц портала.
- **Интуитивно понятный интерфейс** – все кнопки имеют подписи при наведении на них, многие содержат иконки, облегчающие восприятие пользователем. Используются стандартные элементы управления.
- **Быстрый отклик** – используются специальные техники для ускорения работы страниц портала, например, сжатие изображений, отложенное выполнение скриптов, обфускация скриптов (приведение исходного кода к такому виду, что код становится трудночитаемым, но полностью сохраняет свою функциональность; в данном контек-

сте обфускация осуществляется с целью минимизации размера исходного кода) и объединение файлов.

## ***Лабораторная работа 4***

**Цель работы:** разработать высокоуровневые требования к пользовательскому интерфейсу Web-портала.

**Задачи и порядок выполнения работы:** использовать характеристики пользовательского интерфейса, представленные для модельного Web-портала.

**Форма отчета по лабораторной работе** – это текст со списком высокоуровневых требований для разрабатываемого интерфейса.

## ***Вопросы для самоконтроля***

(Укажите основные характеристики пользовательского интерфейса Web-портала).

1. Назовите способы разработки интерфейса для Web-портала.
2. Каковы основные эргометрические требования к интерфейсу Web-портала?

## 4. ПРОЕКТИРОВАНИЕ ПОРТАЛА

Цель этапа проектирования – создание архитектуры системы и дизайна портала.

### Концептуальный дизайн

Для создания функциональной модели портала, отражающей его основные функции и потоки информации наиболее наглядно использовать нотацию IDEF0 [9]. На рисунке 3 приведена концептуальная модель системы. На рисунках 4,5 представлена декомпозиция функциональной модели системы.

### Концептуальная модель системы в нотации IDEF0



Рис. 3. Концептуальная модель системы в нотации IDEF0.

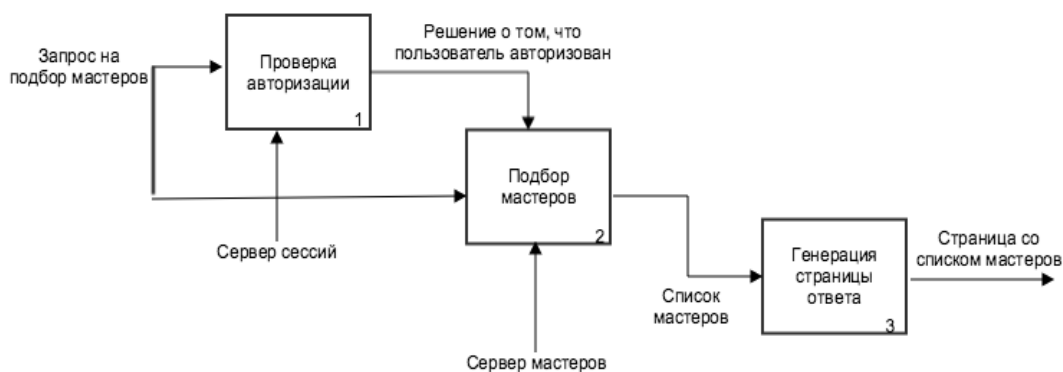


Рис. 4. Детализированная концептуальная модель системы в нотации IDEF0.

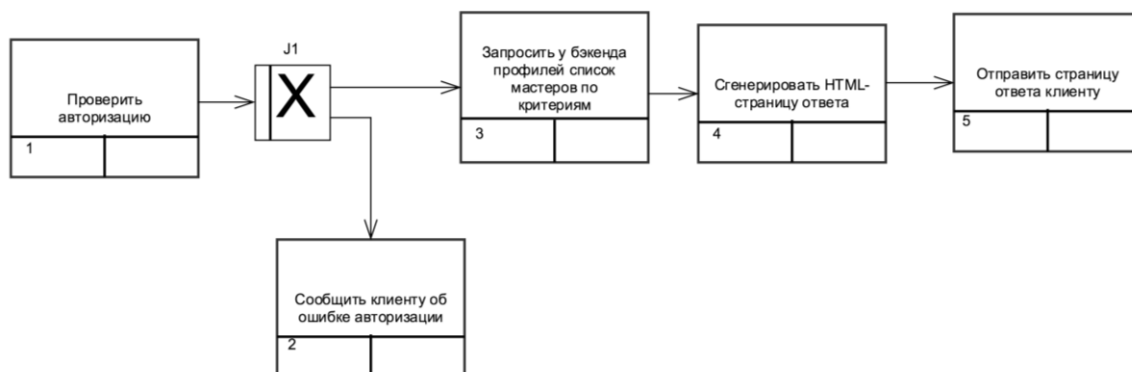


Рис. 5. Схема выбора мастера в нотации IDEF3.

## Сценарии функционирования системы

### Регистрация клиента или мастера

1. Пользователь переходит по ссылке «войти».
2. Пользователь отсылается на сайт «Вконтакте» и подтверждает согласие на использование его данных. Если пользователь не дает согласия, то он перенаправляется на страницу с ошибкой.
3. Пользователь попадает на страницу с выбором типа аккаунта: клиент или мастер.
4. При выборе типа аккаунта пользователь перенаправляется на страницу регистрации, на которой клиент и мастер вводят различные наборы полей. Валидация входных данных осуществляется «на лету» на стороне пользователя. При отправке данных на фронтенд, он тоже производит валидацию.
5. Пользователь нажимает кнопку «Регистрация» и перенаправляется на главную страницу портала.

### Авторизация на портале клиента или мастера

1. Пользователь нажимает на кнопку «войти»
2. Пользователь отсылается на сайт «ВКонтакте» и подтверждает согласие на использование его данных. Если пользователь не дает согласия, то он перенаправляется на страницу с ошибкой.
3. Пользователь перенаправляется на главную страницу портала.



## Выбор клиентом мастера

1. Клиент фильтрует список мастеров, указывая какие его интересуют виды услуг и указывая на карте в каком районе должен находиться мастер.
2. Клиент просматривает информацию об отфильтрованных мастерах и выбирает наиболее подходящего из них.
3. Клиент заходит на страницу профиля интересующего его мастера и нажимает на кнопку «связаться с мастером».
4. Портал регистрирует информацию о том, что данный клиент заинтересовался данным мастером.
5. Клиент получает контактную информацию, которую указал мастер.

Для детальной разработки портала используется унифицированный язык моделирования UML [2]. В системе выделены три роли: мастер, клиент и администратор. На рисунках 6,7,8 представлены диаграммы прецедентов для выделенных ролей и описаны сценарии функционирования наиболее значимых прецедентов.

