**ВВЕДЕНИЕ**

Во время печати 3D принтера возможны «ангельские волосы», «спагетти» или «вермишель», получаемые при срыве модели с рабочего стола. Причиной такого бедствия зачастую служит деформация слоев. Закручивание краев модели вверх приводит к тому, что экструдер цепляет модель и банально ломает или срывает ее с платформы. В целях контроля стабильности и надёжности процесса 3D печати можно рекомендовать организацию системы мониторинга работы вашего 3D принтера. Это позволит Вам не только получить гарантированно высокий результат в ожидаемое время, но и уберечь свое оборудование от поломок и преждевременного выхода из строя его важных узлов.

**1** **ОРГАНИЗАЦИЯ EXADEL**

**1.1 Общие сведенья**

*Exadel* – это компания разработчик программного обеспечения, которая обеспечивает высокое качество услуг для наших клиентов по всему миру. Компаний, стремящихся использовать мобильных и веб-технологий для стратегического преимущества обращаются к нам, потому что:

Компания предлагает полный комплекс услуг от разработки стратегии технологического консалтинга до поставок. Мы работаем совместно с ИТ-директорами и техническими директорами в разработке стратегии, с менеджерами продуктов для разработки приложений и продуктов дорожных карт; и с архитекторами, чтобы разработать архитектуру, делать выбор платформы, и обеспечить безопасность. Мы обеспечиваем разработку, QA и услуги развертывания. Мы берем на себя ответственность за доставку от конца до конца.

Компания принимает гибкие решения для качественных приложений, которые могут быстро повысить рентабельность инвестиций.

Компания предлагает проверенный набор навыков через целый ряд передовых технологий, в том числе мобильных платформ, инструментов пользовательского интерфейса, баз данных, языков программирования и промежуточного программного обеспечения.

Компания предлагает качественную доставку по разумной цене благодаря нашей глобальной сети офисов по развитию в Северной Америке и Восточной Европе.

У нас есть послужной список доставки до крупнейших предприятий в мире, в том числе Банк Америки, *Deutsche Bank, UBS, Hewlett Packard, EMC*, *AT & T*, *eBay, Disney*, и *Deloitte*.

*Exadel* находится в частной собственности со штаб-квартирой в районе залива Сан-Франциско. Основанная в 1998 году, наша глобальная сеть офисов развития стала включать офисы в шести странах.

От разработки продукта до его реализации проекта, большинство клиентов предпочитают гибкой методологии. Быстрота в центре нашей работы.

Компании нравится гибкость, потому что мы преуспеваем на доставку рабочего ПО часто сотрудничая с клиентами, а также реагировать на быстро меняющиеся требования. Тем не менее, мы понимаем, что некоторые клиенты предпочитают другие методики; Таким образом, клиент имеет последнее слово. Мы гибки и могут использовать подход, который имеет наибольший смысл для каждого проекта. Мы предоставляем проекты, основанные на гибкости для многих наших клиентов, в том числе *VeriFone, Wolters Kluwer, HP, Deloitte* и *EMC*.

Опыт *Exadel* показывает, что методология гибкой, при правильном применении, хорошо подходит для проектов с различными характеристиками.

Распределенные или совмещенную команды, короткие (1-2 месяца) долго (> 1 год), длительностей автономных приложений, веб-приложений, мобильных приложений, одиночные и многоуровневые проекты, включая проекты со значительными усилиями по интеграции

Почти все сотрудники *Exadel* принимали участие в гибком проекте в какой-то момент; большинство нашего технического персонала, в том числе разработчиков, тестировщиков, дизайнеров, бизнес-аналитиков и менеджеров проектов, имеет опыт работы в гибкой разработке. Мы также используем гибкую методологию для разработки наших собственных продуктов, таких как *Appery.io* и *RESTXpress*.

**1.2 Руководители компании**

*Fima* *Katz* является президентом и главным исполнительным директором *Exadel*. Он основал компанию в 1998 году с видением, чтобы коренным образом улучшить способ компании используют инновационные технологии для разработки критически важных бизнес-приложений. Он преуспел в реализации этого видения, растущий бизнес *Exadel* по крайней мере на 30% с каждым годом. Благодаря его видению, *Exadel* теперь является лидером на корпоративном рынке решений с открытым кодом и пионером в области использования компонентов на основе архитектуры и подходы с открытым исходным кодом, чтобы радикально снизить корпоративные затраты и время разработки в качестве примера в стратегическом партнерстве *Exadel* с *Red* *Hat* *JBoss*. Совсем недавно он направлял *Exadel* на лидирующую позицию в мобильных приложениях и извлекая преимущества облака, примером которых может служить новаторской облачной *mobileplatform, Appery.io*. Г-н Кац имеет более чем 20-летний опыт работы в индустрии информационных технологий. Он занимал управленческие и технические должности в крупных финансовых компаниях, в том числе *Wells* *Fargo* *Bank*, *Concorde* *Solutions*, и Банк Америки (где он служил в качестве главного архитектора интернет-систем). Кроме того, он основал три других успешных компаний. Его техническая экспертиза включает в себя разработку сложных, критически важных систем распределенных объектов и интеграции сложных устаревших систем предприятия в бизнес-услуг.

Г-н Кац является признанным авторитетом и оратор с открытым исходным кодом, *Java*, *XML* и *DOA* технологий и является соавтором Основное руководство к объекту Мониторы (*Wiley*, 1999), данный человек показан на картинке 1.1.



Рисунок 1.1 **–** *Fima* *Katz*

*Lynne* *M. Walter* несет ответственность за финансовые и административные операции *Exadel*, гарантируя, что компания продолжает сохранять свою послужной финансовой дисциплины и операционного превосходства. Сотрудник с 2000 года, г-жа Вальтер приносит более 25 лет полной исполнительной, оперативный и финансовый опыт работы с государственными и частными компаниями *Exadel*. Г-жа Вальтер провела широкий спектр управленческих позиций, в том числе главный операционный директор, финансовый директор, и ИРО с *Greenbrier Companies*, ведущей производственной компании с годовым доходом, превышающим $ 600 млн. Кроме того, она провела долгосрочную роль аудита в КПМГ и занимал Президентом и Советом роли позиции для нескольких некоммерческих организаций, данный человек показан на картинке 1.2.



Рисунок 1.2 **–** *Lynne* *M. Walter*

*Donna* *Burke*осуществляет контроль технологических решений *Exadel* для мобильных устройств и веб-разработки приложений на рынке корпоративного программного обеспечения. Она приносит более 25 лет глобального маркетинга, управления производством, продажами, и общий опыт управления бизнесом в области программного обеспечения и услуг. Проактивная и получение прибыли, г-жа Берк является превосходным коммуникатор способен построить бизнес с нуля до самых высоких уровней. До прихода в *Exadel*, г-жа Берк был вице-президентом и соучредитель информационных систем *LEC*, профессиональные услуги компания, предоставляющая специализированные разработки программного обеспечения и консультационных услуг для производителей оборудования, поставщиков услуг и программного обеспечения инструментальных компаний. До этого она занимала руководящие позиции в отделе управления производством, маркетинга и развития бизнеса с компанией Эрикссон, Raynet и *US* *West* передовых технологий. Г-жа Берк консультировал *Fortune* *500* компаний, а также стартапов, данный человек показан на картинке 1.3.



Рисунок 1.3 – *Donna* *Burke*

##### *Igor* *Landes* несет ответственность за качество технологических решений, поставляемых клиентам. Он также контролирует и направляет процессы разработки программного обеспечения на *Exadel*.

Игорь имеет более чем 25-летний опыт работы в должности холдинговых индустрии программного обеспечения от инженера-программиста и архитектора решений до вице-президента по применению и разработки программного обеспечения. Он имеет обширный опыт создания инструментов и комплексных интеграционных решений на основе процесса-ориентированный подход и *SOA*-архитектуры / BPM. За последние 10 лет он руководит распределенную разработку путем создания высоко под напряжением глобальные команды с акцентом на снижение рисков программного проекта и на рачительно управление всеми аспектами жизненного цикла проекта программного обеспечения. Игорь приносит *Exadel* более чем 15-летний опыт создания продуктов и решений в финансовой отрасли, в том числе по управлению капиталом, кредитование, операции с ценными бумагами, анализ с фиксированным доходом ценных бумаг, а также варианты арбитража. Это программное обеспечение помогает повысить гибкость бизнеса и эффективность работы для таких организаций, как Банк Америки, *Fidelity*, *UBS*, *Wells* *Fargo* и *Westpac*, данный человек показан на картинке 1.4.



Рисунок 1.4 – *Igor* *Landes*

*Gregory* *Katzman*отвечает за управление высокий рейтинг команд клиента обслуживания клиентов, обучение и поддержку *Exadel* в. Сотрудник с 1999 года, г-н Кацман служил в важной роли, связи, поддержание связи с расширением и оффшорных команд разработчиков *Exadel* в Восточной Европе. Г-н Кацман обеспечивает своевременную доставку проекта и эффективность для предприятия-клиента базы *Exadel* в. Обладая более чем 20-летним опытом, г-н Кацман приносит широкий спектр технических, операционных и управленческих навыков клиента к *Exadel*, Г-н Кацманпоказан на картинке 1.5.

**

Рисунок 1.5 – *Gregory* *Katzman*

**1.3 Достижения**

Управляющая компания *Wealth* *Case* *Study*. Эта компания по управлению активами является одним из крупнейших в мире групп по управлению активами с более чем $ 8 млрд в активах под управлением от имени более чем 25 миллионов частных, профессиональных и институциональных инвесторов.

Мир управления капиталом развивалась, с финансовыми консультантами и менеджерами богатства в настоящее время необходимости быть мобильными и везде, где их клиенты. Это означает, что они получают из-за их столов и попав на дорогу. Они используют свои ноутбуки, планшеты и телефоны, чтобы обеспечить практические службы на рабочем месте клиента или дома, или даже кафе. Где бы они ни были, финансовые консультанты должны иметь доступ к их всей рабочей среде "," для того, чтобы наилучшим образом обслуживать своих клиентов и перещеголять конкурентов. Кроме того, это абсолютно необходимо, чтобы все приложения и данные должны быть защищены. Эта компания по управлению активами признали необходимость гибкости и сиюминутный, секретную информацию, чтобы лучше соответствовать современный способ ведения бизнеса.

Компания поручено *Exadel* с проектированием и строительством архитектуры для мобильных веб-приложение *IPad* с целью повышения функциональности и эффективности. Некоторые из ключевых аспектов приложения включают в себя:

1) портфели клиента;

2) Календари;

3) Планирование деятельности и отслеживание;

4) списки работы;

5) Торги;

6) Геолокационный поиск клиентов "*Near* *Me*";

7) Автономный доступ.

Приложение представляет собой один магазин остановки для менеджеров богатства. Приложение обеспечивает:

1) Более многоканальные коммуникации своевременно;

2) Управление рисками;

3) мониторинг соблюдения;

4) Клиент образования;

5) Улучшение планирования.

В результате приложение было чрезвычайно успешным, со значительным улучшением производительности и эффективности, в том числе: увеличение объема продаж благодаря информации фонда до современных, повышение мобильности от бумажных записей, и более современный профессиональный имидж. С помощью приложения, богатство менеджеры были в состоянии сосредоточиться на углублении отношений консультант-клиент и повышение лояльности клиентов.

*Global* *Healthcare* Биржа, ООО (*GHX*) является одним из крупнейших в мире цепочек поставок медицинских компаний. Благодаря постоянным инновациям, GHX работает над затратами вождения из здравоохранения путем преобразования цепочки поставок здравоохранения за счет эффективного связывающих провайдеров и поставщиков:

1) статистика службы доступности для документооборота;

2) Отчет об исполнении документов.

*GHX* было видение, чтобы расширить свой существующий обмен еще дальше. Компания хотела создать более масштабируемый, надежный, высокой степенью доступности, и поддерживаемую платформу *Exchange*, которая также приспособить высокие темпы роста и новые возможности среди торговых партнеров. (*GHX* необходимо, чтобы быть в состоянии консолидировать различные системы от волны недавних приобретений.)

*GHX* партнерстве с *Exadel* для разработки *CoreX*, в отрасли самые передовые платформу для электронной коммерции для здравоохранения. *CoreX* был построен с нуля и для облака и доступа без облаков. Основные сильные Платформе *CoreX* включают: повышенную безопасность, масштабируемость, расширенную надежность, полную прозрачность и улучшенную практичность над старой платформой.

*Exadel* использовала новейшую технологию, с помощью *AngularJS*, *JavaScript* и *MongoDB*, чтобы построить платформу *GHX* в. Работает на компоненты, которые образуют надежную обработку документов, сопровождения сделок и платформу маршрутизации, он использует технику безопасности и аудита передовой практики. Платформа обеспечивает централизованное и не опровержимым сообщение след для всех видов деятельности, которые происходят в системе. Все данные защищены с шифрованием при хранении и закреплены в процессе транспортировки.

*CoreX* в настоящее время обеспечивает более 22000 медицинских услуг объектов и более 10000 поставщиков медицинских услуг и дистрибьюторов вести бизнес в электронном виде посредством обмена электронной коммерции *GHX* в.

Обеспечение безопасности мобильных *Web* с *McKesson* *Case* *Study* *McKesson*. Как старейших и крупнейших медицинских услуг компании в стране, *McKesson* играет важную роль в здравоохранении и имеет уникальное видение своего будущего. Он обслуживает более 50% американских больниц, 20% врачей, а также 100% планов в области здравоохранения. В качестве крупнейшего фармацевтического дистрибьютора в Северной Америке, он поставляет одну треть всех лекарств, используемых там каждый день.

