# Исследовательская часть

## Постановка задачи проектирования

Целью данной курсовой работы является проектирование прототипа модуля администрирования пользователей веб-приложения для визуализации данных.

Для реализации поставленной цели необходимо решить следующие задачи:

1. Изучить необходимую литературу.
2. Изучить аналоги, определить их недостатки и преимущества.
3. Определить программное обеспечение и технологии для разработки.
4. Разработать прототип графического интерфейса модуля администрирования.
5. Реализовать модуль на языке программирования высокого уровня.
6. Протестировать и выделить преимущества и недостатки модуля администрирования пользователей.

## Описание предметной области

### Общие сведения

Каждый сталкивался с необходимостью пройти авторизацию, будь это в сервисной почте, ICQ, в социальной сети или на форуме. Для этого достаточно лишь ввести правильный логин и пароль, после чего будут доступны какие-либо возможности – например, доступ к контенту, возможность общаться и писать комментарии или ответы в форумах, блогах и т.д.

Итак, авторизация – это процедура подтверждения прав на доступ к ресурсам. Как правило, эти права получить несложно – достаточно пройти регистрацию, однако есть и много исключений, когда авторизоваться могут только некоторые пользователи.

Во время регистрации пользователь обычно указывает логин, пароль и электронную почту. Пароль может быть сгенерирован автоматически и выслан на почту. Кроме этого, администратор ресурса волен потребовать от пользователя указать и другие данные, например, пол, возраст, реальное имя и т.д. Это может быть необходимо по разным причинам.

После регистрации пользователь может авторизоваться на сайте и воспользоваться какими-либо дополнительными возможностями. В социальной сети – это возможность общаться, искать и заглядывать в профили других людей, а в интернет-магазине – заказать какой-либо товар.

Авторизация идентифицирует человека, позволяет администратору или другим пользователям опознать собеседника или клиента.

Авторизация позволяет опознать посетителя сайта и одновременно ограничить права доступа к определенным его ресурсам и возможностям неавторизованным пользователям.

Возникают вопросы: «Зачем ограничить доступ с помощью авторизации? Не проще ли открыть возможности всем посетителям?» Ведь известно, что они не любят лишних процедур, даже если регистрация и авторизация занимает пару минут.

Конечно, хорошо бы полностью избавиться от подобных мер. Однако,например, только авторизация может частично оградить владельца ресурса и его посетителей от спама. Его рассылают специальные программы – спам-боты, которые автоматически заполняют комментариями с рекламой и ссылками любые места, где возможно добавить комментарий. Оградиться от них можно разными способами, и авторизация является наиболее эффективным из них.

Другая, не менее важная, цель авторизации – это получение необходимой информации о человеке, его идентификация. Естественно, эту информацию пользователь предоставляет добровольно, и используется она исключительно в целях распознавания другими посетителями. Если речь идет об интернет-магазине, авторизация значительно упрощает использование сайта. У пользователя появляется личная страничка с историей заказов, в корзине могут храниться добавленные раньше товары, а форму заказа ему заполнять уже не нужно.

Конечно, у авторизации есть много других функций и задач, но эти две являются наиболее важными.

Преимущества владельцу сайта:

1) фильтрация спама (авторизация является тем фильтром, в котором отсеивается большая часть спам-ботов);

2) возможность распознать посетителя, что в свою очередь дает:

- более подробную демографическую, географическую и иную информацию о посетителях, благодаря чему можно лучше настроить фильтры в системе контекстной рекламы и подобрать наиболее оптимальные [ключевые слова](https://www.hmx.ru/keys.html) при [раскрутке сайта](https://www.hmx.ru/);

- возможность предоставить больше услуг и возможностей (например, дополнительные функции в виде личной внутренней почты, сохранения истории заказов и т.п.), что позволяет улучшать удобство использования, привлекая больше постоянных посетителей.

3) ограничение прав пользования тем или иным контентом. Если для некоммерческого сайта это некритично, то для ресурса, где контент или какие-либо услуги предоставляются за деньги, авторизация просто необходима.

Преимущества пользователю ресурса:

1) возможность оградить себя от спама. На ресурсах, где есть авторизация, куда меньше спам-роботов и людей, зарабатывающих на спаме и рекламе;

2) возможность встретить знакомого собеседника снова. Для того, чтобы распознать друг друга, пользователям не нужно представляться каждый раз заново, достаточно пройти авторизацию;

3) дополнительные возможности: даже на форуме пользователи могут заполнять свои профили, загружать фотографии и другие файлы, поднимать друг другу репутацию и т.д. В интернет-магазинах и на других коммерческих ресурсах возможностей еще больше.

### Семантика. Сущности и отношения

Пользователи (Users)

Каждый человек, попадающий на страницу приложения относится к пользователям (user) Пользователь может быть, а может не быть аутентифицированным. Те пользователи, которые не прошли процедуру аутентификации являются анонимными пользователями.

Пользователи, которые прошли процедуру аутентификации, находятся на сайте под своей учётной записью. Если пользователь ранее не посещал сайт и не имеет учётной записи, в большинстве случаев он легко может её создать.

Анонимные пользователи имеют самый низкий уровень доступа, вследствие чего для них существует наибольшее число ограничений.

После того, как пользователь аутентифицировался, он получает роль, соответствующую его учётной записи.