## Диаграммы прецедентов

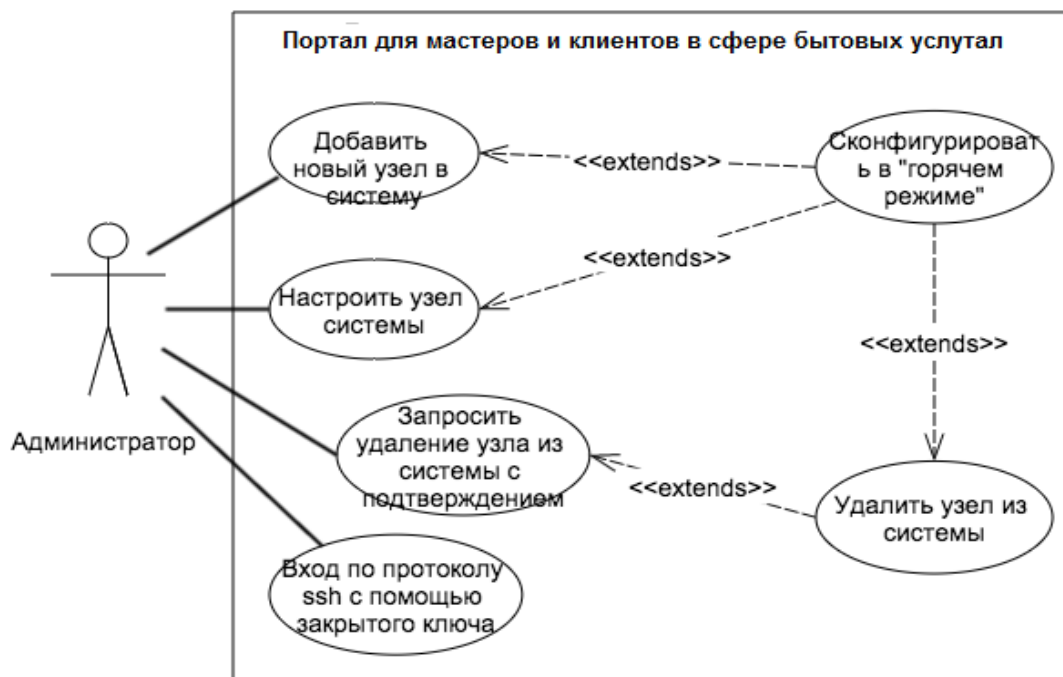


Рис. 6. Диаграмма прецедентов с точки зрения администратора.

## Спецификация сценария удаления узла из системы

### Нормальный ход сценария.

Действия актера	Отклик системы
Вход по протоколу ssh с помощью закрытого ключа	Если закрытый ключ верен, администратору предоставляется консоль.
Запрос удаления узла из системы	Система спрашивает актера, точно ли он хочет удалить узел из системы.
Актер подтверждает удаление узла из системы	Система спрашивает актера, хочет ли он произвести удаление узла в «горячем» режиме
Актер сообщает, что собирается удалить узел в «горячем» режиме	Система удаляет выбранный узел без рестарта системы.

## Спецификация сценария удаления узла из системы.

### Альтернативный ход сценария.

Действия актера	Отклик системы
Запрос удаления узла из системы	Система спрашивает актера, точно ли он хочет удалить узел из системы.
Актер не подтверждает удаление узла из системы	Система не производит удаление узла.

## Спецификация сценария удаления узла из системы.

### Альтернативный ход сценария.

Действия актера	Отклик системы
Вход по протоколу ssh с помощью закрытого ключа	Если закрытый ключ верен, администратору предоставляется консоль.
Запрос удаления несуществующего узла из системы	Система сообщает, что выбранный узел не существует.



Рис. 7. Диаграмма прецедентов с точки зрения клиента.

## Спецификация сценария поиска мастера клиентом.

### Нормальный ход сценария.

Действия актера	Отклик системы
Поиск мастера по критериям (возраст, пол, географическое положение, категории работ)	Система показывает актеру список мастеров, подходящих под выбранные актером критерии.
Просмотр профиля мастера	Система показывает актеру портфолио, личную информацию, фотографию мастера.
Запись к мастеру через сайт	Система записывает актера к выбранному мастеру.

### Альтернативный ход сценария.

Действия актера	Отклик системы
Поиск мастера по критериям (возраст, пол, географическое положение, категории работ)	Система показывает актеру список мастеров, подходящих под выбранные актером критерии.
Просмотр профиля мастера	Система показывает актеру портфолио, личную информацию, фотографию мастера.
Актеру не выбирает данного мастера и либо завершает работу с порталом, либо осуществляет поиск мастера дальше	



Рис. 8. Диаграмма прецедентов с точки зрения мастера.

## Лабораторная работа 5

**Цель работы:** разработать концептуальный дизайн Web-портала.

**Задачи и порядок выполнения работы:** использовать диаграммы, представленные для модельного Web-портала и методику их построения, подробно описанную в [2].

**Форма отчета по лабораторной работе** – это IDF0 диаграммы и диаграммы прецедентов для проектирования Web-портала выбранной тематики.

## Вопросы для самоконтроля

1. Укажите основную роль IDF0 диаграмм при проектировании портала.
2. Сформулируйте основные требования при оформлении диаграмм прецедентов.

## Логический дизайн

Иерархия классов для разработки серверного приложения «Бекенд профилей» представлена в виде диаграммы классов на Рис. 9.