Приложения *McKesson* используются различными группами пользователей. Для того, чтобы служить этим пользователям лучше, компания хотела радикально повысить эффективность, но и поддерживать высокий уровень безопасности, необходимый в сфере здравоохранения.

*McKesson* совместно с *Exadel* реализовать полный набор решений в области безопасности, включая *Corporate* *SSO* *Active* *Directory* (единый вход в систему) и *Identity* *Management* UI. Из-за его обширный опыт безопасности, *Exadel* удалось быстро собрать лучший набор технологий для удовлетворения потребностей в области безопасности *McKesson* по:

1) *OpenAM* (управление доступом, прав, а также сервер федерации);

2) *OpenDJ* (сервер каталогов);

3) *OpenIDM* (система управления идентификацией);

4) *LDAP* (*Lightweight* *Directory* *Access* *Protocol*);

5) *WS*-*Security* (безопасное расширение веб-сервисов для *SOAP* (протокол доступа *Simple* *Object*);

6) Федеративные безопасности;

7) *SAML* (Security Assertion Markup Language);

8) *STS* (*Secure* служба маркеров);

9) *OAuth* (протокол аутентификации).

Пример решения: *Active* *Directory* *SSO***.** Конкретные задачи, адресованные включены, позволяя пользователям корпоративного домена подписать сразу во внутренних и внешних приложениях, имеющих как внутренних, так и внешних пользователей сети, а также плавно автоопределения, если *Windows* *Desktop* *SSO* правильно настроен. Для решения этой задачи, *Exadel* создали решение с этими компонентами:

*1) SPNEGO* на основе *Kerberos* с Откат к обычной проверке подлинности форм. *XMLHttpRequest* плавно доставки маркер *Kerberos* к серверу в фоновом режиме;

2) Расширение через стандартный модуль *Windows* *Desktop* *SSO*;

3) Пример решения: *Identity* *Management* *UI.*

Для управления идентификацией, *McKesson* было множество требований. Раствор должен был бы разрешить как прямой ввод и пакетного импорта для создания учетной записи пользователя. Управление профилем пользователя должны включать в себя делегированное администрирование и возможность пользователям обновлять свои собственные профили. Возможности самообслуживания должны охватывать *Restoral* забытых идентификаторов пользователей и сброс пароля, а принудительные изменения пароля будет необходимо быть частью события безопасности обработки. Для решения этой задачи, *Exadel* создала решение с этими аспектами:

1) На основе *OpenIDM* 2.1.0;

2) Использовано чистый *HTML* / *REST* архитектуры;

3) *JQuery*, *Mustache*, *Require*.*js*, *LESS*;

4) *ForgeRock* *OpenIDM* UI послужили основой для такого развития;

5) *Active* *Directory*, поддержка *OpenDJ*;

6) *OpenAM* агент, используемый для аутентификации и авторизации.

Этот набор решений безопасности превзошел все ожидания *McKesson* в его влиянии. Мы встретились и превысил каждую цель: высокое качество, снижение затрат, улучшение эффективности и высокий уровень удовлетворенности клиентов.

**1.4 Направления развития**

В отличие от традиционных аутсорсеров, наша компания наследие является продукт-ориентированной, с проверенной разработке продуктов, которые продаются ведущими независимыми поставщиками программного обеспечения, включая Red Hat, Progress Software, и наши собственные спин-ФФ компании, Appery. (Смотрите примеры ниже.) Мы имеем большой опыт домена в мобильных, облака, API, с открытым исходным кодом, а также продуктов, которые нацелены на разработчиков программного обеспечения и финансовых услуг учреждений. Мы привлекаем талант, который процветает, когда оспаривается с новыми идеями продукции, превращая их в сильно дифференцированных продуктов для наших клиентов независимых поставщиков. Наша команда даже применяет продукт-ориентированной дисциплины пользовательских приложений, что приводит к более высокому качеству программного обеспечения, которое является более надежным и легче поддерживать.

Жизненный цикл разработки готового продукта. В рамках нашей деятельности в области развития продукта, мы предлагаем услуги, которые охватывают весь жизненный цикл управления продуктом, включая определение, внедрение, поддержка и постоянное управление высвобождением

Наша компания сотрудничаем с вашими менеджерами продуктов, чтобы завершить мысленный процесс, чтобы обеспечить ваши продукты отражают голос клиента, сорвать конкурентов, а также создавать новые категории продукции. В зависимости от ваших потребностей наши идейные услуги могут включать в себя ряд процессов обнаружения от мозговой атаки клиентов, исследования рынка до хакатон.

Службы компании обеспечивает архитектуру программного обеспечения, необходимого для удовлетворения ваших бизнес-требований. Наши архитекторы обеспечат поддержкой ваши продукты, цели, масштабируемость, безопасность, доступность и ремонтопригодность. Кроме того, наши архитекторы сделают покупки *vs*. решений рекомендаций и будет рекомендованы лучшие технологии стеки для удовлетворения ваших потребностей рынка.

*Exadel* разрабатывает комплексный план испытаний, который включает в себя блок тестирование, тестирование системы, тестирование производительности, регрессивное тестирование, и даже альфа и бета-тестирование, чтобы обеспечить ваш продукт отвечает требованиям рынка на требуемом уровне охвата. Наши инженеры QA работать в тесном сотрудничестве с нашими разработчиками, чтобы гарантировать, что дефекты обнаруживаются как можно раньше. Кроме того, мы применяем высокие уровни автоматизации для обеспечения самого высокого уровня эффективности и качества продукции.

Компания сотрудничает с вашей командой, чтобы предложить все необходимые уровни поддержки и соответствующих переключений. Мы можем выступать в качестве переднего конца для поддержки процесса, сделав первый звонок от ваших клиентов по любым каналам (телефон, электронная почта, чат и т.д.), или служить в качестве отправной точки для эскалации поддержки второго или третьего уровня. Наша команда будет адаптироваться к предпочтительной справочной службы инфраструктуры для обеспечения бесперебойной передачи обслуживания на протяжении всего процесса продажи билетов и эскалации.

Является ли ваш продукт в начале его жизненного цикла или в режиме обслуживания, наша команда может обеспечить Укрепляющая инженерные услуги для обеспечения постоянной инновации и текущее обслуживание продукта. Мы можем разработать основные версии внедрить новые функциональные возможности на регулярной каденции или в соответствии с требованиями вашей клиентской базы. Наши инженеры тесно сотрудничают с командой технической поддержки, чтобы гарантировать, что все дефекты отчеты по приоритетам и назначенную предстоящих релизов продукта.

**2 ПРИЛОЖЕНИЕ CONTROLPRINTER**

**2.1 Архитектура приложения**

Приложение *Controlprinter* состоит из 4 компонентов, что можно увидеть на рисунке 2.1.

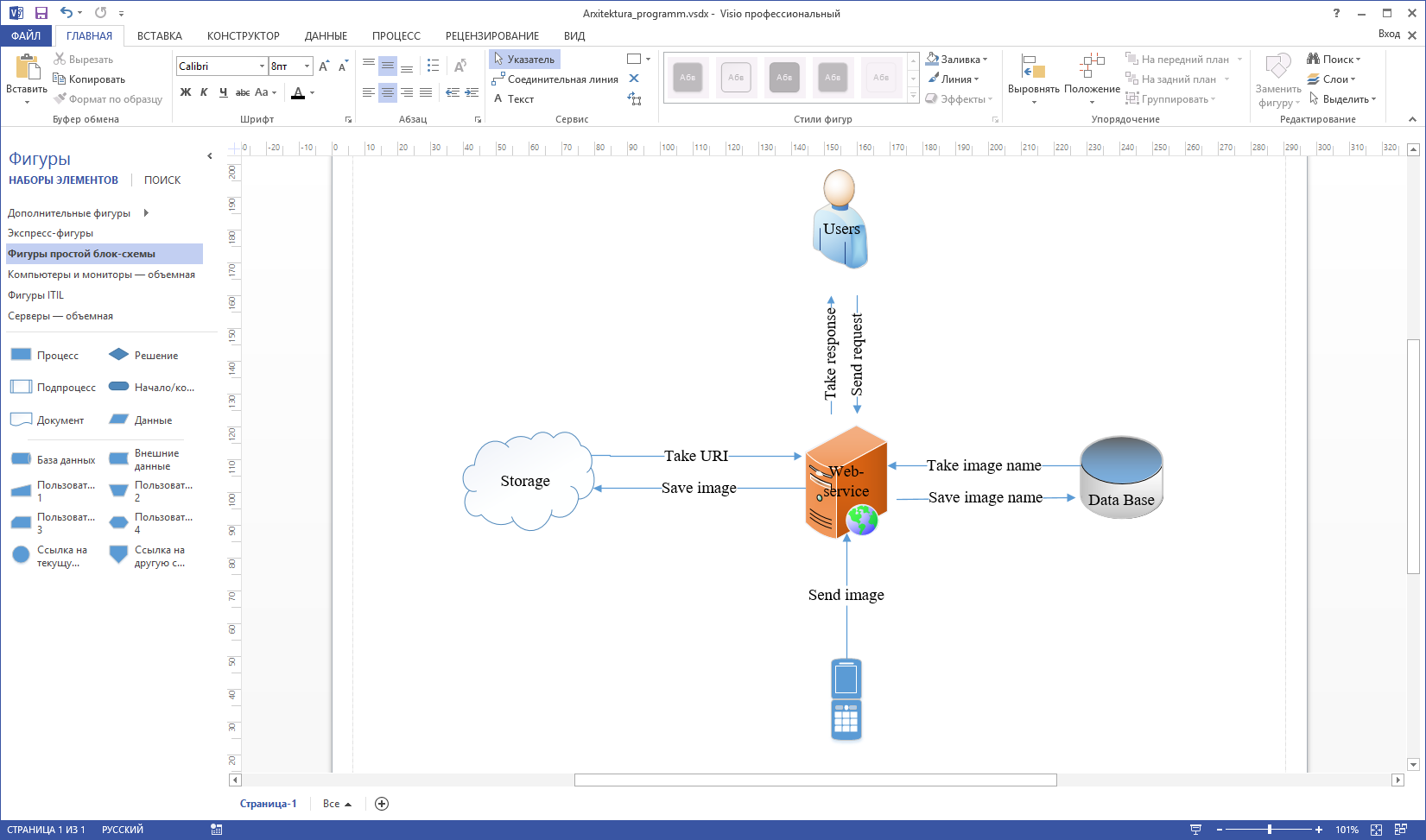


Рисунок 2.1 – Общая архитектура приложения

Приложение было разработано на языке C#, в среде разработки Visual Studio используя MVC. В качестве *storage* я использовал *Cloudinary*. Облако позволяет легко преобразовать ваши изображения на лету в любой требуемой форме, стилю и размеру, а также оптимизирует изображения, чтобы иметь минимальный размер файла для улучшения пользовательского опыта и для сохранения пропускной способности. Делать это делается путем внедрения динамического преобразования изображений и URL-адреса доставки для доступа изображения. Вы можете изменить необходимые преобразования в любое время и все преобразованные изображения будут создаваться по требованию (лениво) и доставлены своим пользователям через быстрый CDN с оптимизированным кэширования[1].

Вы можете указать необходимую высоту и ширину, определить способ, чтобы обрезать изображение и выбрать формат изображения, который соответствует вашим потребностям. Вы также можете использовать наши технологии возделывания сельскохозяйственных культур на основе распознавания лиц для фокусировки на наиболее значимой части загруженные фотографиям пользователя. Для сложных преобразований, можно использовать консоль управления или администратора API для определения названных преобразований и даже запускать множество сцепленных преобразований на изображениях. Кроме того, можно использовать консоль управления для просмотра отчетов об использовании доставки и оптимизации понимания.

База данных была создана методом *Code Fisrt* и храниться на *SQL-Server*. *SQL-Server* – это система управления реляционными базами данных (РСУБД), разработанная корпорацией *Microsoft*. Основной используемый язык запросов – *Transact-SQL*, создан совместно *Microsoft* и *Sybase*. *Transact-SQL* является реализацией стандарта *ANSI/ISO* по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия[3].

Web-service был размещён на хостинге *AppHarbor*. *AppHarbor* является полностью организованной платформой .NET в качестве службы. *AppHarbor* может развертывать и масштабировать любое стандартное приложение .NET в облако. Размещения web-service заключается в следующий шагах[2]:

1) Посылая *.NET* и *Windows*, код *AppHarbor* с помощью *Git*, *Mercurial*, *Subversion* или *Team Foundation Server* с бесплатного сервиса *Git* или через интеграции, предлагаемых в сотрудничестве с *Bitbucket*, *CodePlex* и *GitHub*;

2) Когда AppHarbor получает ваш проект будет построен сервер сборки. Если проект компилирует все модульные тесты, содержащиеся в скомпилированных сборках, проект будет запущен. Результат и прогресс состояния испытательной сборки и устройства можно контролировать на приборной панели приложения. *AppHarbor* будет вызывать любые “сервисные крючки”, которые вы добавляете, чтобы уведомить вас о результатах сборки;

3) Если все проверки пройдены приложение развернётся и настроиться на *AppHarbor* сервер. *AppHarbor* может масштабировать приложение по вертикали и по горизонтали в течение нескольких секунд для лучшего запроса для, производительности и восстановления после сбоя. *AppHarbor* балансирует нагрузки во всех случаях, работающих под управлением этого приложения. Масштабирование приложение дает более высокую пропускную способность запросов, избыточность в случае сбоя экземпляра и более высокую производительность все этапы изображены на рисунке 2.2.