Роли (Roles)

Роль - это логическая категория пользователей. Вместо того, чтобы устанавливать список прав доступа для каждого пользователя, права доступа определяются для каждой роли. Пользователю может не соответствовать никакая роль, а может соответствовать (одна или несколько). Например, пользователь может быть участником (роль member) портала и администратором (роль manager). Каждая роль определяется некоторым именем,например: Member.

Назначаемыми являются роли, которые администратор может назначить пользователям (назначенные роли будут соответствовать пользователям после прохождения аутентификации).

Неназначаемые роли это роли, которые нельзя раздать пользователям, но которые, однако, имеют место на сайте. Например, администратор не назначает пользователю роль аноним.

Неназначаемые роли следующие:

Аноним: это пользователь, который вошел на ресурс в первый раз или не авторизовался.

Авторизованный: данная роль соответствует любому пользователю, который вошёл на сайт под своей учетной записью независимо от его роли.

Назначаемые роли:

Администратор: это специальная роль, обладающая доступом ко всем ресурсам приложения.

### Описание процессов предметной области

Незарегистрированный пользователь может просматривать информацию.

Зарегистрированный обладает правом добавления информации в промежуточную базу данных приложения.

Администратор может управлять пользователями и манипулировать записями в базе данных. На основании данных, предоставленных авторизованным пользователем, администратор принимает решение о занесении данных в актуальную базу. Если же данные окажутся не подтвержденными, то администратор вправе удалить эти записи.

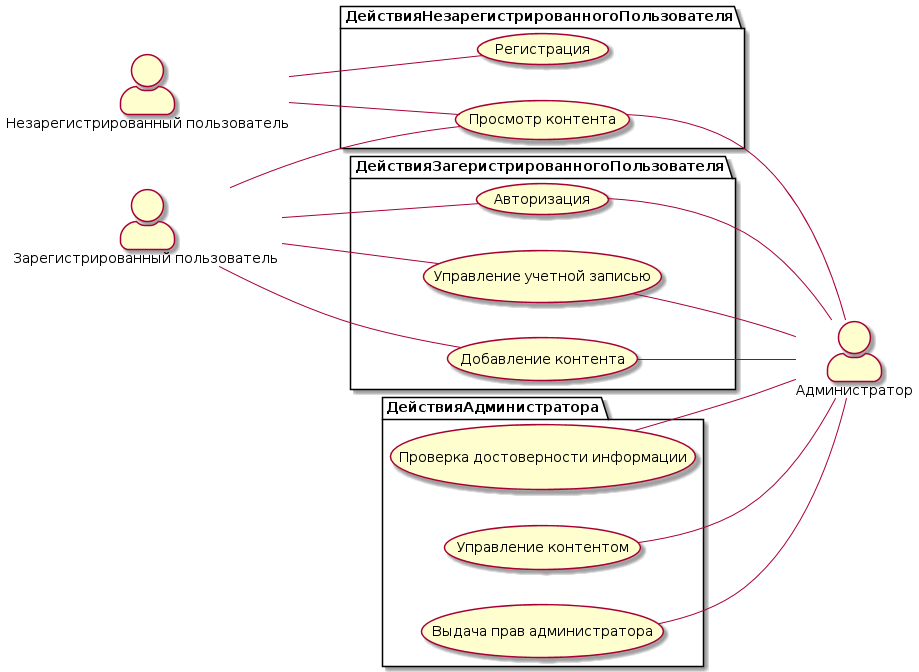


Рисунок 1 - диаграмма сценариев использования

## Анализ аналогов

Для предоставления услуг онлайн-сервисам необходимо идентифицировать пользователей. Можно создать собственное решение, но иногда проще встроить готовое, например, от популярных соцсетей.

В некоторых случаях недостаточно подтвержденного адреса электронной почты и номера телефона (которых у пользователя может быть несколько), а надо быть уверенным в его личности. Это может понадобиться, например, для сервисов из сферы финтеха или телемедицины. Тогда нужна надежная система аутентификации — от крупного банка или госорганов.

### Сервисы авторизации

#### Trusted.Id

Trusted.ID - это сервис идентификации и авторизации пользователей. Единое пространство приложений и пользователей формирует общую систему входа и пользовательских данных. Набор виджетов и модулей позволяет сделать аутентификацию через сервис незаметной для конечного пользователя, а разнообразие используемых методов и протоколов дают возможность настроить аутентификацию под требования безопасности приложения.

Сервис реализован на базе открытого протокола авторизации OAuth 2.0. Размещая виджет Trusted.ID в своей информационной системе, можно переложить все процессы идентификации и проверки пользователя на сервис. Пользователь, проходя аутентификацию, проверяется на сервисе и впоследствии он сообщает информационной системе, что пользователь себя подтвердил и соответствует требованиям.

Возможности:

1. Простая аутентификация (логин+пароль)
2. Аутентификация через соц.сети
3. Аутентификация по номеру телефона и смс
4. Регистрация в два клика
5. Аутентификация с проверкой номера телефона или адреса эл.почты позволит получать проверенные данные уже при регистрации
6. Сервис использует технологию аутентификации “единого окна”, зарегистрировавшись один раз на любом из подключенных сайтов, дальнейшее запоминание множества разнообразных паролей не требуется.