### Диаграмма классов

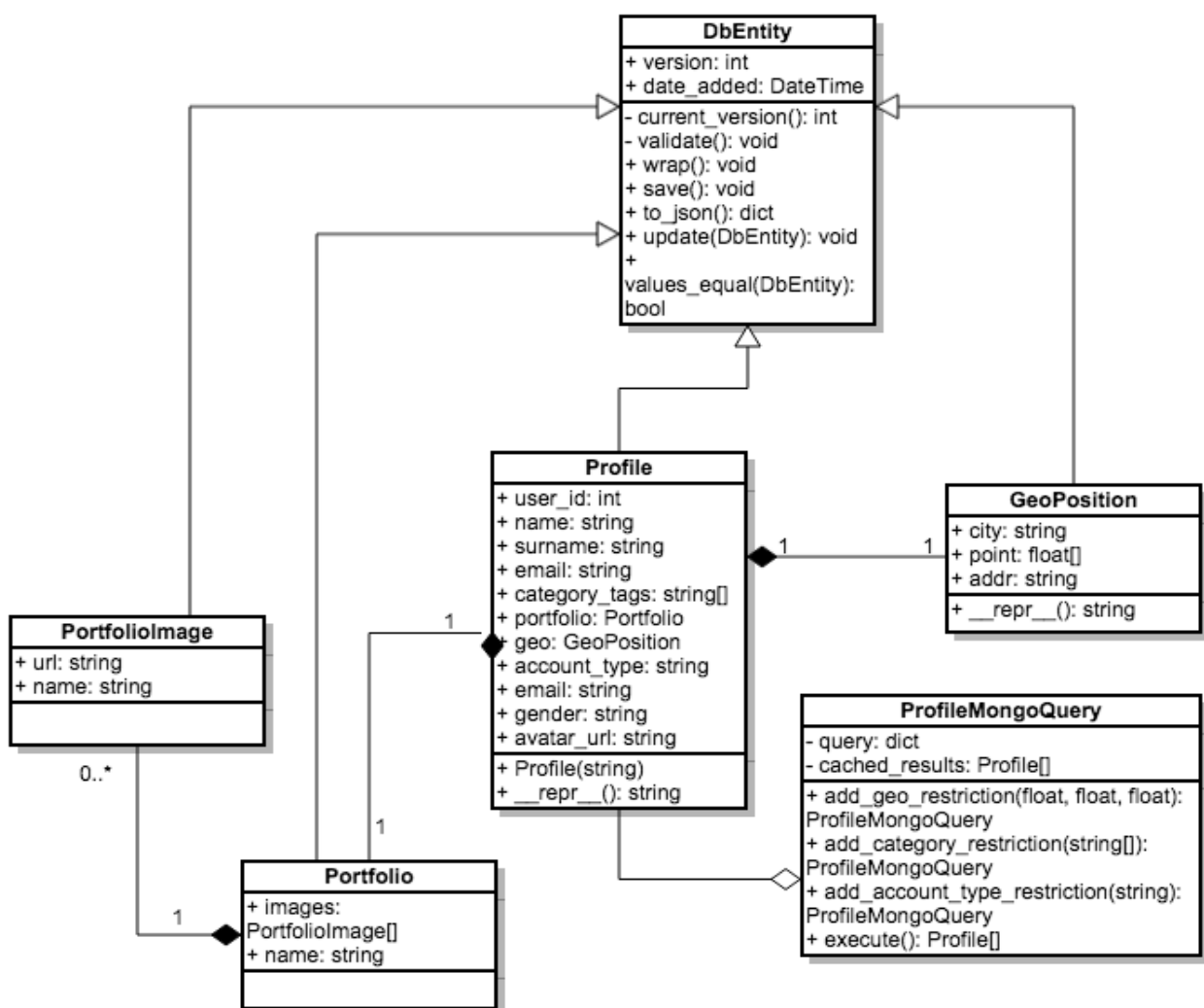


Рис. 9. Диаграмма классов «Бекенда профилей» (уровень реализации).

### Спецификации классов

**Класс DbEntity** представляет собой базовый абстрактный класс для всех сущностей, хранящихся в базе данных профилей.

Методы класса DbEntity:

Название метода	Описание метода
current_version(): int	Возвращает номер версии сущности.
validate()	Валидация объекта.
wrap()	Добавление общих для всех объектов свойств.
save()	Сохранение объекта в базу данных.
to_json(): dict	Представляет объект в JSON формате.
update(DbEntity)	param: entity [ DbEntity - in ] Обновление объекта атрибутами другого объекта.

Атрибуты класса DbEntity:

Имя атрибута	Тип атрибута	Описание атрибута
version	public : <i>int</i>	Номер версии объекта.
date_added	public : <i>DateTime</i>	Дата создания объекта.

**Класс Profile** представляет собой профиль клиента или мастера. Класс используется и для работы с профилями, и для хранения профилей в базе данных с помощью технологии ORM, которая позволяет работать с сущностями базы данных как с обычными объектами.

Методы класса Profile:

Название метода	Описание метода
Profile(string)	param: str [ string - in ] Конструирует объект профиля из строки.
__repr__(): string	Возвращает строковое представление профиля.

Атрибуты класса Profile:

Имя атрибута	Тип атрибута	Описание атрибута
user_id	public : <i>int</i>	Идентификатор профиля
name	public : <i>string</i>	Имя пользователя
surname	public: string	Фамилия пользователя
email	public: string	Email пользователя
category_tags	public: string[]	Список категорий, в которых мастер осуществляет деятельность
portfolio	public: Portfolio	Портфолио мастера
geo	public: GeoPostion	Геопозиция пользователя
account_type	public: string	Тип аккаунта (клиент или мастер)

gender	public: string	Пол пользователя
avatar_url	public: string	URL адрес фотографии пользователя

**Класс GeoPosition** представляет собой географическую позицию пользователя.

Методы класса GeoPosition:

Название метода	Описание метода
__repr__(): string	Возвращает строковое представление позиции.

Атрибуты класса GeoPosition:

Имя атрибута	Тип атрибута	Описание атрибута
city	public: string	Город пользователя
point	public: float[]	Широта и долгота GPS
addr	public: string	Адрес пользователя

**Класс ProfileMongoQuery** представляет собой класс для осуществления запросов к СУБД MongoDB для получения списка профилей.

Методы класса ProfileMongoQuery:

Название метода	Описание метода
add_geo_restriction(float, float, float): ProfileMongoQuery	param: longitude [ float - in ] param: latitude [ float - in ] param: max_distance [ float - in ] Добавление ограничения на географическое положение профиля.
add_category_restriction(string[]): ProfileMongoQuery	param: categories [string[] – in] Добавление ограничения на категорию деятельности мастера.
add_account_type_restriction(string): ProfileMongoQuery	param: account_type [string – in] Добавление ограничения на тип аккаунта.

Атрибуты класса ProfileMongoQuery:

Имя атрибута	Тип атрибута	Описание атрибута
query	public: dict	Параметры запроса
cached_results	private: Profile[]	Результаты выполнения запроса

**Класс Portfolio** представляет собой портфолио мастера.

Атрибуты класса Portfolio:

Имя атрибута	Тип атрибута	Описание атрибута
images	public: PortfolioImage[]	Изображения работ из портфолио.
name	public: string	Краткое описание портфолио.

**Класс PortfolioImage** представляет собой изображение работы из портфолио мастера.

Атрибуты класса PortfolioImage:

Имя атрибута	Тип атрибута	Описание атрибута
url	public: string	URL адрес изображения.
name	public: string	Краткое описание изображения.