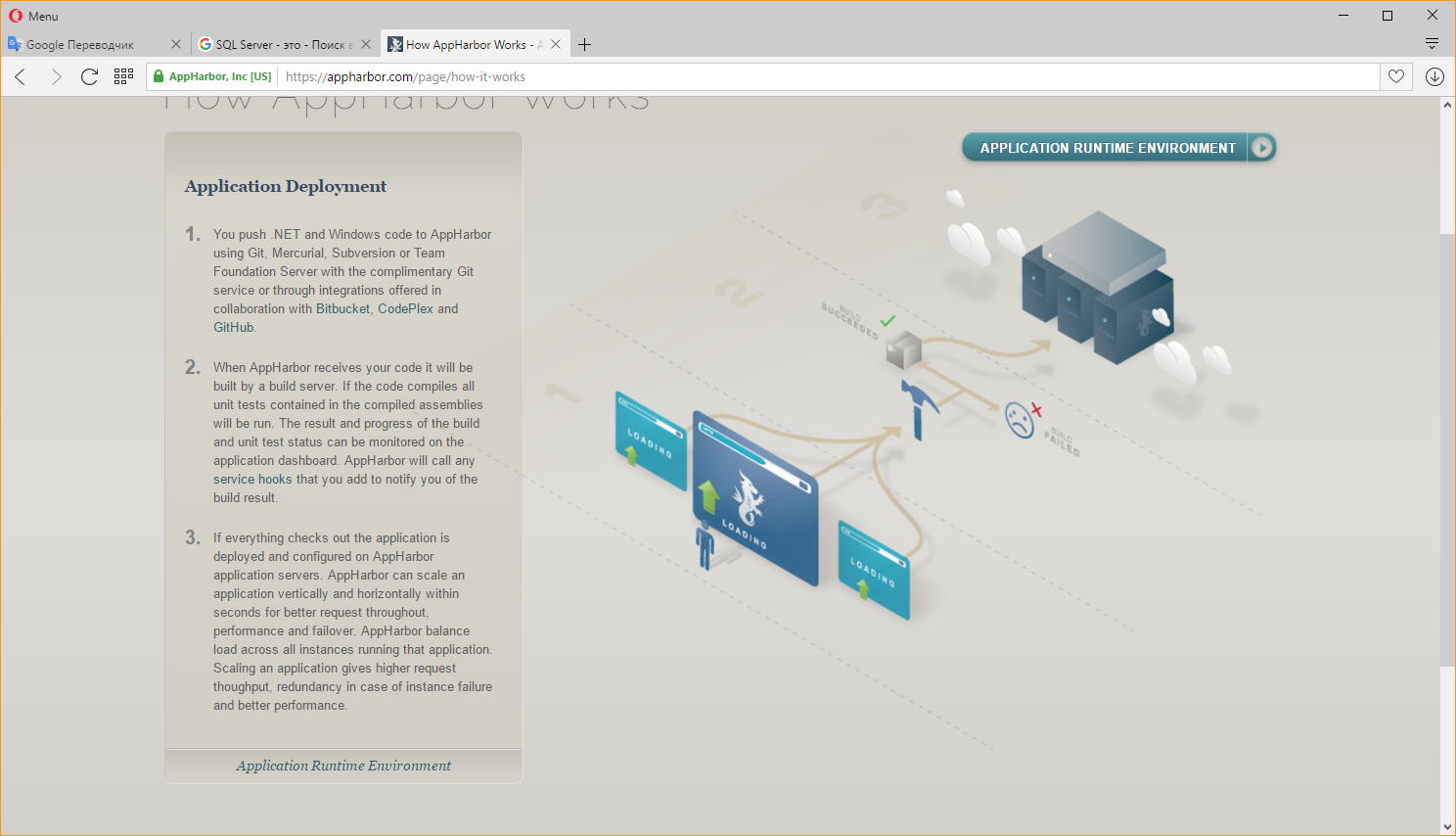


Рисунок 2.2 – Этапы размещения *web-service* на хостинге *AppHarbor*

*GitHub* – это самый большой веб-сервис для хостинга проектов и совместной их разработки. Сервис основан на системе контроля версий Git, а разработан на *Ruby on Rails* и *Erlang*.

Сервис бесплатен для *open source* проектов и предоставляет им все возможности, в том числе и SSL. Для частных проектов предусмотрены различные тарифные планы.

Слоган сервиса – «*Social Coding*» (Пишем код вместе). На футболках иногда печатают совершенно другую фразу «*Fork you*» («Ответвись!»). Фраза с одной стороны похожа на англоязычное ругательство и напоминает о неформальной атмосфере совместной разработки. С другой стороны, фраза напоминает, что создать новые форки с *GitHub* – легко и удобно. Талисманом *Github* является осьмикот, который ничего не означает, а просто был найден разработчиком Томом Престон-Вернером на фотобанке и сочтен забавным.

*GitHub* называют «социальной сетью для разработчиков». Помимо размещения кода, участники имеют возможность общаться, комментировать правки, следить за новостями друзей. С помощью больших возможностей сервиса, программисты могут объединять свои репозитории и сервис предлагает удобный для этого интерфейс. Для проектов имеются личные страницы, мини Вики и система отслеживания багов. На сервисе можно посмотреть файлы проектов с подсветкой синтаксиса.

Код проекта можно копировать и скачивать в виде архива. Первый частный репозиторий был создан 12 января 2008 года. На конце прошлого года проект уже насчитывал 1 миллион участников и более 2 миллионов репозиториев.

Многие крупные проекты размещают свои официальные репозитории на сервисе. Среди них: *Facebook*, *Twitter*, *Yahoo*, *Ruby* *on Rails*, *PHP*, *JUnit*, *jQuery*, *Microsoft* *IronRuby*, *osCommerce*. Для *open-souce* проектов использование сайта бесплатно. При необходимости иметь приватные репозитории, есть возможность перейти на платный тарифный план, что показано на рисунке 2.3.

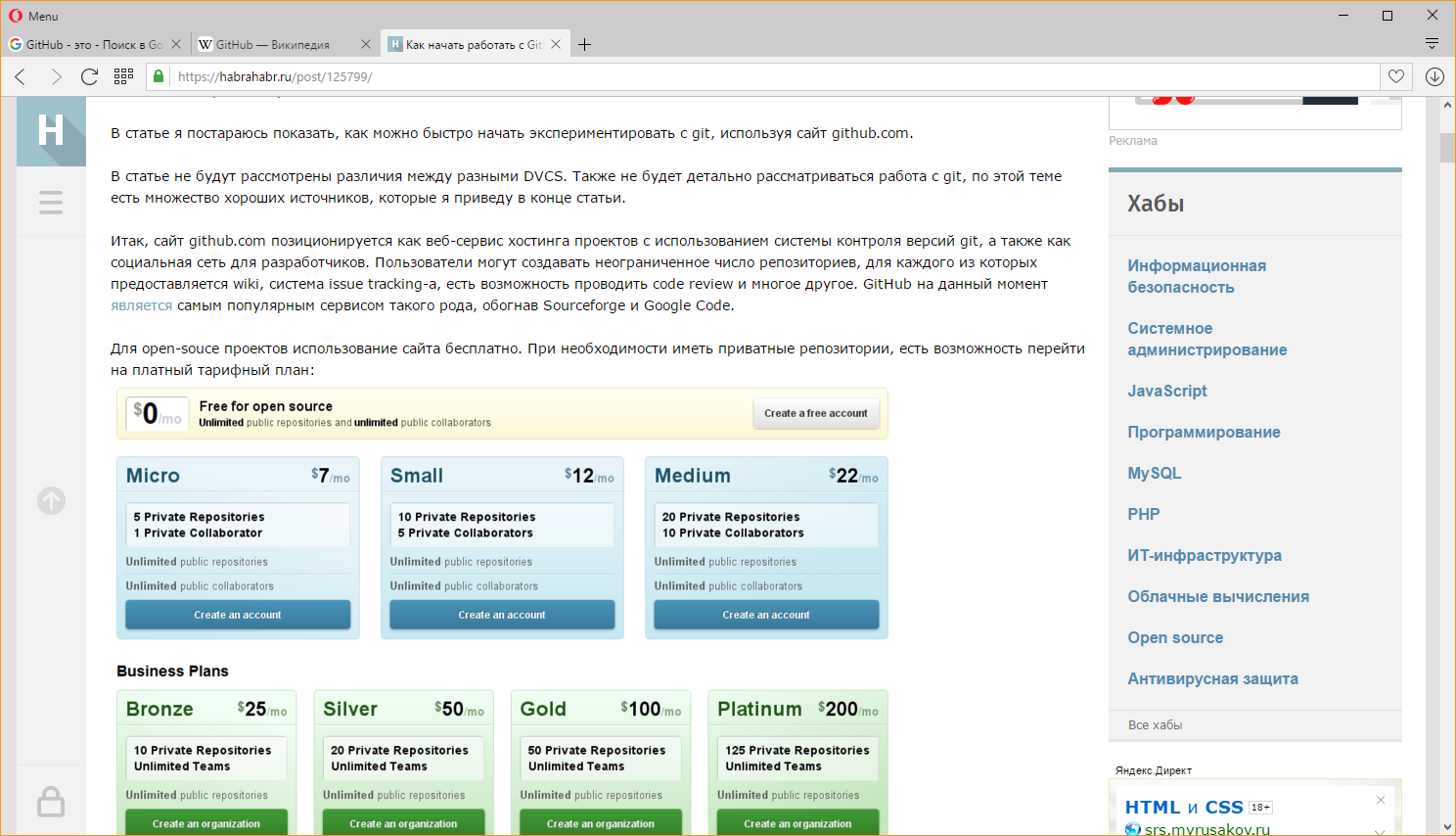


Рисунок 2.3 – Тарифный план *GitHub*

Главное отличие *Git'а* от любых других СКВ (например, *Subversion* и ей подобных) — это то, как *Git* смотрит на свои данные. В принципе, большинство других систем хранит информацию как список изменений (патчей) для файлов. Эти системы (*CVS*, *Subversion*, *Perforce*, *Bazaar* и другие) относятся к хранимым данным как к набору файлов и изменений, сделанных для каждого из этих файлов во времени, как показано на рисунке 2.4.



Рисунок 2.4 – Другие системы хранят данные как изменения к базовой версии для каждого файла

*Git* не хранит свои данные в таком виде. Вместо этого *Git* считает хранимые данные набором слепков небольшой файловой системы. Каждый раз, когда вы фиксируете текущую версию проекта, *Git*, по сути, сохраняет слепок того, как выглядят все файлы проекта на текущий момент. Ради эффективности, если файл не менялся, *Git* не сохраняет файл снова, а делает ссылку на ранее сохранённый файл. То, как *Git* подходит к хранению данных, похоже на рисунок 2.5.



Рисунок 2.5 – *Git* хранит данные как слепки состояний проекта во времени

Это важное отличие *Git'а* от практически всех других систем контроля версий. Из-за него *Git* вынужден пересмотреть практически все аспекты контроля версий, которые другие системы переняли от своих предшественниц. *Git* больше похож на небольшую файловую систему с невероятно мощными инструментами, работающими поверх неё, чем на просто СКВ.

Для совершения большинства операций в *Git'е* необходимы только локальные файлы и ресурсы, т.е. обычно информация с других компьютеров в сети не нужна. Если вы пользовались централизованными системами, где практически на каждую операцию накладывается сетевая задержка, вы, возможно, подумаете, что боги наделили *Git* неземной силой. Поскольку вся история проекта хранится локально у вас на диске, большинство операций кажутся практически мгновенными.

К примеру, чтобы показать историю проекта, *Git'у* не нужно скачивать её с сервера, он просто читает её прямо из вашего локального репозитория. Поэтому историю вы увидите практически мгновенно. Если вам нужно просмотреть изменения между текущей версией файла и версией, сделанной месяц назад, *Git* может взять файл месячной давности и вычислить разницу на месте, вместо того чтобы запрашивать разницу у СКВ-сервера или качать с него старую версию файла и делать локальное сравнение.

Кроме того, работа локально означает, что мало чего нельзя сделать без доступа к Сети или VPN. Если вы в самолёте или в поезде и хотите немного поработать, можно спокойно делать коммиты, а затем отправить их, как только станет доступна сеть. Если вы пришли домой, а VPN-клиент не работает, всё равно можно продолжать работать. Во многих других системах это невозможно или же крайне неудобно. Например, используя *Perforce*, вы мало что можете сделать без соединения с сервером. Работая с *Subversion* и *CVS*, вы можете редактировать файлы, но сохранить изменения в вашу базу данных нельзя (потому что она отключена от репозитория). Вроде ничего серьёзного, но потом вы удивитесь, насколько это меняет дело.

Перед сохранением любого файла *Git* вычисляет контрольную сумму, и она становится индексом этого файла. Поэтому невозможно изменить содержимое файла или каталога так, чтобы *Git* не узнал об этом. Эта функциональность встроена в сам фундамент *Git'а* и является важной составляющей его философии. Если информация потеряется при передаче или повредится на диске, *Git* всегда это выявит.

Механизм, используемый *Git'ом* для вычисления контрольных сумм, называется *SHA-1* хешем. Это строка из 40 шестнадцатеричных символов (0-9 и a-f), вычисляемая в *Git'е* на основе содержимого файла или структуры каталога. *SHA-1* хеш показан на рисунке 2.6.