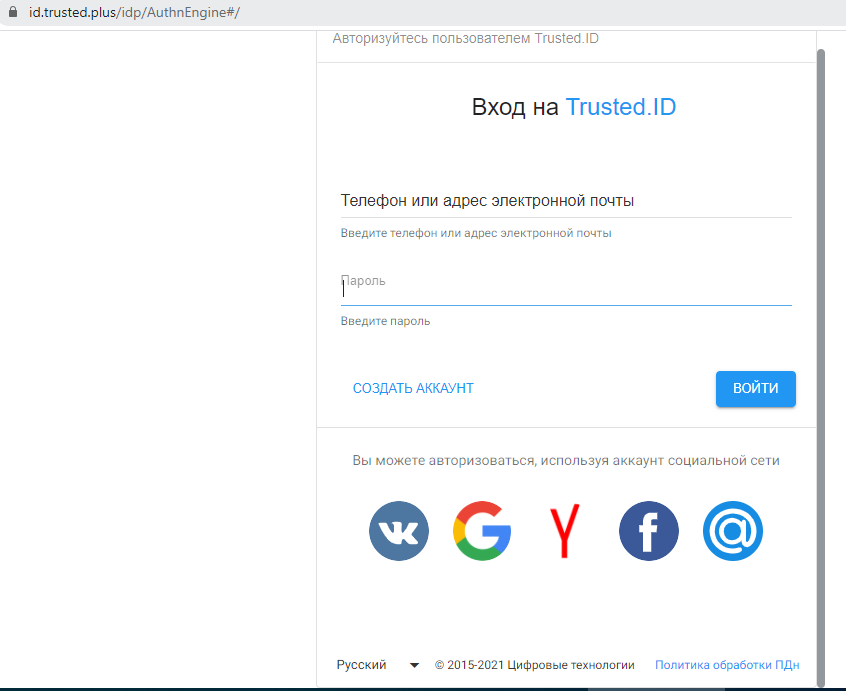


Рисунок 2 - окно входа в Trusted.Id

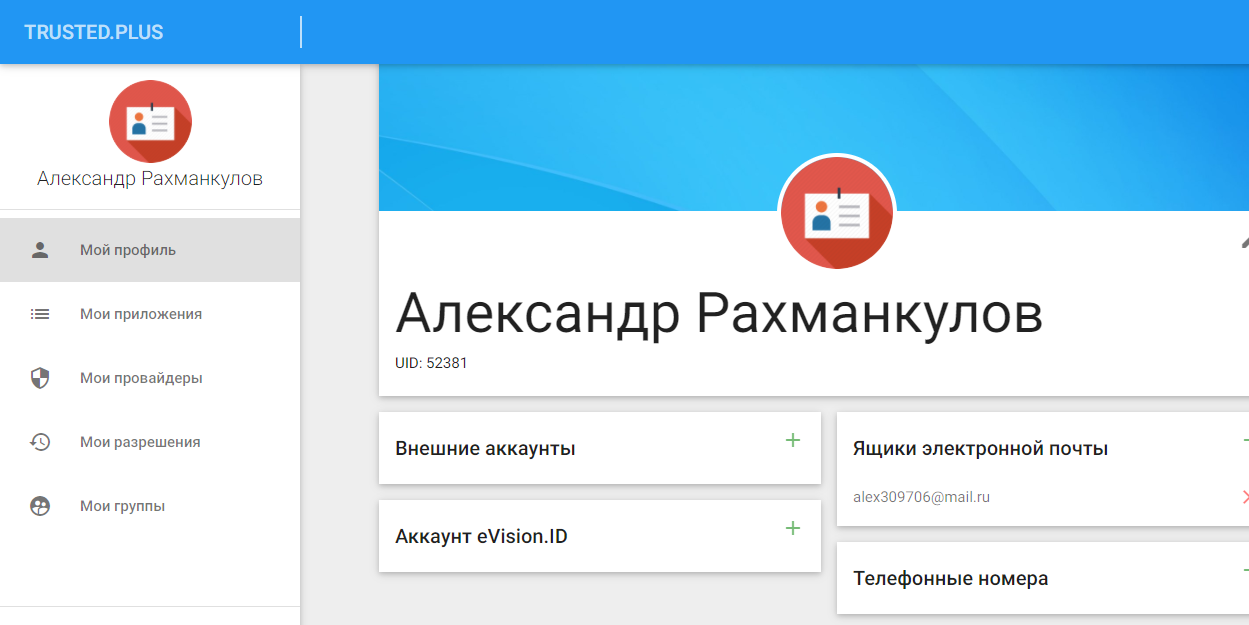


Рисунок 3 - окно настроек пользователя

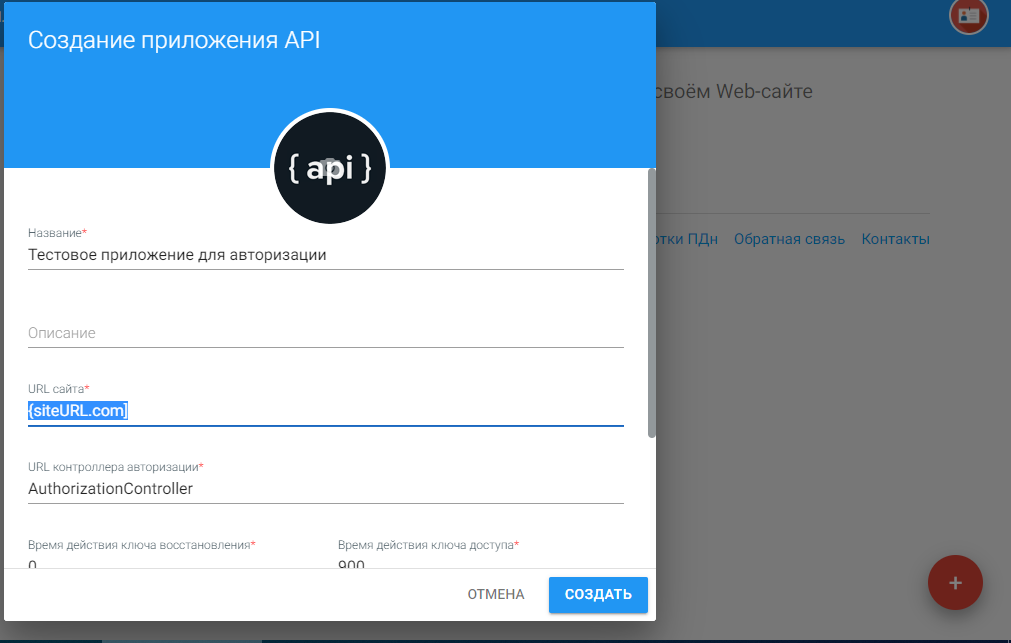


Рисунок - создание приложения для авторизации

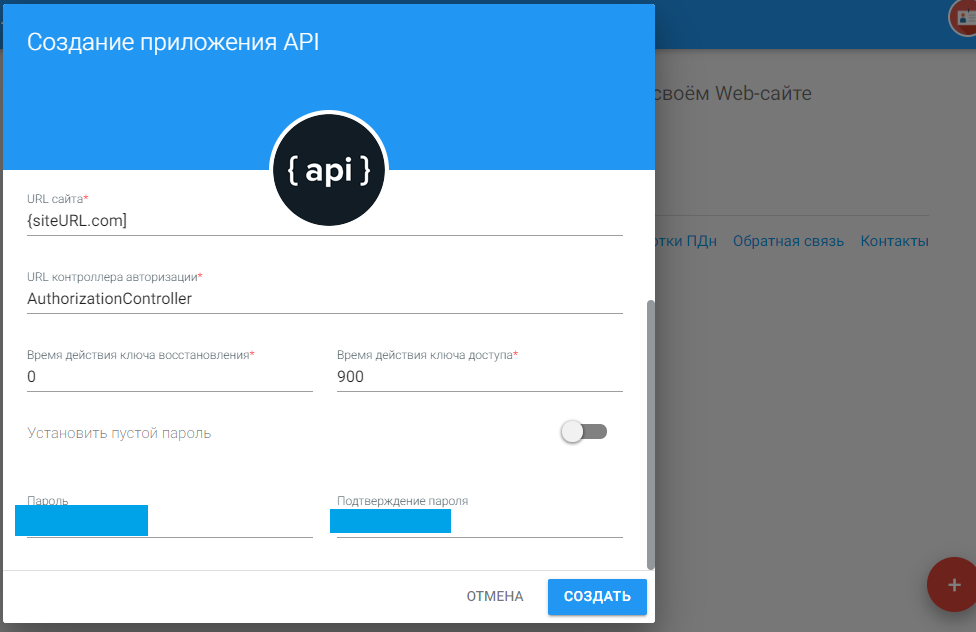


Рисунок - создание приложения для авторизации

#### Яндекс Паспорт

Яндекс.Паспорт служит для авторизации пользователей на сервисах Яндекса, а также на сторонних сервисах. В нем хранятся логин, имя и фамилия, привязанный номер телефона, платежные данные.

API Паспорта использует протокол OAuth — через него авторизованный пользователь Яндекса разрешает внешнему сайту или приложению доступ к данным своей учетной записи. С дополнительного разрешения Яндекс.Паспорт также вернет приложению некоторые личные данные: имя, пол, дату рождения.

Чтобы подключить авторизацию через Яндекс необходимо зарегистрировать приложение на OAuth-сервере, реализовать в приложении подходящий способ получения токена, использовать полученный токен в запросах к API сервисов Яндекса, добавить в интерфейс ресурса вход через Яндекс. Среди преимуществ решения — авторизация без ботов и уведомлений, на почту не приходит спам и уведомления о регистрациях, простое подключение.