## Диаграмма деятельности

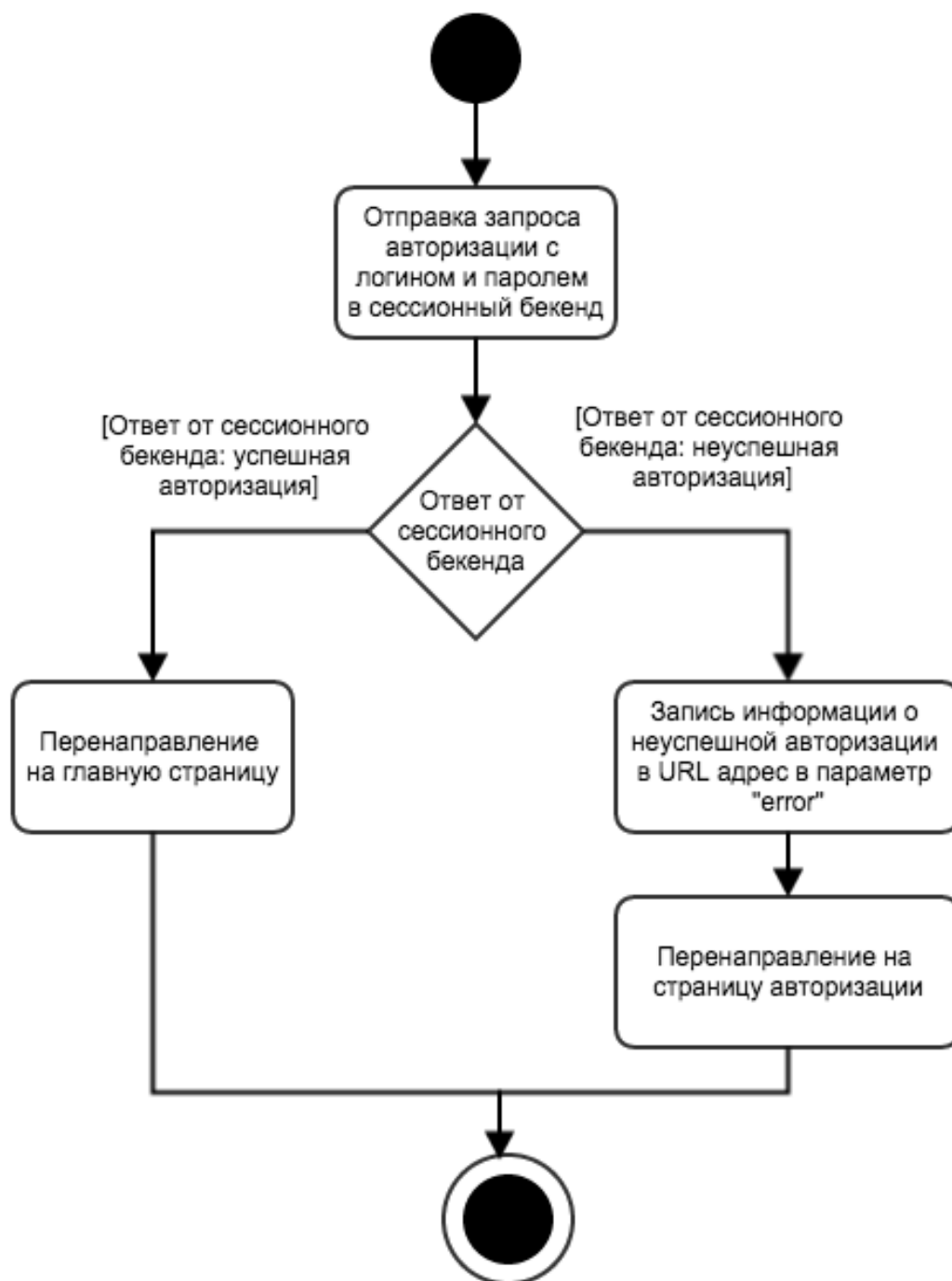


Рис. 10. Диаграмма деятельности для варианта использования «Авторизация» для мастера и пользователя.

## Диаграммы последовательности действий

(концептуальный уровень)

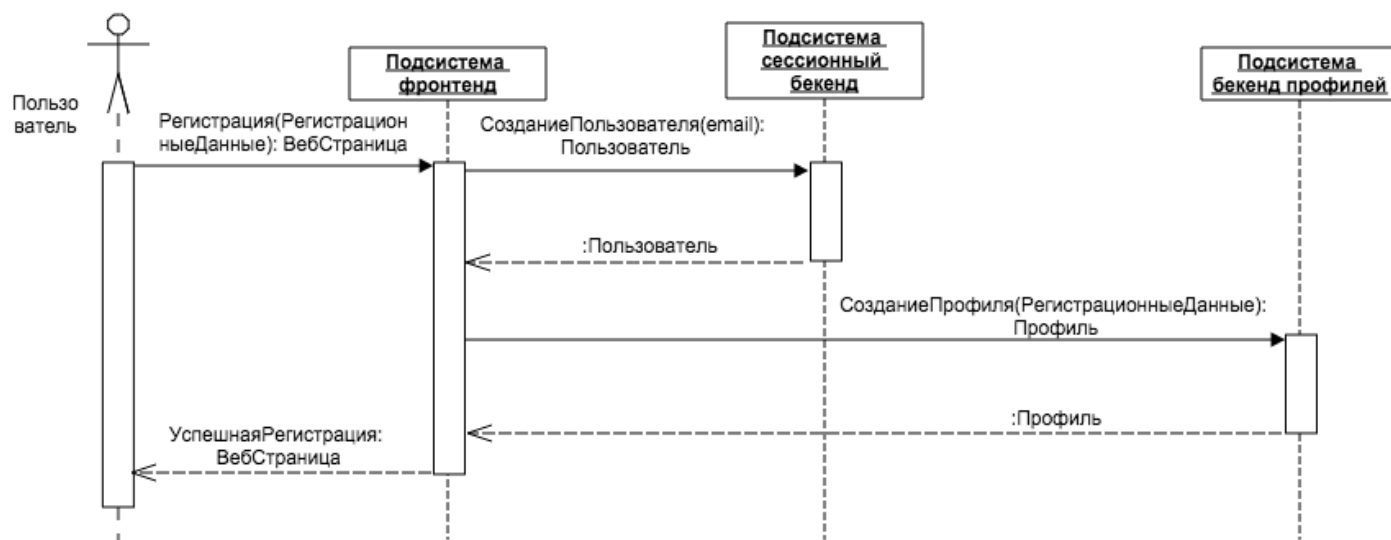


Рис. 11. Диаграмма последовательности действий при регистрации клиента

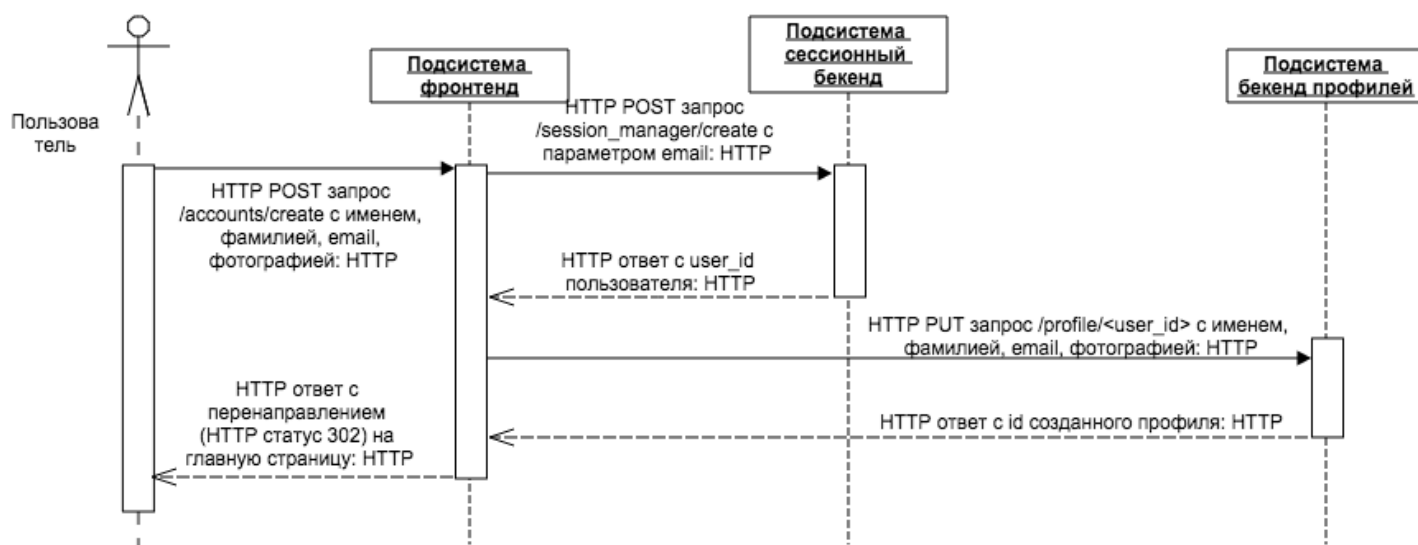


Рис. 12. Диаграмма последовательности для регистрации клиента: детальный уровень.