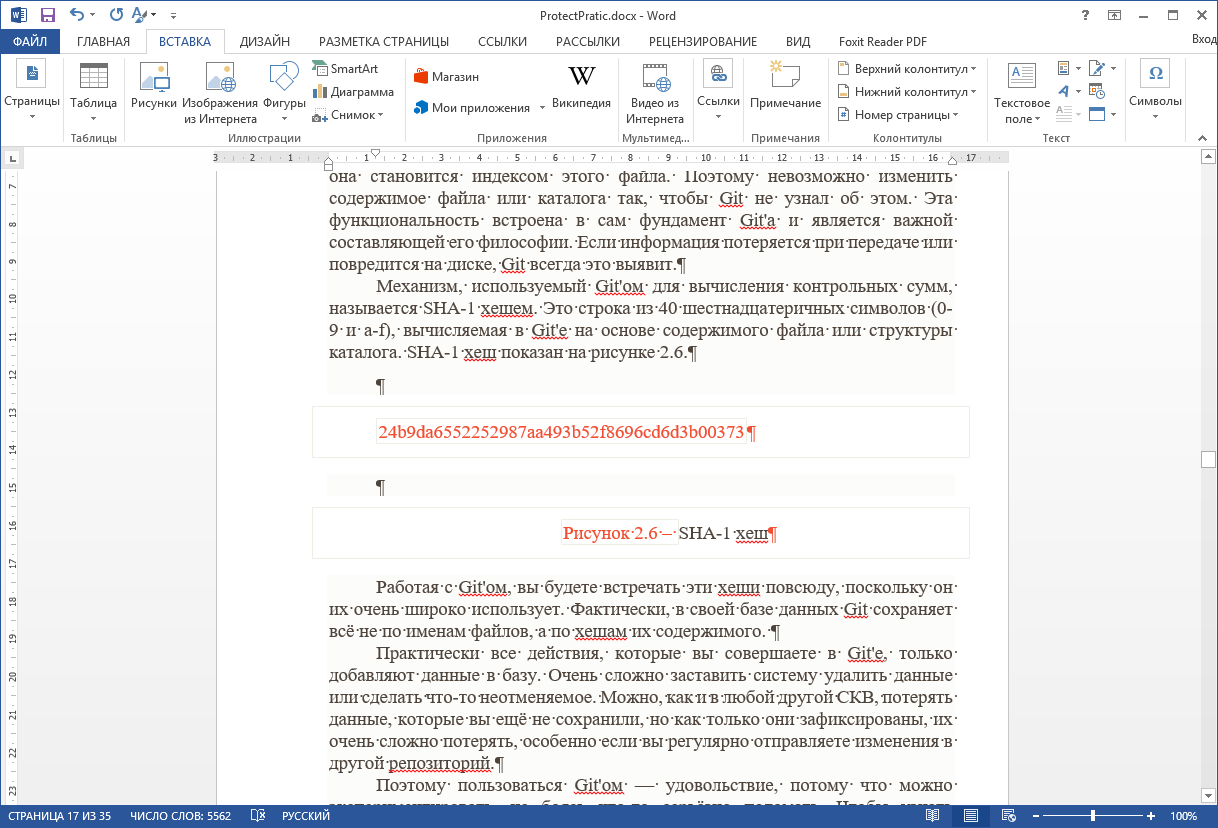


Рисунок 2.6 – *SHA-1* хеш

Работая с *Git'ом*, вы будете встречать эти хеши повсюду, поскольку он их очень широко использует. Фактически, в своей базе данных *Git* сохраняет всё не по именам файлов, а по хешам их содержимого.

Практически все действия, которые вы совершаете в *Git'е*, только добавляют данные в базу. Очень сложно заставить систему удалить данные или сделать что-то неотменяемое. Можно, как и в любой другой СКВ, потерять данные, которые вы ещё не сохранили, но как только они зафиксированы, их очень сложно потерять, особенно если вы регулярно отправляете изменения в другой репозиторий.

Поэтому пользоваться *Git'ом* – удовольствие, потому что можно экспериментировать, не боясь что-то серьёзно поломать.

В *Git'е* файлы могут находиться в одном из трёх состояний: зафиксированном, изменённом и подготовленном. "Зафиксированный" значит, что файл уже сохранён в вашей локальной базе. К изменённым относятся файлы, которые поменялись, но ещё не были зафиксированы. Подготовленные файлы — это изменённые файлы, отмеченные для включения в следующий коммит.

Таким образом, в проектах, использующих *Git*, есть три части: каталог *Git'а (Git directory),* рабочий каталог *(working directory)* и область подготовленных файлов *(staging area),* что показано на рисунке 2.7.



Рисунок 2.7 – Рабочий каталог, область подготовленных файлов, каталог *Git'а*

Каталог *Git'а* – это место, где *Git* хранит метаданные и базу данных объектов вашего проекта. Это наиболее важная часть *Git'а*, и именно она копируется, когда вы клонируете репозиторий с другого компьютера.

Рабочий каталог – это извлечённая из базы копия определённой версии проекта. Эти файлы достаются из сжатой базы данных в каталоге *Git'а* и помещаются на диск для того, чтобы вы их просматривали и редактировали.

Область подготовленных файлов – это обычный файл, обычно хранящийся в каталоге *Git'а*, который содержит информацию о том, что должно войти в следующий коммит. Иногда его называют индексом *(index)*, но в последнее время становится стандартом называть его областью подготовленных файлов *(staging area)*.

Стандартный рабочий процесс с использованием *Git'а* выглядит примерно так:

1) Вы вносите изменения в файлы в своём рабочем каталоге;

2) Подготавливаете файлы, добавляя их слепки в область подготовленных файлов;

3) Делаете коммит, который берёт подготовленные файлы из индекса и помещает их в каталог *Git'а* на постоянное хранение.

Если рабочая версия файла совпадает с версией в каталоге *Git'а*, файл считается зафиксированным. Если файл изменён, но добавлен в область подготовленных данных, он подготовлен. Если же файл изменился после выгрузки из БД, но не был подготовлен, то он считается изменённым.

В качестве мобильного устройства было использовано Android устройство. Под него было разработано приложение которые использует камеру для получения изображения и отправки его на *web-service*[4].

**2.2 Структура приложения**

Проект представляет собой разбитый на отдельные компоненты части, которые отвечают за определённую роль. В зависимости что делает определяет его роль в проекте. предоставлен на рисунке 2.8.

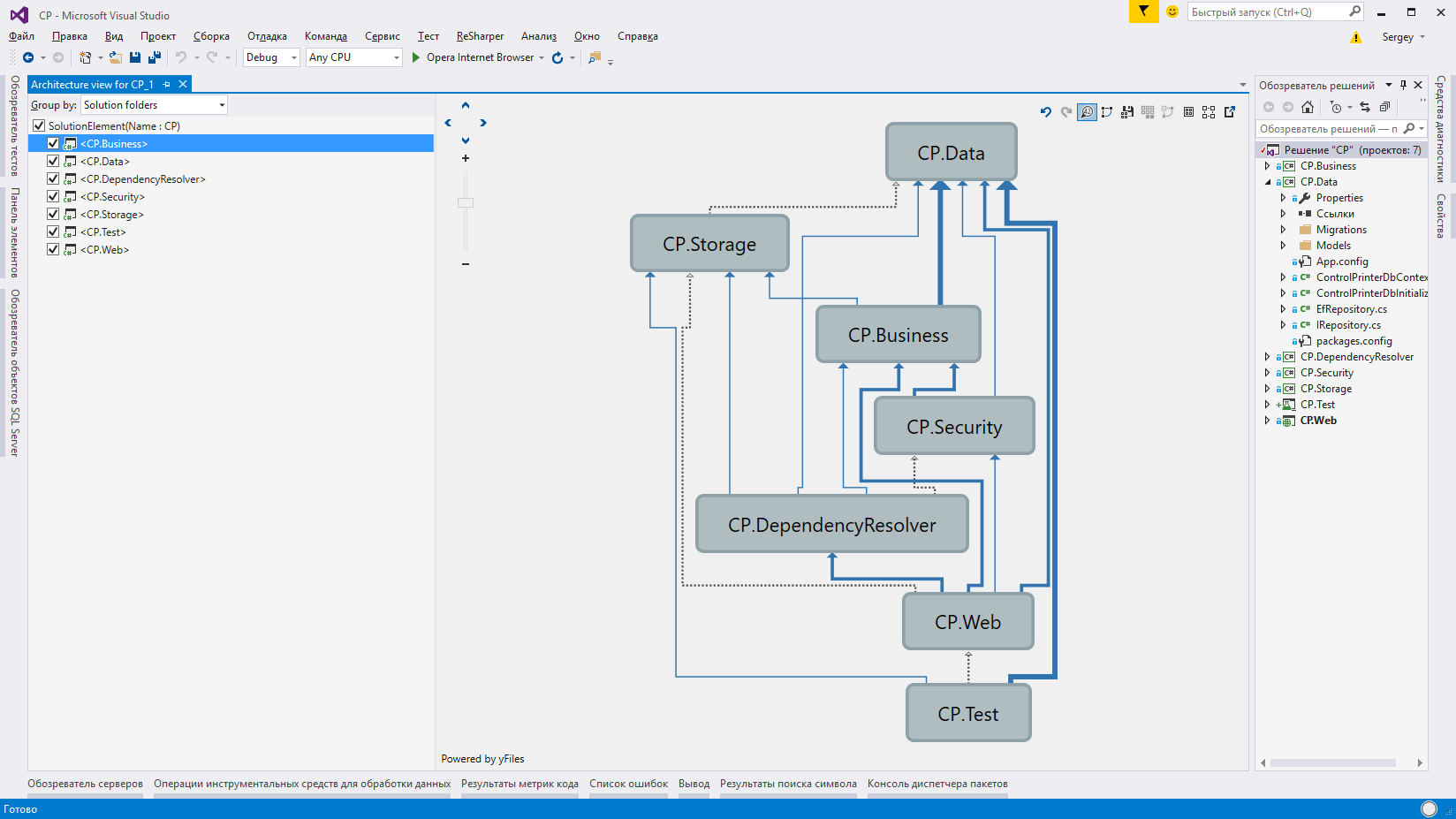


Рисунок 2.8 – Общая структура приложения

Для того что бы использовать не стандартную, а свою авторизацию необходимо было написать свою реализацию, общая структура авторизации показана на рисунке 2.9.

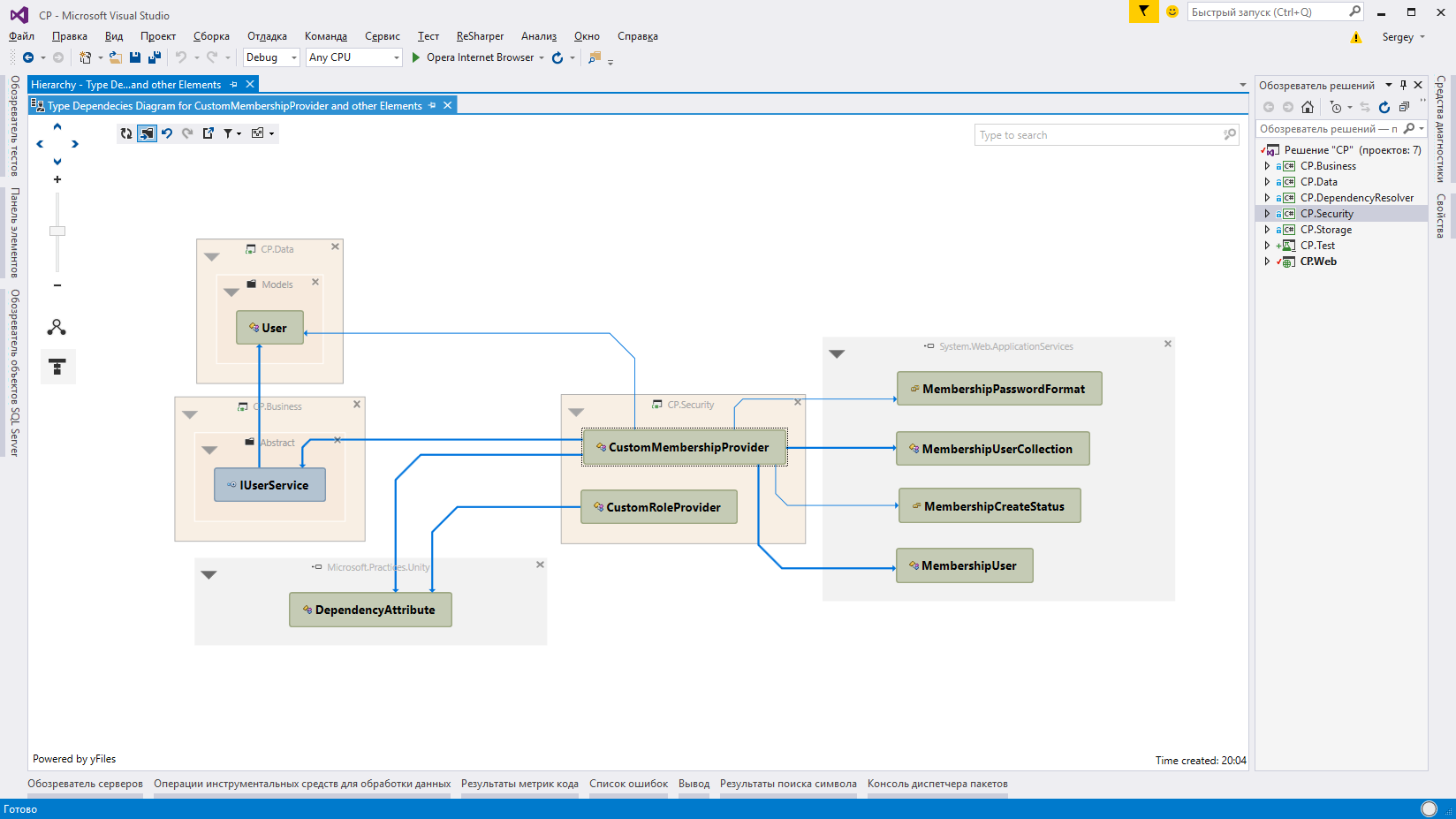


Рисунок 2.9 – Общая структура авторизации

Для того что бы в контроллерах не использовать логику приложения, было принято решения вынести её в отдельную часть программы, что показано на рисунке 2.10.