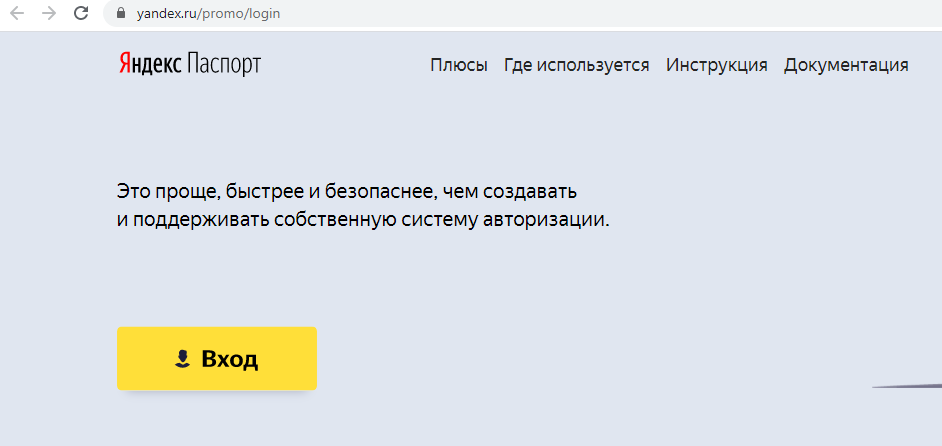


Рисунок 6 - главная страница

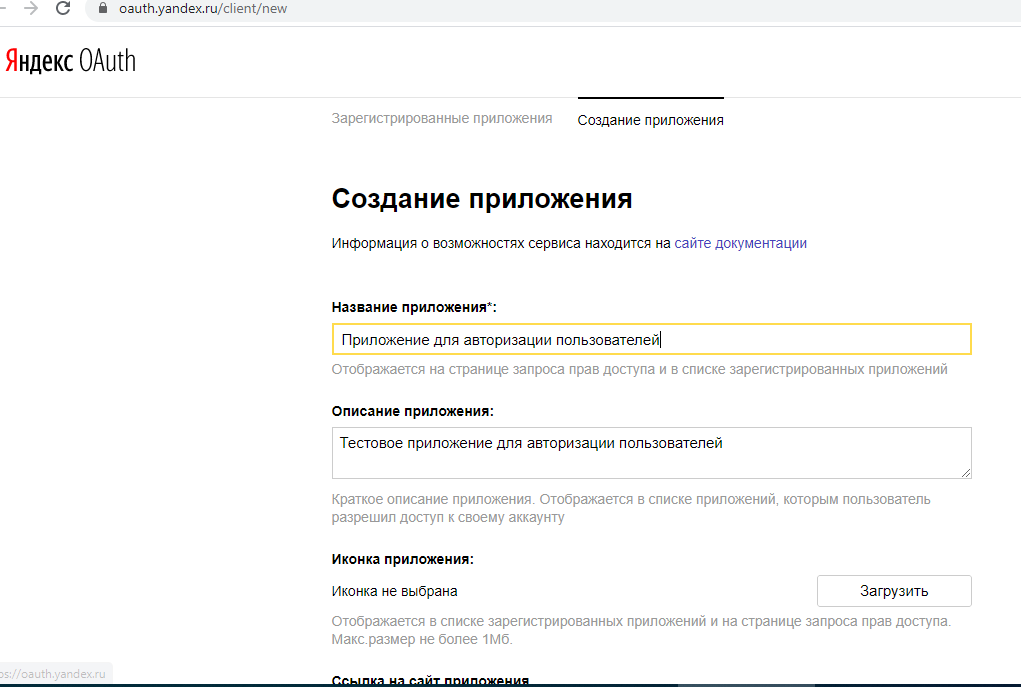


Рисунок 7 - создание приложения

Достоинства :

1. Простое подключение (не нужно программировать и верстать — подключение состоит из четырёх шагов)
2. Без ботов и ненужных уведомлений (почта не переполнится от спама и уведомлений о регистрациях)
3. Достаточно подробная документация

Недостатки:

1. Отсутствие контроля над процессом т.к. решение «из коробки»
2. Отсутствие добавления собственного функционала

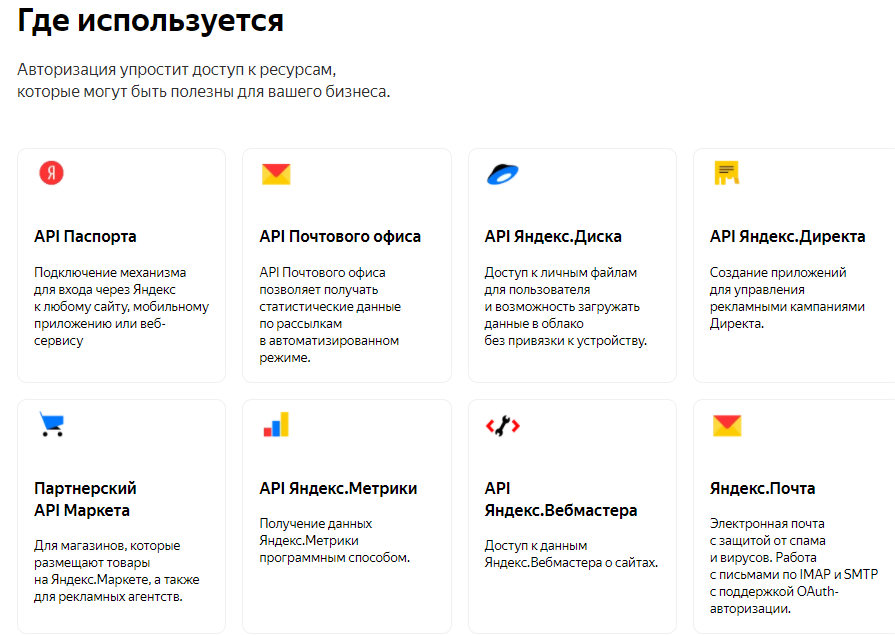


Рисунок 8 - список сервисов

### Программные реализации

#### ASP NET Identity

ASP.NET Identity представляет встроенную в ASP.NET систему аутентификации и авторизации. Данная система позволяет пользователям создавать учетные записи, аутентифицироваться, управлять учетными записями или использовать для входа на сайт учетные записи внешних провайдеров, таких как Facebook, Google, Microsoft, Twitter и других.