## Диаграмма потоков данных



Рис. 13. Диаграмма потоков данных системы.

## Лабораторная работа 6

**Цель работы:** разработать логический дизайн Web-портала.

**Задачи и порядок выполнения работы:** использовать диаграммы, представленные для модельного Web-портала и методику их построения, подробно описанную в [2].

**Форма отчета по лабораторной работе** – это диаграммы классов и диаграммы последовательностей действий и диаграммы потоков данных и деятельности Web-портала выбранной тематики.

## Вопросы для самоконтроля

1. Укажите роль диаграмм классов при проектировании портала.
2. Сформулируйте основные назначения диаграмм последовательности действий и диаграмм деятельности.

## Высокоуровневый дизайн пользовательского интерфейса

Пользовательский интерфейс в разрабатываемой системе представляет собой Web-интерфейс, доступ к которому осуществляется через браузер (тонкий клиент).

Страница портала состоит из «шапки» (верхней части страницы, в которой находится логотип и верхнее меню со ссылками на основные разделы портала), основной части и «футера» (нижней части страницы, в которой обычно размещают ссылки на редко посещаемые, но необходимые, страницы, например, страницы с пользовательским соглашением).

Обобщенно структуру страниц портала можно представить следующим образом:

- Главная страница
- Список мастеров с фильтрами
- Карта мастеров с фильтрами
- Страница с информацией о мастере
  - Оплата услуг мастера
  - Онлайн-запись к мастеру
- О нас
  - Гарантия
  - Способы оплаты
  - Контакты
- Статьи
  - Новости
  - Новости портала
  - Познавательный материал
- Регистрация
  - Клиент
  - Мастер
- Личный кабинет
  - Клиент
  - Мастер

Ниже приведены основные формы портала. В форме регистрации клиента, изображенной на рисунке 14, клиент должен ввести свой e-mail, имя и фамилию, указать пол и согласиться с пользовательским соглашением. Также клиент может выбрать фотографию, по умолчанию фотография выбирается из профиля клиента в социальной сети.

### Завершение регистрации

☐ Я согласен с пользовательским соглашением

### Фотография на сайте




Рис. 14. Форма регистрации клиента.

В форме регистрации мастера, изображенной на рисунке 15, мастер должен ввести свой e-mail, имя, фамилию, пол, выбрать хотя бы одну категорию услуг и ввести свой адрес по карте либо в текстовом поле. Также мастер должен согласиться с пользовательским соглашением. Мастер, как и клиент, может выбрать фотографию (данная часть формы не отображена, она аналогична полю выбора фотографии из рисунка 14).

### Завершение регистрации

Имя

Фамилия

Пол

Выберите категории услуг, которые вы оказываете

Волосы


Выпрямление волос

Лицо

Макияж

Адрес

Найти



Тверь, Дубна, Клин, Солнечногорск, Зеленоград, Королев, Сергиев Посад, Александров, Орехово-Зуево, Люберцы, Подольск, Воскресенск, Коломна, Ступино, Серпухов, Чехов, Наро-Фоминск, Обнинск, Калуга, Алексин, М-3, 50 км, © Яндекс Условия Яндекс

☐ Я согласен с пользовательским соглашением

Рис. 15. Форма регистрации мастера.

На рисунке 16 отображена форма редактирования аккаунта мастера, данную форму мастер видит при входе в «личный кабинет». Форма позволяет изменять значение всех полей, которые мастер вводил при регистрации.

## Редактирование аккаунта

тема@my.com

Татьяна

Анисимова

Женщина

Выберите категории услуг, которые вы оказываете

Волосы

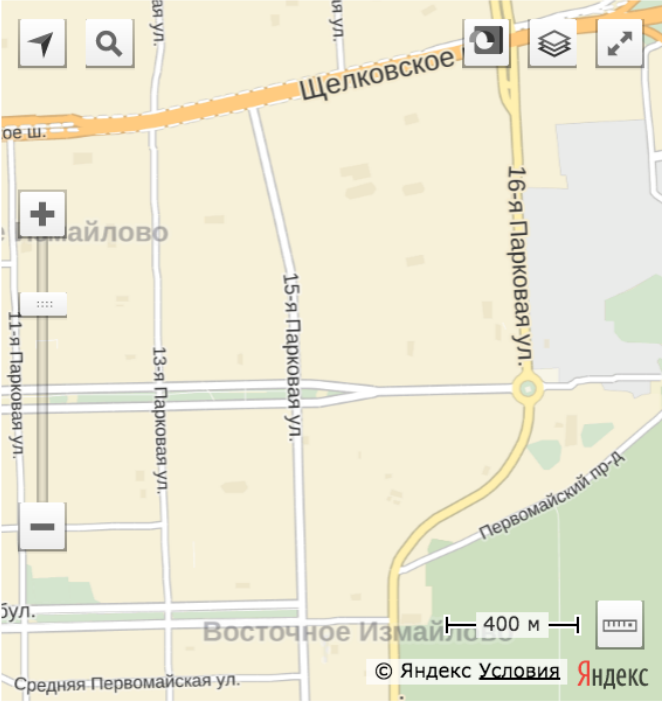
Выпрямление волос

Лицо

Макияж

Адрес

Найти



☒ Я согласен с пользовательским соглашением

Рис. 16. Форма редактирования аккаунта мастера.

## **Лабораторная работа 7**

**Цель работы:** разработать высокоуровневый дизайн пользовательского интерфейса Web-портала.

**Задачи и порядок выполнения работы:** использовать основные этапы, представленные для разработки высокоуровневого дизайна модельного Web-портала.

**Форма отчета по лабораторной работе** – оформить таблицу Структура пользовательского интерфейса и все формы диалогового интерфейса Web-портала. Предусмотреть возможность редактирование интерфейса для пользователей и контроль по входным данным.

### **Вопросы для самоконтроля**

1. Укажите возможные способы контроля по входным данным портала.
2. (Сформулируйте роль описания структуры пользовательского интерфейса для разработки портала.)
2. В каком виде можно описать структуру пользовательского интерфейса?