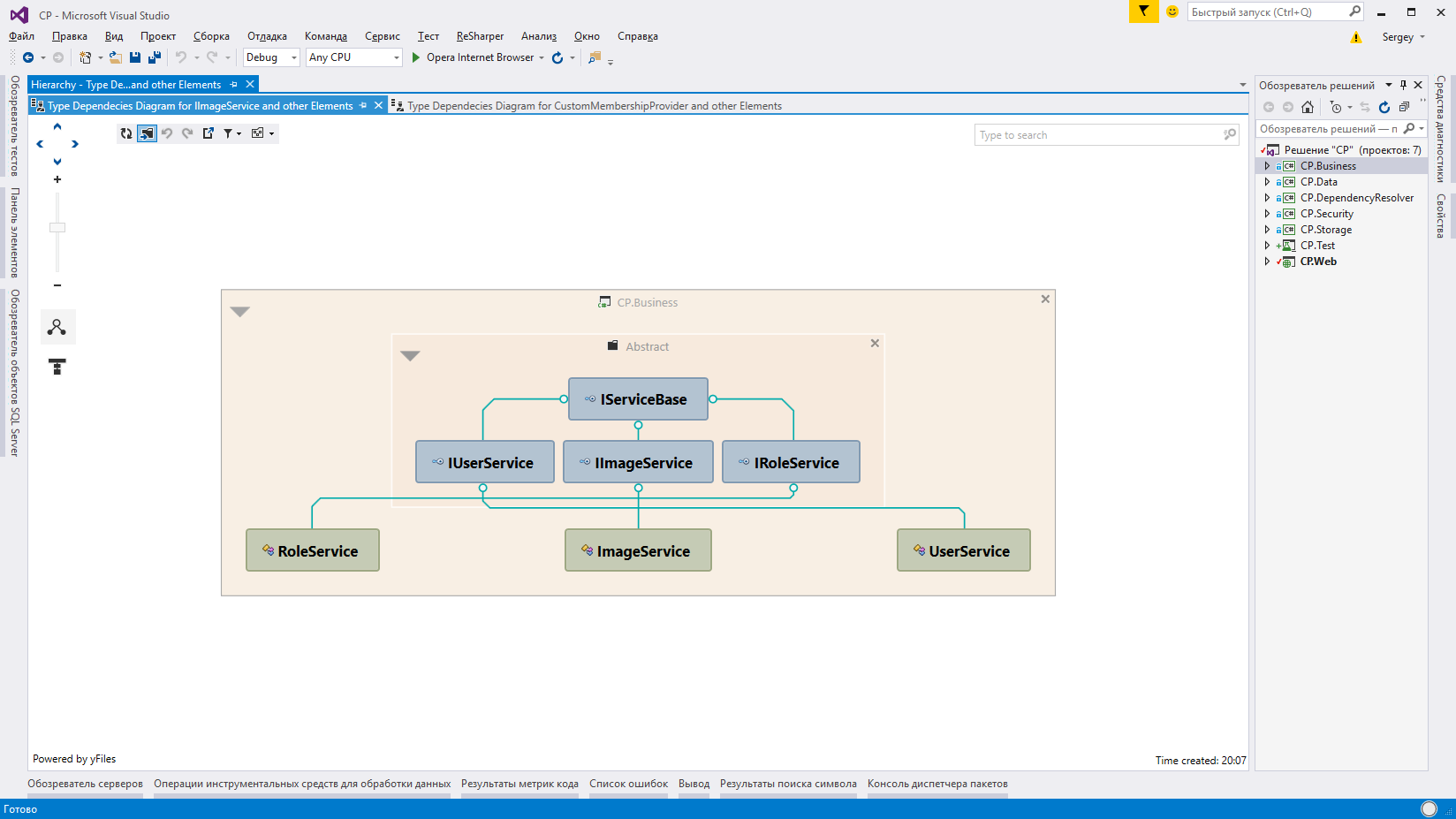


Рисунок 2.10 – Структура логической части приложения

Так же необходимо было вынести часть, отвечающую за базу данных. Заполнение её таблицами и занесение некоторых необходимых данных в неё, что вы можете наблюдать на картинке 2.11

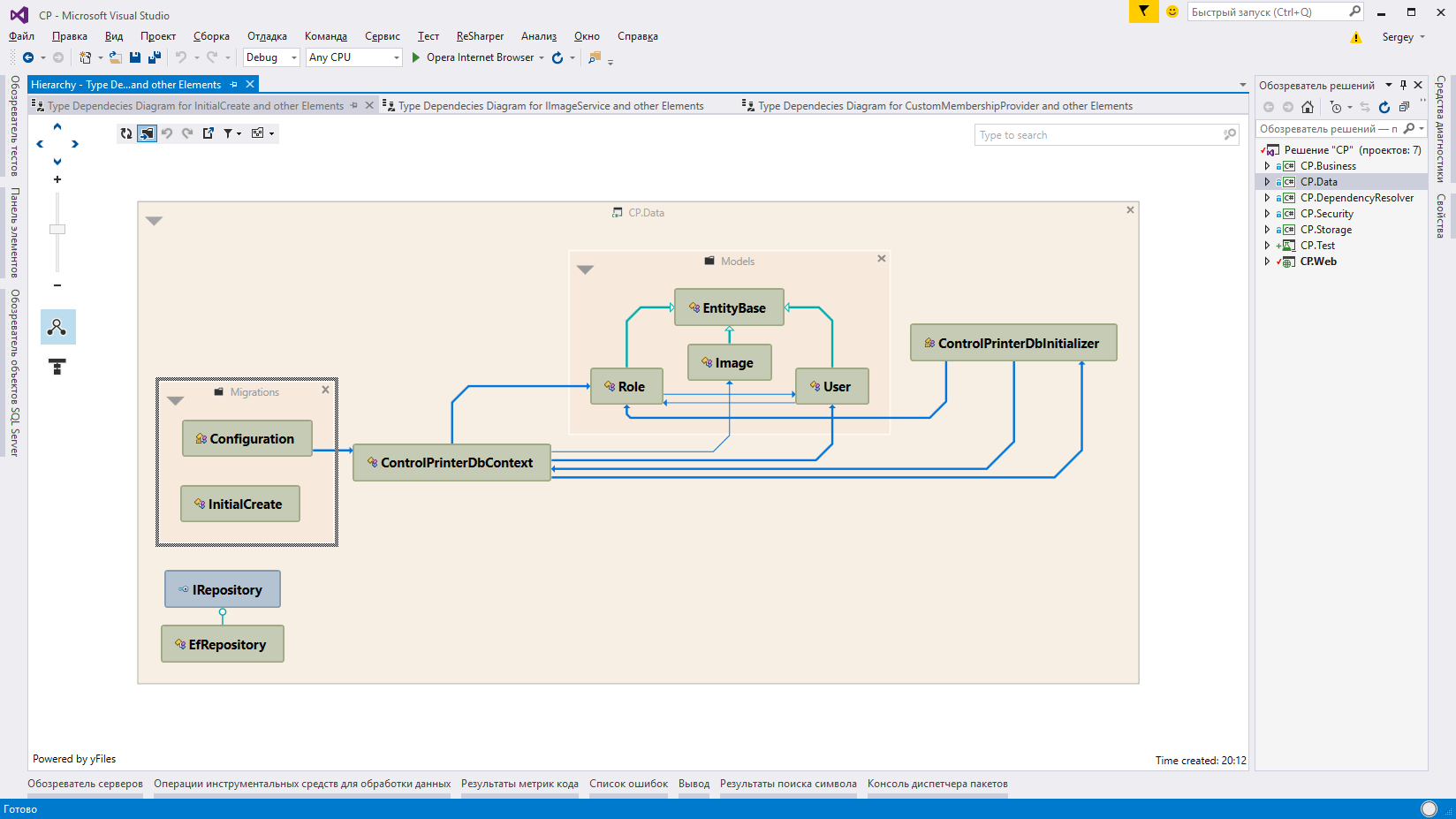


Рисунок 2.11 – Структура части приложения, отвечающая за базу данных

При работе с файлами, а в данном случаи с изображениями необходимо было разработать часть приложения, которое бы сохраняла на облако, получала ссылку на это изображение и сохраняла его, данная часть показана на рисунке 2.12.

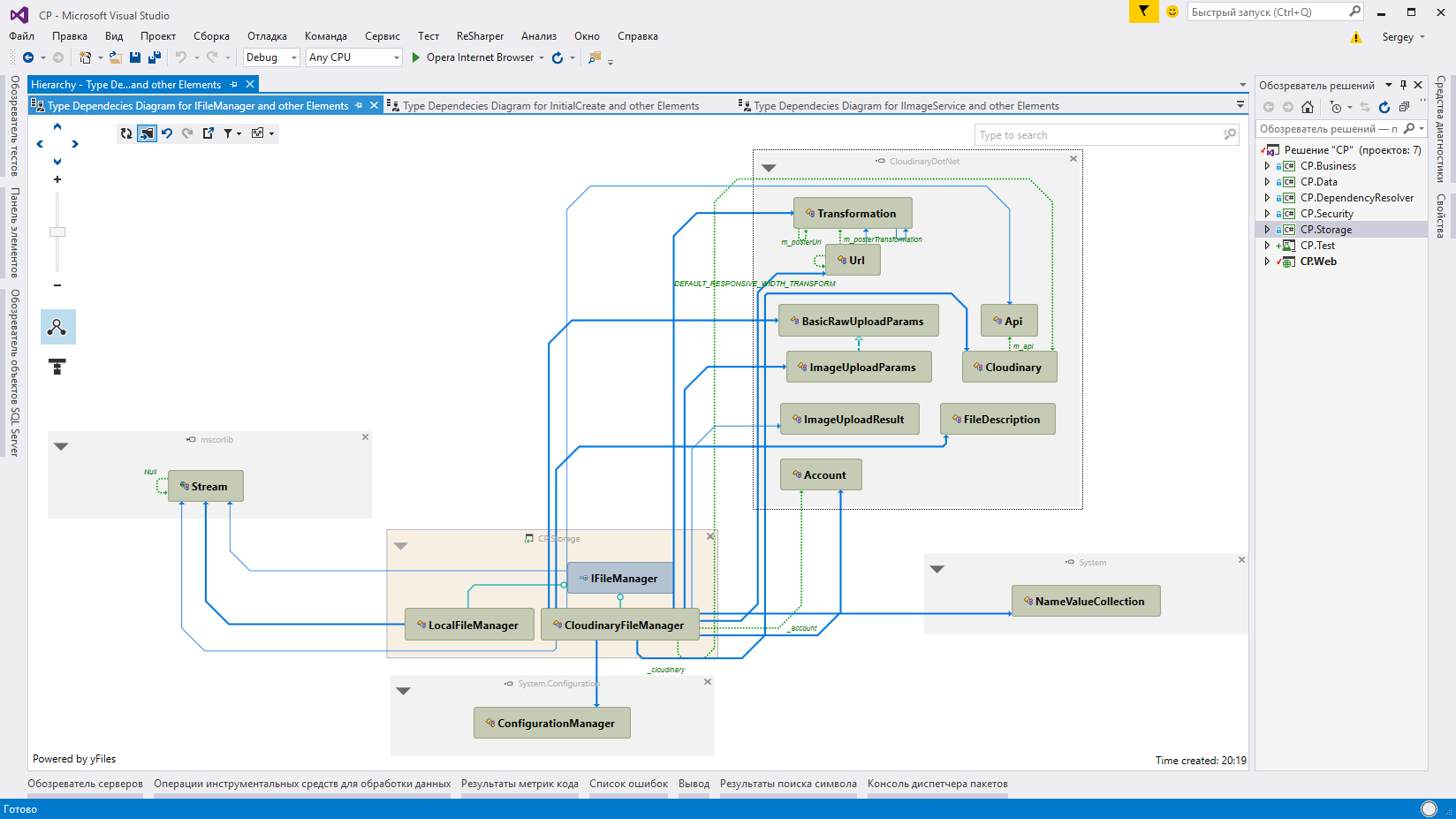


Рисунок 2.12 – Структура части приложения, отвечающая за работу с изображением

Для обработки пользователей используется класс Login, сюда переходят не авторизированные пользователи, им предлагают зарегистрироваться, в случаи если у них нету аккаунта, если аккаунта имеется, то можно ввести свои данные для авторизации на сайте. Также тут есть класс Image, который выводит пользователю изображение, структура данной части приложения показана на рисунке 2.13.