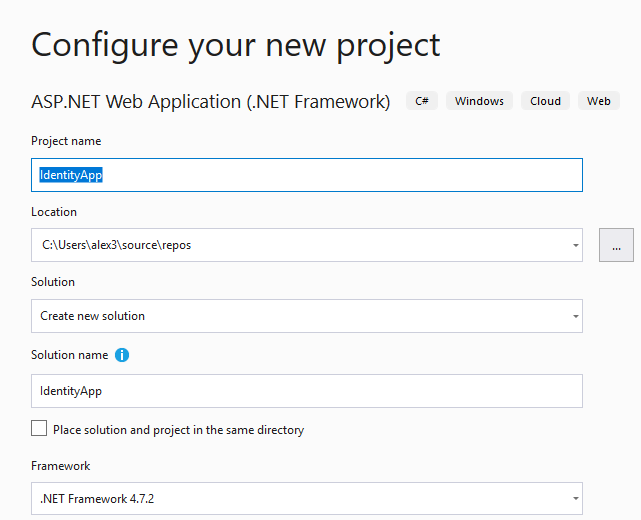


Рисунок - создание проекта

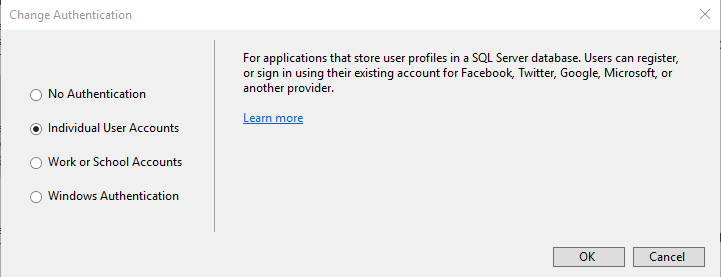


Рисунок - выбор типа аутентификации

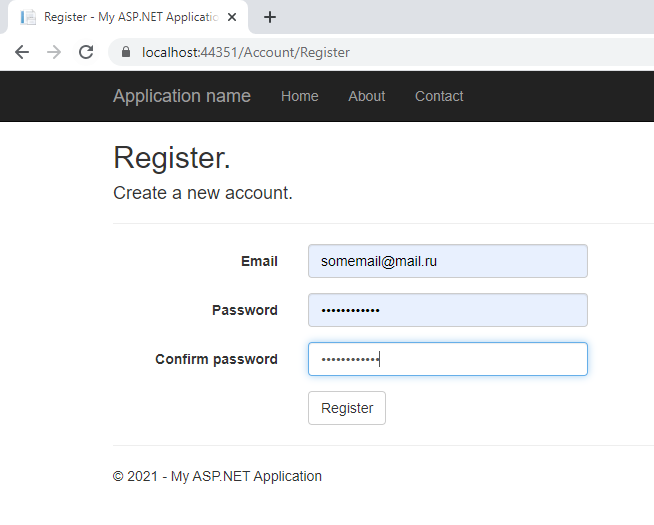


Рисунок - регистрация пользователя

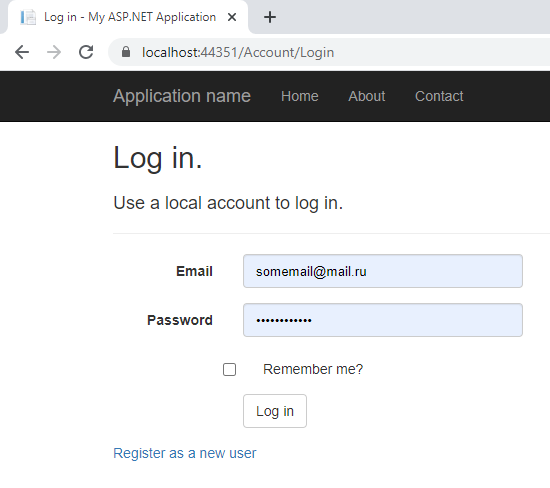


Рисунок - авторизация пользователя

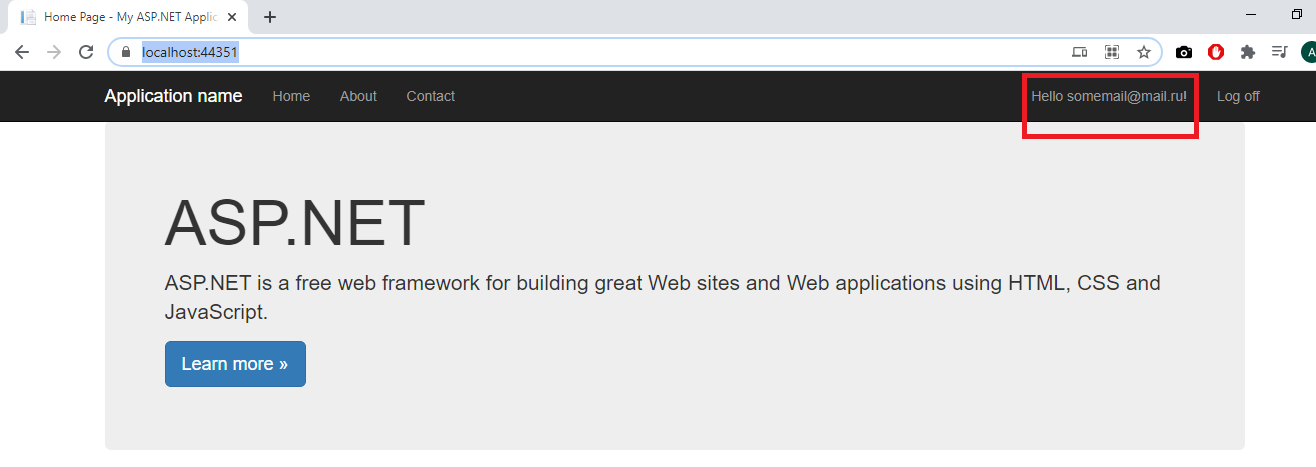


Рисунок - главная страница с приветствием авторизованного пользователя

Ключевыми объектами в AspNet Identity являются пользователи и роли.