## **Физический дизайн**

### **Выбор системы развертывания компонентов распределенной системы**

Согласно требованиям технического задания, разрабатываемая система должна быть распределенной. Характерной особенностью распределенных систем является высокое многообразие используемых технологий, языков программирования и библиотек [4]. Особенно непростая ситуация возникает, когда разные приложения используют разные версии одной и той же библиотеки. Для того чтобы фронтенды и бекенды распределенной системы не конфликтовали друг с другом, необходимо ввести требование изолированности. В этом случае приходится использовать отдельные серверы. На начальном этапе проекта это может быть экономически нецелесообразно. Тестирование такой системы также вызовет сложности. Альтернативой является использование виртуальных машин. Виртуализация - это современным подходом к созданию надежной и эффективной серверной инфраструктуры. Она дает возможность, с одной стороны, динамического распределения существующих аппаратных ресурсов между приложениями в соответствии с их текущими потребностями в производительности, оперативной и дисковой памяти, а с другой стороны – дает возможность оперативной миграции приложений между физическими серверами, что существенно сокращает время восстановления при сбоях оборудования.

В качестве программного средства для построения платформы виртуализации на физические серверы часто используются продукты американских компаний VMware и Microsoft. В таблице 1 приведена сравнительная характеристика двух лидеров в области построения платформ для серверной виртуализации. Сравнение программных средств проведено по ряду важных параметров, в том числе, и свойств гипервизора.

**Гипервизор** – это программа, которая обеспечивает разделение и управление ресурсами операционных систем, установленных на одном компьютере. Она и выполняет виртуализацию реального аппаратного обеспечения. Из таблицы 1 следует, что VMware vSphere превосходит Microsoft Hyper-V. Кроме того, VMware vSphere имеет значительно более совершенную, чем в Hyper-V систему обеспечения безопасности в рамках гипервизора, существенное превосходство при работе с системами хранения и сетью, преимущества в области обеспечения высокой надежности. К сожалению, рассмотренные программные продукты имеют существенный недостаток - высокую стоимость.

**Таблица 1. Сравнительная характеристика платформ виртуализации**

<b>Функционал</b>	<b>Реализация в VMware vSphere (ESX)</b>	<b>реализация в Microsoft Hyper-V</b>
Архитектура гипервизора	Аппаратные ресурсы предоставляются виртуальным машинам непосредственно напрямую через гипервизор	Для работы требуется дополнительно, так называемая «родительская» область (partition), через которую проходит часть команд, что приводит к потере производительности
Размер гипервизора	Относительно небольшой, 144 Мб	Microsoft Hyper-V R2 SP1 с Server Core занимает около 3.6Gb
Алгоритмы оптимизации памяти	Множество	Ограниченное число
Прямой доступ к устройствам хранения данных (снижает нагрузку на CPU узла)	Да	Нет
Поддержка Fibre Channel, iSCSI, NAS – разделяемых хранилищ и локальных SCSI/SAS/SATA	Да, всех указанных типов	Да, кроме поддержки NAS
Поддержка форматов виртуальных машин	С помощью бесплатного vCenter Converter	Необходима покупка лицензий SCVMM
Наборы ресурсов (resource pools) – позволяет объединять вычислительные ресурсы нескольких узлов в рамках кластера	Да	Нет, только в рамках одного узла
Динамическое распределение ресурсов (DRS-Dynamic Resource Scheduling), в т. ч. и хранилищ данных	Да	Нет
Расширение виртуальных дисков «на лету»	Да	Нет, только добавление новых дисков SCSI
«Прозрачный» проброс USB устройств напрямую виртуальному серверу	Да	Нет

Сформулируем требования для системы развертывания компонентов разрабатываемой системы:

- **Изоляция.** Компоненты системы не должны иметь доступа друг к другу, за исключением доступа, предусмотренного протоколом взаимодействия.
- **Управление ресурсами.** Компонент системы должен потреблять ограниченное число ресурсов (процессорного времени, оперативной памяти, места на жестком диске).
- **Отслеживание зависимостей.** Компонент системы должен разворачиваться монолитно со всеми своими библиотеками. Это требование гарантирует, что компонент будет использоваться с теми же библиотеками, с которыми был протестирован.
- **Низкие накладные расходы.** Использование системы развертывания не должно нести высокие накладные расходы для распределенной системы.

Полностью удовлетворяет представленным требованиям программа Docker [11].

Википедия определяет этот инструмент виртуализации следующим образом:

**«Docker** — программное обеспечение для автоматизации развёртывания и управления приложениями в среде виртуализации на уровне операционной системы Linux. Позволяет «упаковать» приложение со всем его окружением в контейнер, который может быть перенесён на любую Linux-систему, а также предоставляет среду по управлению контейнерами. Разрабатывается и поддерживается одноимённой компанией - стартапом, распространяется как свободное программное обеспечение под лицензией Apache 2.0. Написан на языке Go».

**Контейнером** называется образ изолируемой части системы, содержащий приложение со всеми его зависимостями. В контейнере могут содержаться приложения и их среда выполнения: нужные библиотеки, конфигурационные файлы. Для облегчения работы используется единый репозиторий (хранилище) образов. Это позволяет использовать уже готовые образы, которые необходимо только доработать для нужной задачи.

Docker эффективно используется для следующих задач:

- тестирование распределенной системы;
- отслеживание зависимостей между компонентами разрабатываемой системы.

Преимуществами Docker по сравнению с виртуальными машинами (Oracle VM VirtualBox, VMware vSphere, Microsoft Hyper-V) являются:

- низкие накладные расходы;
- наличие репозитория.

Использование программного обеспечения Docker позволит обеспечить высокую производительность при создании виртуальных вычислительных узлов и уменьшить потре-

ния процессорных ресурсов. Недостатком Docker по сравнению с другими виртуальными машинами является то, что он работает только под ОС Linux.

## Выбор операционной системы

Согласно требованиям технического задания, разрабатываемый портал должен обладать высокой доступностью, работать на типичных архитектурах ЭВМ (Intel x86, Intel x64), а так же быть экономически недорогим для сопровождения. Таким образом, можно сформулировать следующие требования к операционной системе:

- **Распространенность.** На рынке труда должно быть много специалистов, способных администрировать распределенную систему, работающую под управлением выбранной операционной системы.
- **Надежность.** Операционная система должна широко использоваться в стабильных проектах, таких как Mail.Ru, Vk.com, Google.com. Эти компании обеспечивают высокую работоспособность своих сервисов, и на их опыт можно положиться.
- **Наличие требуемого программного обеспечения.** Выбор операционной системы не должен ограничивать разработчиков в выборе программного обеспечения, библиотек.
- **Цена.**

Под данные требования лучше всего подходит ОС Ubuntu [13]. Ubuntu — это дистрибутив, использующий ядро Linux. Как и все дистрибутивы Linux, Ubuntu является ОС с открытым исходным кодом, бесплатным для использования. Его предком является дистрибутив Debian. Ubuntu прост в эксплуатации дистрибутив, имеющий систему управления пакетами (APT). Поставляется как в клиентской (с графическим интерфейсом), так и в серверной (без графического интерфейса) версиями. Ubuntu поставляется с современными версиями ПО, в отличие от дистрибутивов Debian, CentOS. Преимуществом Ubuntu по сравнению с ArchLinux, Slackware и Gentoo являются низкие требования к квалификации системных администраторов. Однако Ubuntu менее стабильна в работе, чем Debian.