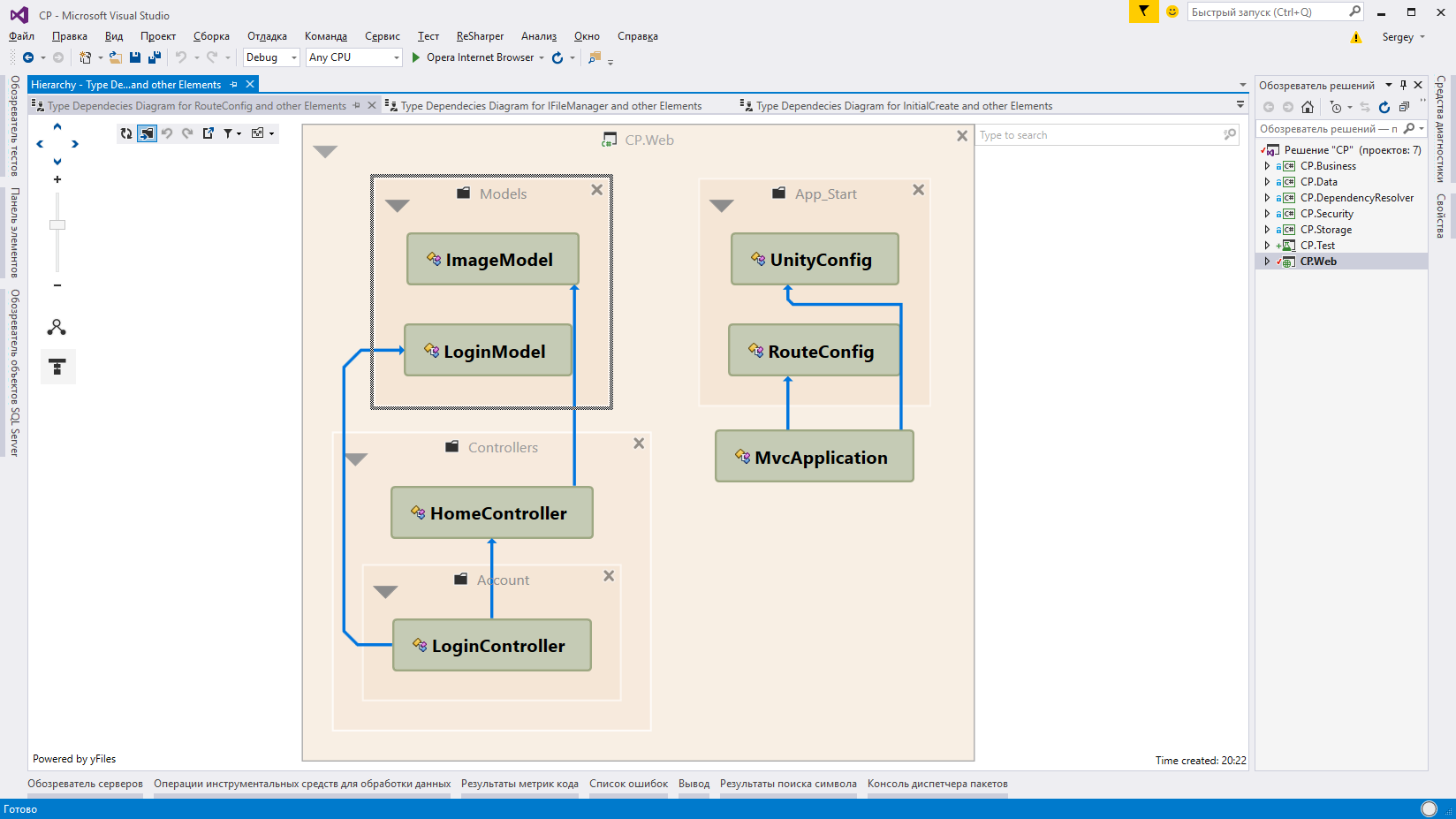


Рисунок 2.13 – Структура части приложения, отвечающая за обработку запросов от пользователей

**2.3 Тестирование разработанного приложения**

При входе на сайте, будет предложено зарегистрироваться, что показано на рисунке 2.14.

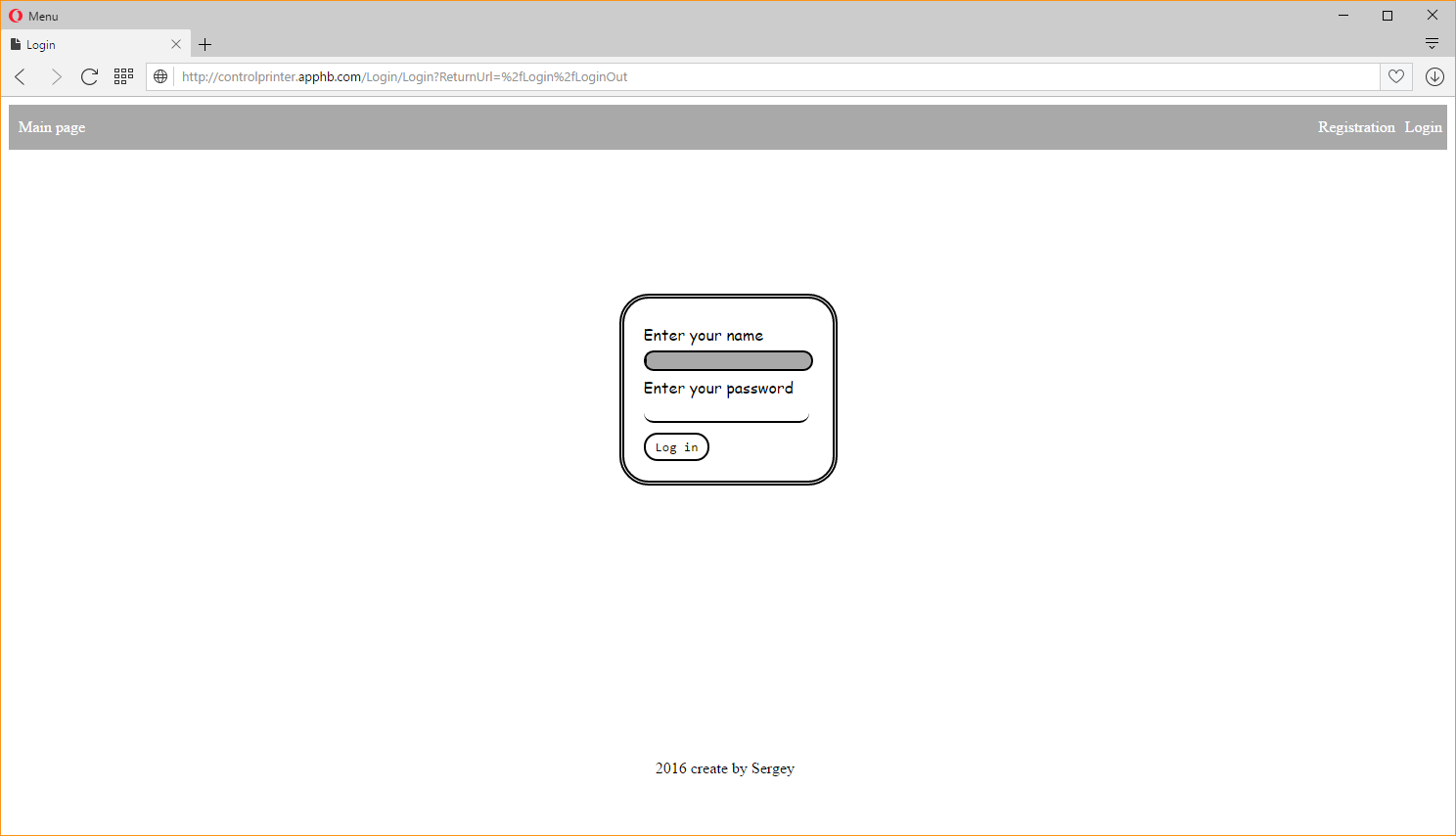


Рисунок 2.14 – Авторизация пользователя

При введение пользователем неверного логина или пароля будет показана соответствующее уведомление, что вы можете увидеть на рисунке 2.15.

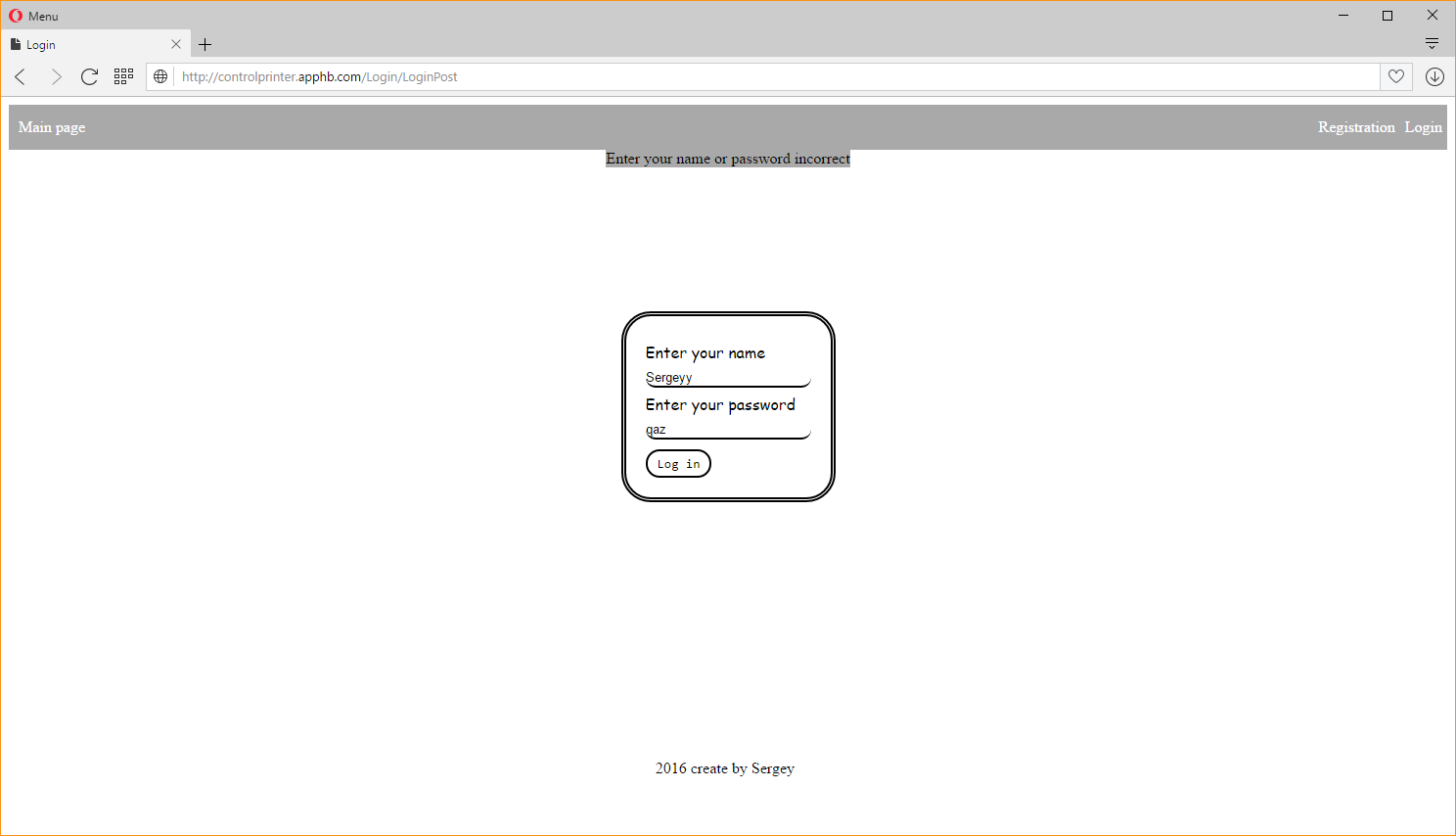


Рисунок 2.15 – Отображение уведомляющего сообщения об введённом неверном логине или пароле

Если у пользователя нету аккаунта на данном сайте, он может перейти на страницу регистрации и зарегистрироваться на данном сайте, что показано на рисунке 2.16.

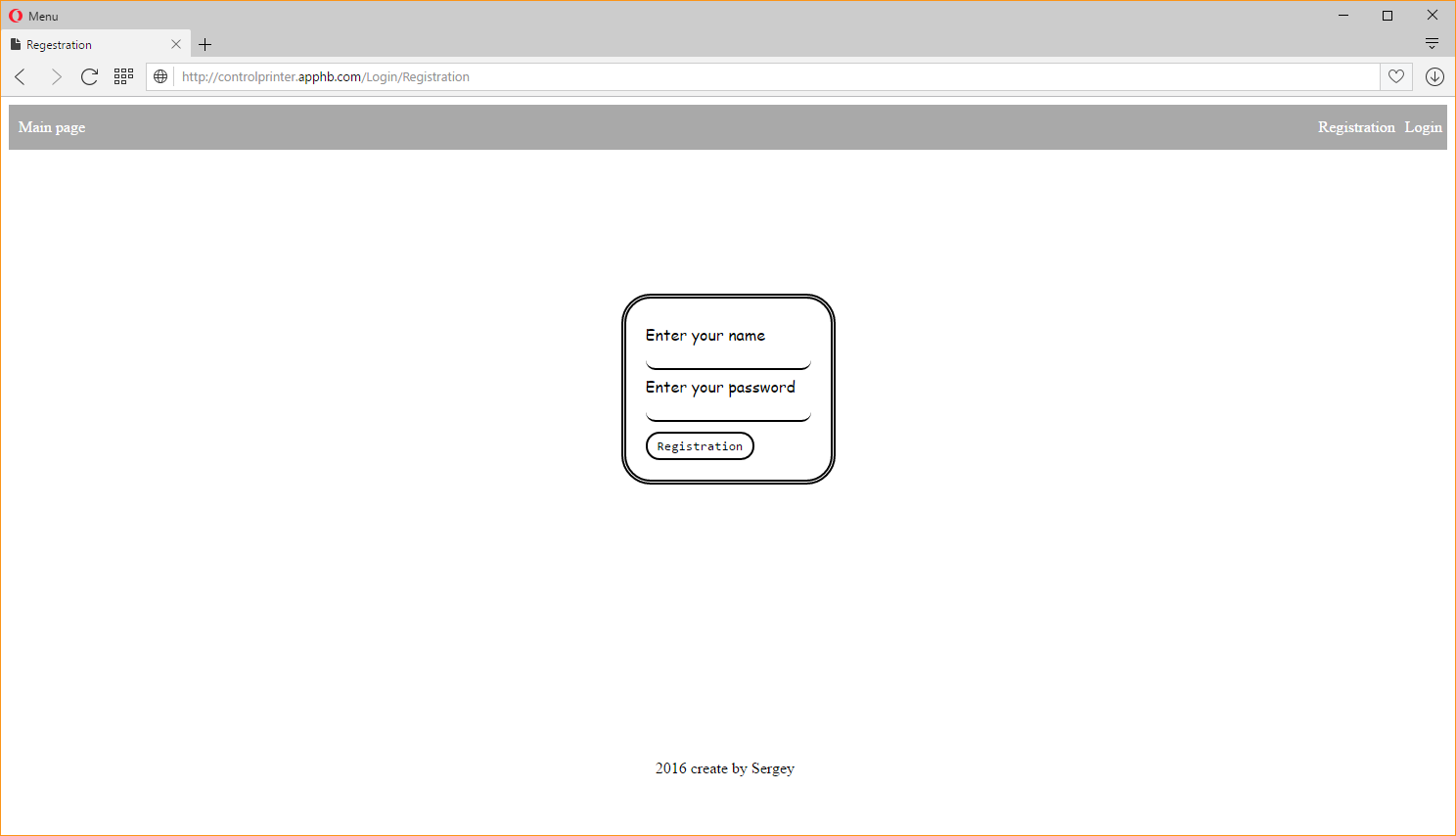


Рисунок 2.16 – Страница регистрации на сайте

Если пользователь ввёл имя, которое уже зарегистрировано, то выдаст соответствующее сообщение об этом, что вы можете увидеть на рисунке 2.17.