Архитектура ASP NET Identity позволяет взять уже готовый функционал и при необходимости добавить новый, например, добавить для пользователя новое свойство или добавить новую таблицу в бд.

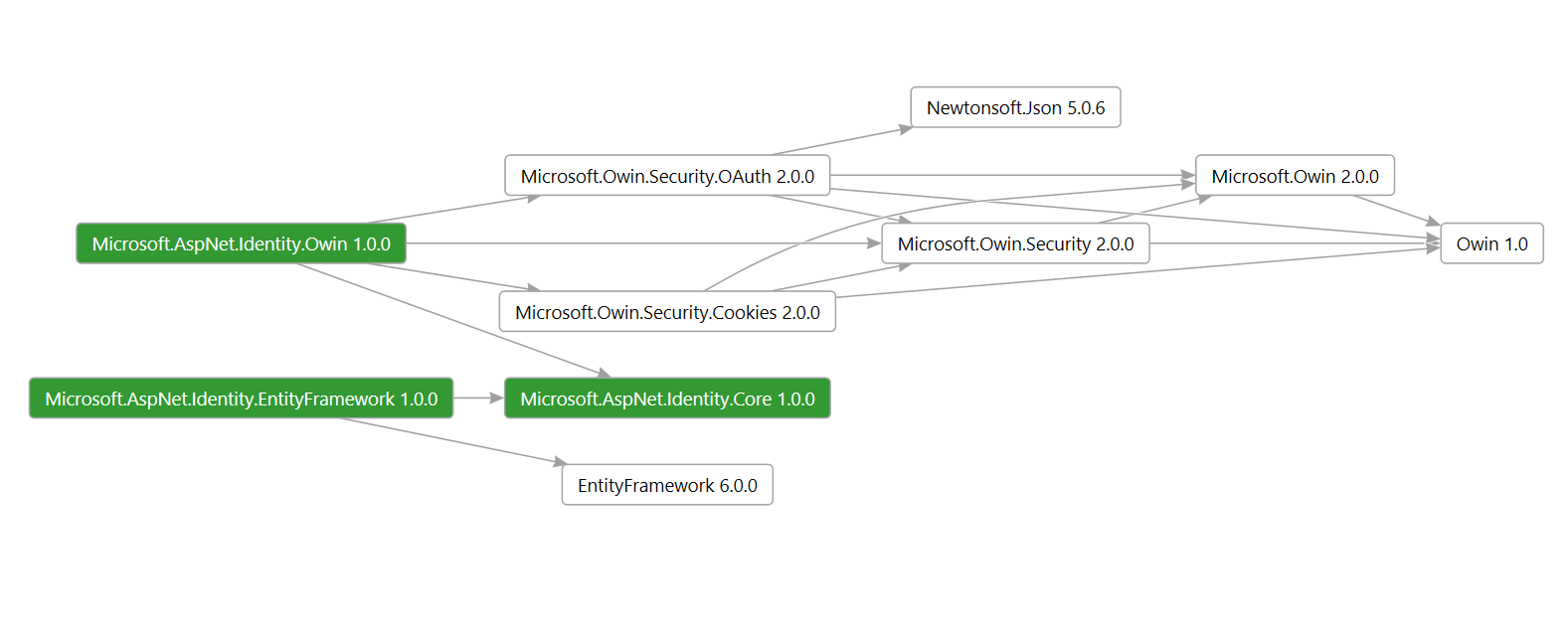


Рисунок – компоненты (зависимости) ASP NET Identity

Несмотря на то, что по умолчанию нам уже предоставляется готовый функционал, однако в ряде случаев, например, для отправки смс или электронной почты необходима дополнительная настройка.

Достоинства:

1. Простота использования
2. Возможность добавления собственного функционала
3. Подробная документация
4. Возможность управления всем процессом

Недостатки:

1. Продолжительное время освоения материала
2. Существует вероятность потери времени при изучении смежного материала
3. Вероятность потери времени при отладке приложения, т.к. инструмент не «из коробки»

## Обоснование выбора инструментов и платформы для разработки

### С# и .Net

На сегодняшний момент язык программирования C# один из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. В настоящий момент на нем пишутся самые различные приложения: от небольших десктопных программок до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей.

#### История развития языка

Язык программирования С# был разработан в 1993—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft.



Рисунок 15 – Андрес Хейлсберг

К 2000 году у Microsoft были готовы промышленные версии новых технологий и решений для обмена сообщениями и данными, а также для создания Internet-приложений. Была выпущена и новая платформа для разработки под новые решения — .NET. В ней объединились сразу несколько языков программирования, что было в новинку для того времени.