## Выбор СУБД для профилей пользователей

Проанализируем техническое задание на разработку бекенда профилей. Из его сценариев использования можно выделить следующие важные моменты:

- **Отсутствует необходимость комбинирования и сравнения профилей друг с другом.** Отсюда возможно сделать вывод, что нам не требуется использовать реляционные СУБД.

- **Структура профилей может меняться.** В процессе развития портала могут появляться новые поля. По этой причине лучше всего использовать СУБД, которая не задает жесткую схему хранения данных.
- **СУБД должна быть быстрой и масштабируемой.** По этой причине СУБД должна иметь высокую скорость отклика, поддерживать репликацию и шардинг.
- **Бекенд должен осуществлять географический поиск.** Для реализации географического поиска СУБД должна поддерживать географические индексы, а также поиск объектов в заданном радиусе от выбранной географической точки.

Нереляционная СУБД MongoDB [12] полностью отвечает приведенным выше требованиям. MongoDB - нереляционная система управления базами данных. Относится к классу документо-ориентированных СУБД. В качестве языка запросов используется язык, в основе которого лежит JavaScript. JavaScript легок для освоения, что снижает порог вхождения для эксплуатации этой СУБД. Документы хранятся в виде JSON - стандартном для JavaScript способе представления данных. Эта СУБД поддерживает индексацию, географические типы данных, профилирование запросов (выявление статистики потребления ресурсов запросами), репликацию, шардирование. MongoDB не поддерживает транзакции. Эта СУБД не требует описания схемы данных.

В данном проекте выбрана MongoDB, так как она умеет работать с географическими индексами. Это позволяет осуществлять географический поиск. Наличие репликаций и шардирования повышает масштабируемость и надежность системы. Данная СУБД использует асинхронную репликацию (при асинхронной репликации изменение данных в репликах базы данных происходит асинхронно, то есть не обязательно во время транзакции, возможно, после ее завершения) типа «ведущий – ведомый» (данные может изменять только «ведущий», читать данные может как «ведущий», так и «ведомый»).

## Выбор СУБД для учета финансовой информации

В соответствии с техническим заданием разработка бекенда биллинга предусматривает следующие требования:

- **Безопасность хранения данных.** Несанкционированный доступ к счетам пользователей должен быть невозможен.
- **Транзакционность.** Должен соблюдаться принцип “ACID” (Atomicity — Атомарность, Consistency — Согласованность, Isolation — Изолированность, Durability — Надежность). Атомарность гарантирует, что транзакция не может быть зафиксирована частично. Согласованность гарантирует, что успешное завершение транзакции

оставит систему в согласованном состоянии. Изолированность гарантирует, что параллельно выполняемые транзакции не будут влиять друг на друга. Надежность гарантирует, что успешно завершенная транзакция будет зафиксирована в системе, а в случае сбоя, после восстановления системы, результаты транзакции не будут утеряны.

- **Масштабируемость.** Выбранная СУБД должна поддерживать репликацию, шардирование.

PostgreSQL [14] - реляционная система управления базами данных. Она является не-коммерческим ПО с открытым исходным кодом. Для работы с этой СУБД существуют библиотеки для таких распространенных языков программирования как Python, Ruby, Perl, PHP, C, C++, Java, C#, Go. Она работает под управлением многих операционных систем: Linux, MacOS, Windows, FreeBSD, Solaris и многих других. PostgreSQL поддерживает распределенные транзакции, что позволяет использовать его в нашем проекте для хранения финансовых данных. По сравнению с MySQL система PostgreSQL лучше работает с репликацией, так как в ней существует журнал (средство восстановления системы в случае сбоя) физической модификации страниц. PostgreSQL осуществляет асинхронную репликацию типа «ведущий – ведомый».

Выбор СУБД PostgreSQL для хранения финансовой информации обеспечит надежность, безопасность и масштабируемость системы.

## Выбор языка разработки компонент портала

Проанализируем техническое задание на разработку портала. Исходя из приведенных требований к системе, можно выявить требования к языку программирования:

- **Наличие разнообразных библиотек.** Использование готовых библиотек ускоряет разработку программного обеспечения. Также важно, что благодаря использованию распространенных оттестированных библиотек, снижается вероятность ошибки. Это повышает надежность программного обеспечения. К этому же требованию можно отнести наличие стандартных средств управления библиотеками.
- **Совместимость с выбранными ранее технологиями.** Выбранный язык должен уметь взаимодействовать с ОС Linux, СУБД MongoDB и PostgreSQL.
- **Высокая скорость разработки.** На ранних этапах разработки портала технические требования часто меняются. Язык программирования должен позволять как можно

быстрее вносить изменения в коды программ . Это позволяет успешнее конкурировать с аналогичными сервисами.

Под данные требования хорошо подходит язык Python [15]. Python — высокоуровневый мультипарадигменный язык общего назначения. Относится к классу языков с динамической типизацией. Python обладает мощной стандартной библиотекой. Под этот язык существует хорошая инфраструктура: менеджеры пакетов (EasyInstall, pip), средства изоляции окружения (VirtualEnv), различные фреймворки (Django, Flask) для разработки веб-приложений. Python активно развивается.

По сравнению с C/C++ этот язык избавляет от ручного управления памятью. Но при этом скорость его работы ниже.

По сравнению с Java/C# язык Python избавляет от необходимости перекомпиляции проекта после каждого изменения. Также Python позволяет вести разработку быстрее за счет высокоуровневости языка, а наличия большого числа библиотек для разработки серверных приложений дает ему дополнительные конкурентные преимущества.

Важными преимуществами языка Python являются возможность создавать на нем кроссплатформенные приложения. Кроме того, он является свободно распространяемым языком программирования, что также является большим преимуществом.

Выбранный язык позволяет вести эффективную разработку, а богатая стандартная библиотека позволяет писать безопасное программное обеспечение. Кроме того, python позволяет работать с выбранными ранее технологиями. Таким образом, этот язык полностью удовлетворяет требованиям технического задания.

## **Выбор фреймворка для разработки портала**

К выбору фреймворка в нашей работе предъявляются те же требования, что и к выбору языка программирования:

- большое число стандартных возможностей;
- совместимость с выбранными ранее технологиями;
- высокая скорость разработки.

Для разработки системы «Портал для мастеров и клиентов в сфере бытовых услуг» будем использовать фреймворки, основанные на парадигме MVC (Model-View-Controller, модель - вид - элемент управления):

- модель представляет объекты, хранимые в системе: информацию о пользователях, данные платежных документов, информацию об активных сессиях;

- вид отвечает за визуализацию моделей. К примеру, для мобильной версии портала и для портала, ориентированного на ПК, могут применяться разные варианты отображения;
- элемент управления отвечает за взаимодействие пользователя и программного обеспечения.

Django[16] — свободно распространяемый фреймворк для разработки серверных приложений на языке Python (поддерживается вторая и третья версии языка). Фреймворк Django позволяет разрабатывать серверные приложения по парадигме MVC. В *Модели* прописываются классы для ORM (Object-Relation Mapping), которые позволяют связывать объекты классов с сущностями базы данных. В представлении (View) прописывается то, как отображать модель пользователь (обычно это HTML шаблоны). Элемент управления (Controller) должен содержать в себе логику каждой конкретной страницы.