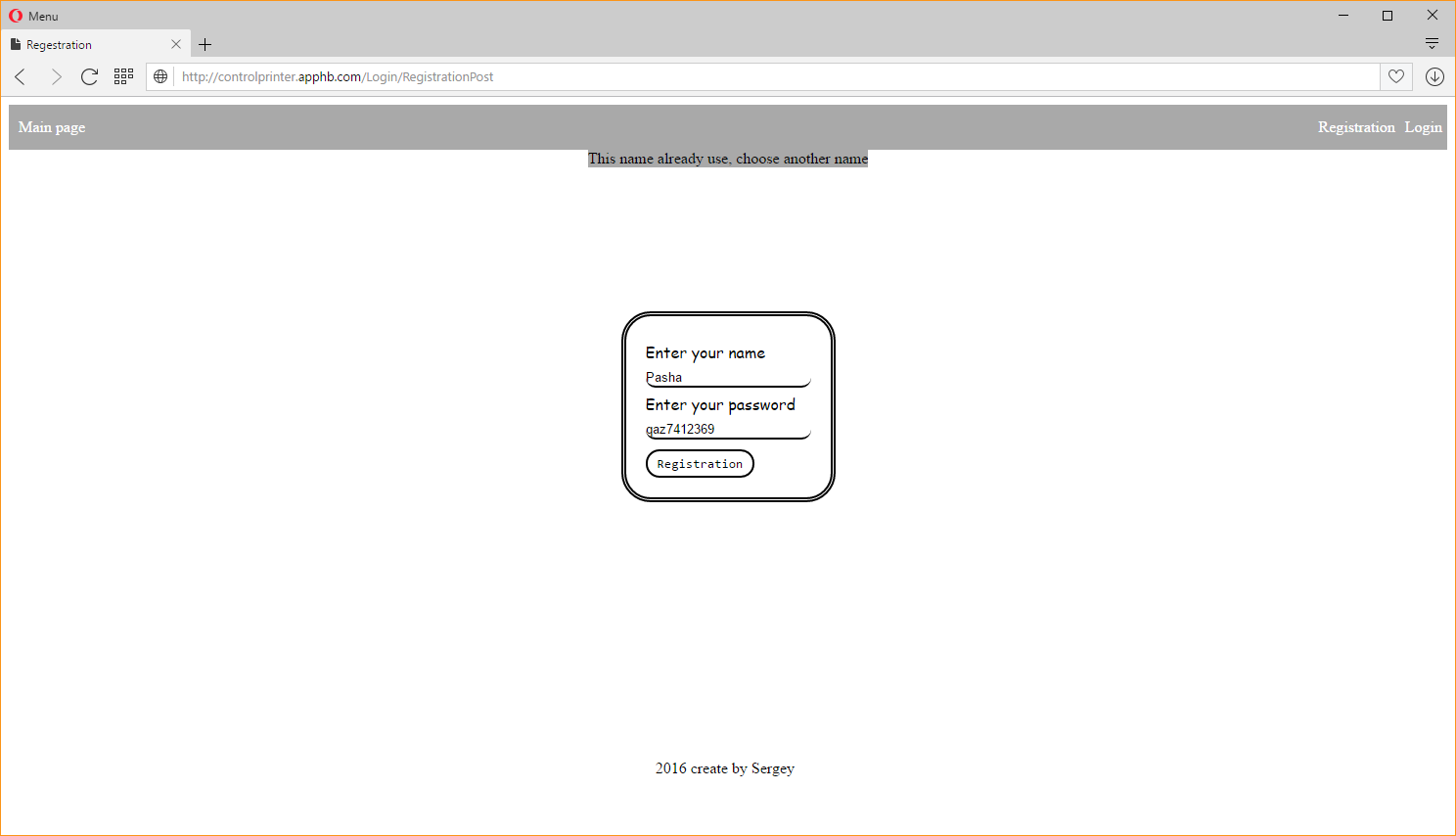


Рисунок 2.17 – Уведомление об уже использовании такого имя на сайте

При успешной регистрации вы можете авторизоваться под своим аккаунтом, после чего вы перейдёте на главную страничку, что показано на рисунке 2.18.

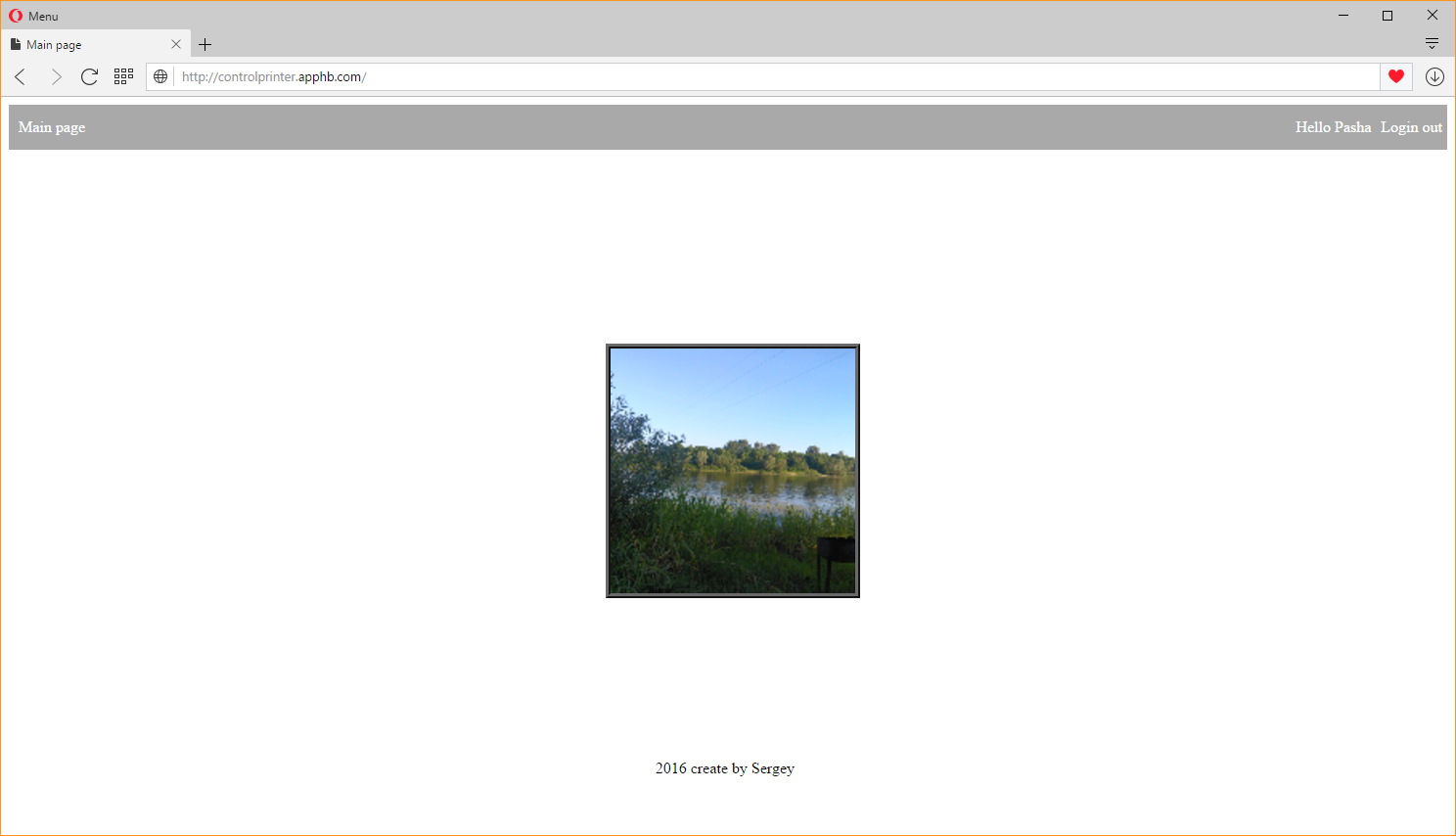


Рисунок 2.18 – Главная страница сайта

На рисунке 2.18 вы можете наблюдать, что сайт распознал вас как авторизированного пользователя в верхнем правом углу. Так же вы можете увидеть картинку в центре её, это картинка загружается автоматически при использовании приложения находящегося на android устройстве. Если авторизоваться под аккаунтом администратора, то появляется дополнительная возможность загрузить локальные изображения, что показано на рисунке 2.19.

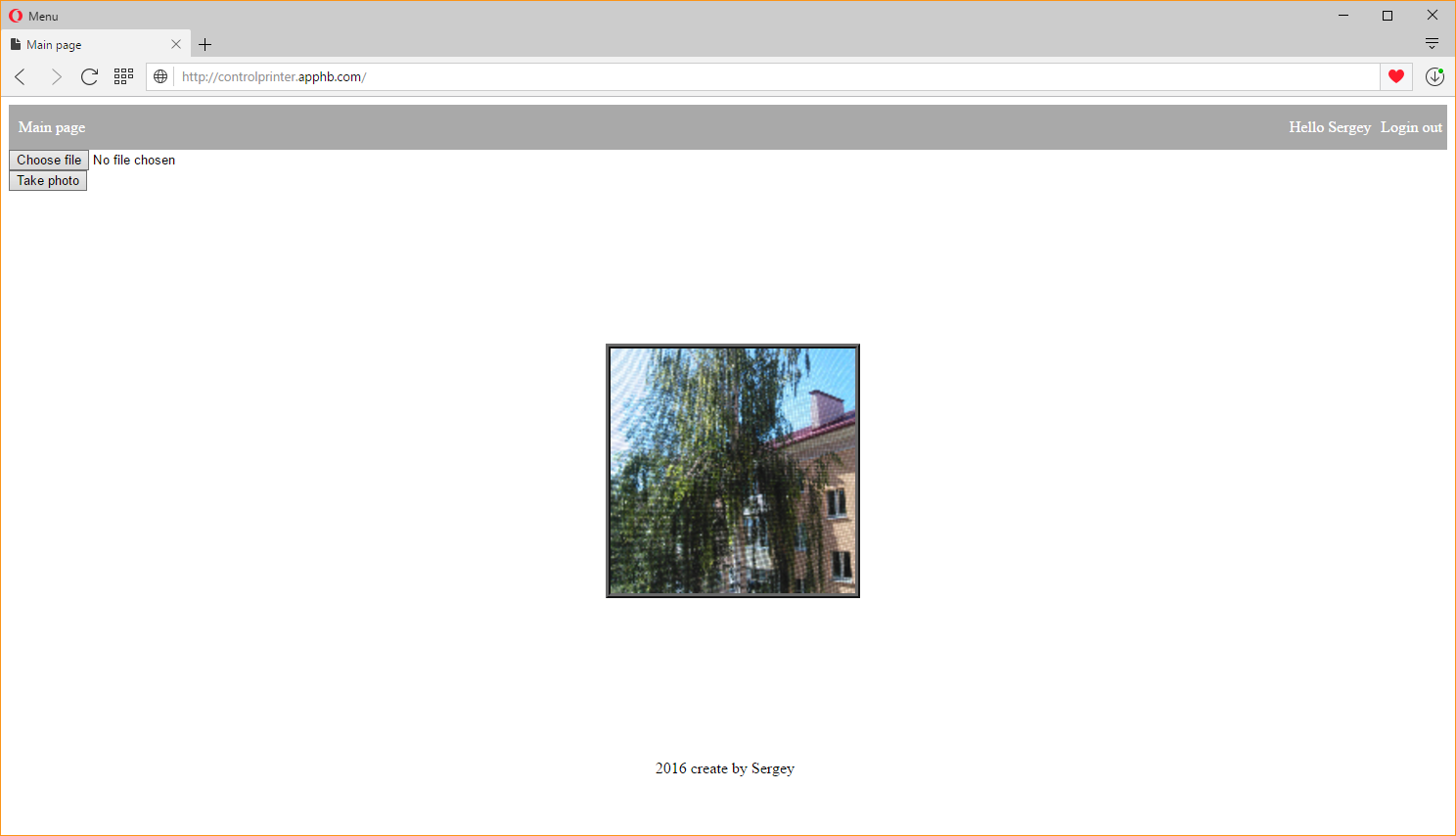


Рисунок 2.19 – Авторизация под аккаунтом администратора

Главное меню приложения на android устройстве показано на картинке 2.20.

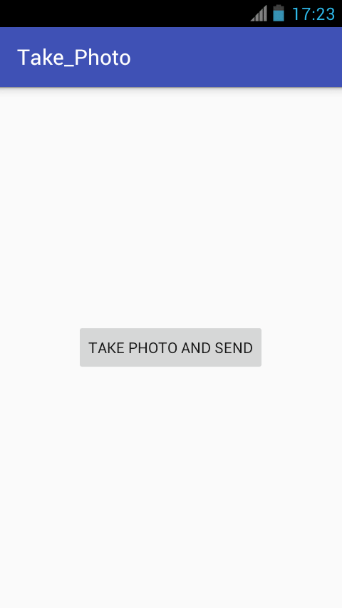


Рисунок 2.20 – Вид главного меню приложения под android

После нажатия кнопки *Take Photo and send,* мобильное устройство сделает снимок и отправит его на *web-service*, где его сохранит и поместит в storage, после чего его отобразит на главной странице, что показано на рисунке 2.21.

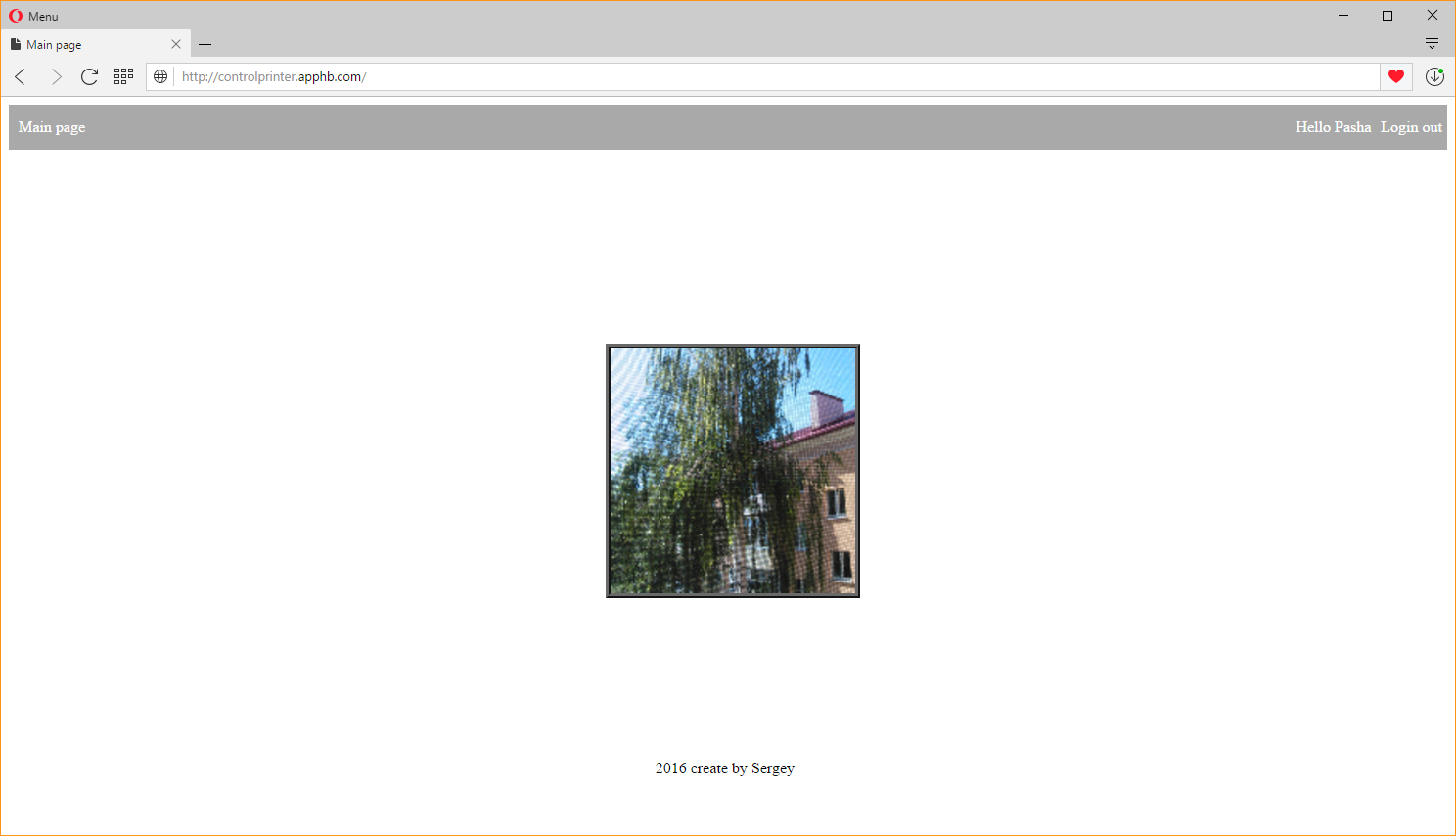


Рисунок 2.21 – Посланная картинка отобразилась на главной странице сайта

**ЗАКЛЮЧЕНИЕ**

В течение технологической практики, было разработано приложение *Controlprinter*. Данное приложение может стать частью контроля печати 3D принтера, и помогать избегать ненужных затрат материалов и времени, при печати сложных деталей. Программа проста в использовании и легко интегрируется в 3D принтер.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Облако *Cloudinary* [Электронный ресурс]. – Режим доступа: http://cloudinary.com/ – Дата доступа: 25.06.2016.

2. Веб хостинг AppHarbor [Электронный ресурс]. – https://appharbor.com/page/how-it-works – Дата доступа: 25.06.2016.

3. Герберт Шилдт Полное руководство *C#* 5.0: Питер, 2008.

4. Герберт Шилдт «Java. руководство для начинающих»: Москва, 2005.

**Приложение А**

**Код приложения**

**Класс ImageService**

public class ImageService : IImageService //Класс для работы с изображением

{

public ImageService(IRepository<Image> imageRepository,IFileManager fileManager)

{

this.ImageRepository = imageRepository;

this.FileManager = fileManager;

}

private IFileManager FileManager { get; set; }

private IRepository<Image> ImageRepository { get; set; }

public Image GetImage()

{

return this.ImageRepository.Table.OrderByDescending(o => o.DateLoad).FirstOrDefault();

}

public string GetImageUri(string fileName)//Получает URI файла

{

return this.FileManager.GetURI(fileName);

}

public void SaveImage(Image image,Stream imageStream)

{

string fileName = Guid.NewGuid().ToString();

this.FileManager.SaveFile(fileName, imageStream);

image.NameImage = fileName;

this.ImageRepository.Insert(image);

}

public void Dispose()

{

if (this.ImageRepository != null)

{

this.ImageRepository.Dispose();

}

}

}

**Класс UserService**

public class UserService : IUserService

{

public UserService(IRepository<User> userRepository)

{

this.UserRepository = userRepository;

}

private IRepository<User> UserRepository { get; set; }

public bool ValidateUser(string userName, string password)

{

return this.UserRepository.Table.Any(u => u.UserName == userName && u.Password == password);

}

public User GetUser(string userName)

{

return this.UserRepository.Table.FirstOrDefault(f => f.UserName == userName);

}

public bool AddUser(User user)

{

if (!this.UserRepository.Table.Any(u => u.UserName == user.UserName))

{

User item = new User {UserName = user.UserName, Password = user.Password, RoleId = 1};

this.UserRepository.Insert(item);

return true;

}

else

{

return false;

}

}

public void Dispose()

{

if (this.UserRepository != null)

{

this.UserRepository.Dispose();

}

}

}

**Класс RoleService**

public class RoleService : IRoleService

{

public RoleService(IRepository<User> userRepository, IRepository<Role> roleRepository)

{

this.UserRepository = userRepository;

this.RoleRepository = roleRepository;

}

private IRepository<Role> RoleRepository { get; set; }

private IRepository<User> UserRepository { get; set; }

public bool IsUserInRole(string username, string roleName)

{

return this.UserRepository.Table.Any(u => u.UserName == username && u.Role.NameRole == roleName);

}

public bool RoleExists(string roleName)

{

return this.RoleRepository.Table.Any(r => r.NameRole == roleName);

}

public string[] GetRolesForUser(string userName)

{

User user = this.UserRepository.Table.First(u => u.UserName == userName);

return (new string[] {user.Role.NameRole});

}

public void Dispose()

{

if (this.RoleRepository != null)

{

this.RoleRepository.Dispose();

}

if (this.UserRepository != null)

{

this.UserRepository.Dispose();

}

}

}

**Класс EfRepository<T>**

public class EfRepository<T> : IRepository<T> where T : EntityBase

{

private readonly DbContext context;

private IDbSet<T> entities;

string errorMessage = string.Empty;

public EfRepository(DbContext context)

{

this.context = context;

}

public T GetById(object id)

{

return this.Entities.Find(id);

}

public void Insert(T entity)

{

try

{

if (entity == null)

{

throw new ArgumentNullException("entity");

}

this.Entities.Add(entity);

this.context.SaveChanges();

}

catch (DbEntityValidationException dbEx)

{

foreach (var validationErrors in dbEx.EntityValidationErrors)

{

foreach (var validationError in validationErrors.ValidationErrors)

{

errorMessage += string.Format("Property: {0} Error: {1}",

validationError.PropertyName, validationError.ErrorMessage) + Environment.NewLine;

}

}

throw new Exception(errorMessage, dbEx);

}

}

public void Update(T entity)

{

try

{

if (entity == null)

{

throw new ArgumentNullException("entity");

}

this.context.SaveChanges();

}

catch (DbEntityValidationException dbEx)

{

foreach (var validationErrors in dbEx.EntityValidationErrors)

{

foreach (var validationError in validationErrors.ValidationErrors)

{

errorMessage += Environment.NewLine + string.Format("Property: {0} Error: {1}",

validationError.PropertyName, validationError.ErrorMessage);

}

}

throw new Exception(errorMessage, dbEx);

}

}

public void Delete(T entity)

{

try

{

if (entity == null)

{

throw new ArgumentNullException("entity");

}

this.Entities.Remove(entity);

this.context.SaveChanges();

}

catch (DbEntityValidationException dbEx)

{

foreach (var validationErrors in dbEx.EntityValidationErrors)

{

foreach (var validationError in validationErrors.ValidationErrors)

{

errorMessage += Environment.NewLine + string.Format("Property: {0} Error: {1}",

validationError.PropertyName, validationError.ErrorMessage);

}

}

throw new Exception(errorMessage, dbEx);

}

}

public virtual IQueryable<T> Table

{

get

{

return this.Entities;

}

}

private IDbSet<T> Entities

{

get

{

if (entities == null)

{

entities = context.Set<T>();

}

return entities;

}

}

public void Dispose()

{

if (this.context != null)

{

this.context.Dispose();

}

}

}

**Класс LoginController**

public class LoginController : Controller

{

[Dependency]

public IUserService UserService { get; set; }

public ActionResult Login()

{

return this.View();

}

public ActionResult LoginPost(LoginModel model)

{

if (Membership.ValidateUser(model.UserName, model.Password))

{

FormsAuthentication.RedirectFromLoginPage(model.UserName, true);

return this.RedirectToRoute(new {controller = "Home", action = "HomePage"});

}

else

{

this.ModelState.AddModelError(string.Empty, "Enter your name or password incorrect");

return this.View("Login");

}

}

public ActionResult Registration()

{

return this.View();

}

public ActionResult RegistrationPost(LoginModel user)

{

if (this.UserService.AddUser(new User {UserName = user.UserName, Password = user.Password, RoleId = 1}))

{

return this.RedirectToAction("HomePage","Home");

}

this.ModelState.AddModelError(string.Empty, "This name already use, choose another name");

return this.View("Registration");

}

public void LoginOut()

{

FormsAuthentication.SignOut();

FormsAuthentication.RedirectToLoginPage();

}

protected override void Dispose(bool disposing)

{

if (this.UserService != null)

{

this.UserService.Dispose();

}

base.Dispose(disposing);

}

}

**Класс HomeController**

public class HomeController : Controller

{

[Dependency]

public IImageService ImageService { get; set; }

[Authorize(Roles = "user,admin")]

public ActionResult HomePage()

{

Image img = this.ImageService.GetImage();

return this.View("HomePage",new ImageModel {FileName = img.NameImage,

FilePath = this.ImageService.GetImageUri(img.NameImage)});

}

[HttpPost]

public ActionResult HomePagePost(HttpPostedFileBase fileBase)

{

Image image = new Image {NameImage = fileBase.FileName, DateLoad = DateTime.Now};

this.ImageService.SaveImage(image,fileBase.InputStream);

return this.View("HomePage",

new ImageModel {FileName = image.NameImage, FilePath = this.ImageService.GetImageUri(image.NameImage)});

}

protected override void Dispose(bool disposing)

{

if (this.ImageService != null)

{

this.ImageService.Dispose();

}

base.Dispose(disposing);

}

}