Среди аналогов Django стоит отметить Flask (Python), но его возможности сильно ограничены по сравнению с Django. Для языка программирования Ruby существует фреймворк Ruby On Rails, который имеет схожие возможности.

По сравнению с аналогичными фреймворками Django обладает следующими преимуществами:

- проработанная документация (существует и на русском языке);
- парадигма разработки “явное лучше, чем неявное”, что позволяет удешевить поддержку приложений;
- разнообразие возможностей.

В результате проведенного анализа для разработки портала был выбран фреймворк Django.

## **Обеспечение надежности портала**

В данной системе «Портал для мастеров и клиентов в сфере бытовых услуг» для обеспечения надежности функционирования СУБД будет применяться репликация типа «ведущий – ведомый» [17] и шардинг [12]. Для обеспечения надежности данных СУБД необходимо разработать скрипт для автоматического создания резервной копий базы данных по расписанию.

Для фронтенда и бекендов целесообразно применить зеркалирование [18]. Это обеспечит отказоустойчивость системы: в случае сбоя любого из ее узлов запросы на чтение данных будут выполняться. Отказоустойчивость фронтенда возможно за счет его зеркалирования на несколько серверов. Выбор одного из зеркалируемых серверов осуществляется



с помощью прописывания в DNS (система доменных имен) записях адресов всех серверов, на которых располагаются фронтенды. При выполнении запроса к portalу браузер клиента выполняет запрос к DNS серверам и получает в ответ список ip-адресов фронтендов portalа. Далее браузер выбирает один из данных фронтендов и осуществляет к нему запрос. Отказоустойчивость бекендов будет выполняться за счет их зеркалирования, выбором нужного бекенда (по запросу к portalу) занимается фронтенд.

## ***Лабораторная работа 8***

**Цель работы:** разработать физический дизайн Web-portalа.

**Задачи и порядок выполнения работы:** изучить современные технологии разработки распределенных систем [11-18], (например, представленные для модельного Web-portalа) и рассмотреть возможность их использования для разработки Web-portalа выбранной тематики.

**Форма отчета по лабораторной работе** – это текст в произвольной форме с обоснованием выбора ОС, СУБД, языка разработки компонент portalа и фреймворка. Указать используемые технологии по обеспечению надежности portalа.

## ***Вопросы для самоконтроля***

1. Сформулируйте роль виртуализации для разработки Web-portalа.
2. Укажите возможные требования к выбору СУБД при проектировании portalа.
3. Перечислите известные Вам фреймворки.
4. Назовите технологии по обеспечению надежности portalа.

## **ЗАКЛЮЧЕНИЕ**

В методических указаниях «Методология программной инженерии» даны задания для выполнения 8 лабораторных работ по курсу «Методология программной инженерии». Указаны порядок выполнения работы и форма представления отчета по лабораторной работе. Предложены рекомендации по выбору и использованию методологий и технологий программной инженерии для проектирования модельного Web-портала.

Использование данных методических рекомендаций позволит выполнить задания лабораторного практикума в соответствии с учебной программой по курсу «Методология программной инженерии» и обеспечит формирование у студентов требуемых профессиональных компетенций:

- формализовать требования заказчика и написать в соответствии с ГОСТ техническое задание (ТЗ) на разработку ПО;
- использовать современные парадигмы, методы и нотации программной инженерии для построения моделей предметной области, для моделирования процессов и структур данных;
- выделять основные компоненты при разработке ПО и определять связи между ними, описывать интерфейс и поведение всех компонент;
- выбирать технологическую платформу для разработки ПО;
- умение оформлять научно-технические отчеты по результатам выполненной работы.

## СПИСОК ЛИТЕРАТУРЫ

1. Исаев Г.Н. Проектирование информационных систем. – М.: Издательство “Омега-Л”, 2013. – 424 с.
2. Вишневская Т.И., Романова Т.Н. Технология программирования: Мет. указания к лабораторному практикуму. - Ч. 1. – М: Изд-во МГТУ им. Н.Э. Баумана, 2007, 59с.
3. Вишневская Т.И., Романова Т.Н. Технология программирования: Мет. указания к лабораторному практикуму. - Ч. 2. – М: Изд-во МГТУ им. Н.Э. Баумана, 2010, 46с.
4. Вишневская Т.И., Романова Т.Н. Технология программирования: Мет. указания к лабораторному практикуму. - Ч. 3. – ФГБОУ ВПО МГТУ им. Н.Э. Баумана, 2012 , №0321203886. [Электронный ресурс] Режим доступа: URL:<http://catalog.inforeg.ru/Inet/GetEzineByID/293408>
5. Техническое задание. Требования к содержанию и оформлению (ГОСТ 19.201-78). [Электронный ресурс] Режим доступа: URL: <http://fmi.asf.ru/library/book/Gost/19201-78.html>.
6. NGENIX | Российский оператор CDN. [Электронный ресурс] Режим доступа: URL: [www.ngenix.ru](http://www.ngenix.ru)
7. Артур Ejsmont. Веб Масштабируемость для инженеров Startup. – McGraw Hill Профессиональный, 2015, 416с.
8. Дэвид Кокран. Twitter Bootstrap Веб-разработка Как-К. – Packt Издательский ООО, 2012, 68с.
9. Людоговский А. Общие положения IDF. [Электронный ресурс] Режим доступа: <http://dit.isuct.ru/ivt/books/CASE/case10/idef0/met1.htm>
10. Грекул В. Проектирование информационных систем. [Электронный ресурс ] Режим доступа: <http://www.intuit.ru/studies/courses/2195/55/info>
11. Turnbull J. The Docker Book: Containerization is the new virtualization. 160th ed. - James Turnbull, 2014.
12. Seguin K. The Little MongoDB Book, 2013. [Электронный ресурс ] Режим доступа: <http://www.goodreads.com/book/show/13394729-the-little-mongodb-book>.
13. Русскоязычное сообщество Ubuntu Linex. [Электронный ресурс] Режим доступа: [www.lubuntu.ru](http://www.lubuntu.ru).
14. Richard Stones N.M. Beginning Databases with PostgreSQL. Illustrated edition ed. - Wrox Press, 2001.
15. Python. [Электронный ресурс] Режим доступа: [www.python.ru](http://www.python.ru)

16. Adrian Holovaty J.K.M. The Definitive Guide to Django: Web Development Done Right. 2nd ed. - Apress, 2009.
17. Шлоснейгл Д. Профессиональное программирование на PHP. Вильямс, 2006. 389с.
18. Зеркалирование данных веб-серверов (сайта). [Электронный ресурс] Режим доступа: <http://miranor.ru/uslugi/zerkalirovanie-dannih-serverov-saita/>
19. Буртунов О. PostgreSQL. [Электронный ресурс] Режим доступа: [http://www.sai.msu.su/~megeera/postgres/talks/what\\_is\\_postgresql.html](http://www.sai.msu.su/~megeera/postgres/talks/what_is_postgresql.html)