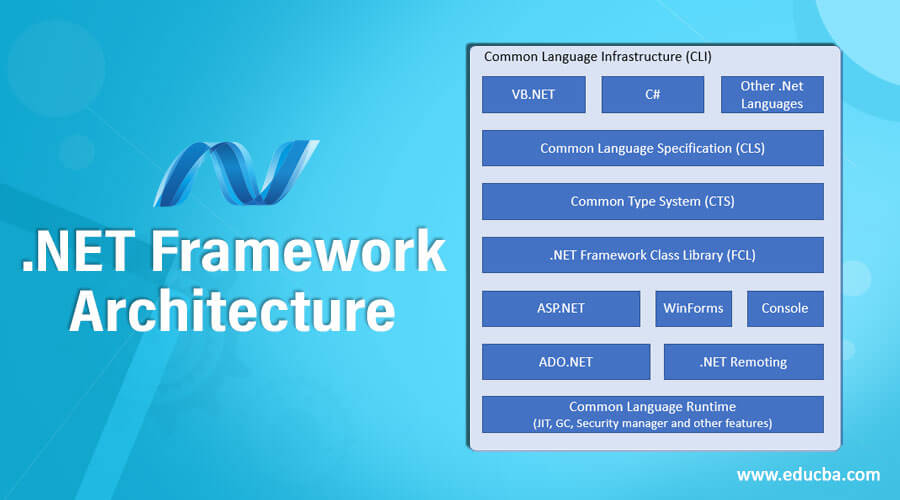


Рисунок 16 – архитектура .Net Framework

Ещё одним новшеством платформы .NET была технология активных серверных страниц ASP.NET (Active Server Page). С её помощью можно было относительно быстро разработать веб-приложения, взаимодействующие с базами данных. Специально для ASP.NET был создан язык программирования C#. Да и сама ASP.NET была полностью написана на нём.

По сравнению с другими языками C# достаточно молодой, но в то же время он уже прошел большой путь. Первая версия языка вышла вместе с релизом Microsoft Visual Studio .NET в феврале 2002 года. Текущей версией языка является версия C# 8.0, которая вышла в сентябре 2019 года вместе с релизом .NET Core 3.

C# — это язык с C-подобным синтаксисом. Здесь он близок в этом отношении к C++ и Java.

Будучи объектно-ориентированным языком, он много перенял у Java и С++. Как и Java, C# изначально предназначался для веб-разработки, и примерно 75% его синтаксических возможностей такие же, как у  Java. C# также называют «очищенной версией Java». Ещё 10% наш герой позаимствовал из C++ и 5% – из Visual Basic. Оставшиеся 10% C# — это реализация собственных идей разработчиков. Объектно-ориентированный подход позволяет строить с помощью C# крупные, но в то же время гибкие, масштабируемые и расширяемые приложения.

C# уже давно поддерживает много полезных функций:

* инкапсуляция,
* наследование,
* полиморфизм,
* перегрузка операторов,
* статическая типизация.

При этом он всё ещё активно развивается, и с каждой новой версией появляется всё больше интересного — например лямбды, динамическое связывание, асинхронные методы и т.д.

#### Роль платформы .NET

Когда говорят C#, нередко имеют в виду технологии платформы .NET (Windows Forms, WPF, ASP.NET, Xamarin). И наоборот, когда говорят .NET, нередко имеют в виду C#. Однако, хотя эти понятия связаны, отождествлять их неверно. Язык C# был создан специально для работы с фреймворком .NET, однако само понятие .NET несколько шире.



Рисунок 17– технологии .Net Framework

Когда-то Билл Гейтс сказал, что .NET — это лучшее, что сделала компания Microsoft. У него есть весомые основания так считать. Фреймворк .NET представляет мощную платформу для создания приложений. Вот его киллер-фичи:

* Поддержка нескольких языков. В основе  .NET — общеязыковая среда исполнения Common Language Runtime (CLR), благодаря чему платформа поддерживает несколько языков: наряду с C# это VB.NET, C++, F#, а также различные диалекты других языков, привязанные к .NET, например, Delphi.NET. Код на любом из этих языков компилируется в сборку на общем языке CIL (Common Intermediate Language) — своего рода ассемблер платформы .NET. Поэтому можно сделать отдельные модули одного приложения на разных языках.
* Мощная библиотека классов. .NET представляет единую для всех поддерживаемых языков библиотеку классов. Какое бы приложение мы ни собирались писать на C# — текстовый редактор, чат или сложный веб-сайт — так или иначе мы задействуем библиотеку классов .NET.
* Разнообразие технологий. Общеязыковая среда исполнения CLR и базовая библиотека классов — это основа для целого стека технологий, которые разработчики могут задействовать при создании разных приложений. Например, для баз данных в этом стеке имеется технология ADO.NET и Entity Framework Core. Для графических приложений с насыщенным интерфейсом — технологии WPF и UWP. Для более простых графических приложений — Windows Forms. Для разработки мобильных приложений — Xamarin. Для создания веб-сайтов - ASP.NET и т.д.

#### Преимущества и недостатки языка C#

У «шарпа» выделяют много преимуществ:

* Поддержка подавляющего большинства продуктов Microsoft
* Бесплатность ряда инструментов для небольших компаний и некоторых индивидуальных разработчиков — Visual Studio, облако Azure, Windows Server, Parallels Desktop для Mac Pro и др.
* Типы данных имеют фиксированный размер (32-битный int и 64-битный long), что повышает «мобильность» языка и упрощает программирование, так как вы всегда знаете точно, с чем вы имеете дело.
* Автоматическая «сборка мусора». Это значит, что нам в большинстве случаев не придётся заботиться об освобождении памяти. Вышеупомянутая общеязыковая среда CLR сама вызовет сборщик мусора и очистит память.
* Большое количество «синтаксического «сахара» — специальных конструкций, разработанных для понимания и написания кода. Они не имеют значения при компиляции.
* Низкий порог вхождения. Синтаксис C# имеет много схожего с другими языками программирования, благодаря чему облегчается переход для программистов. Язык C# часто признают наиболее понятным и подходящим для новичков.

Но есть у C# и некоторые недостатки:

* Приоритетная ориентированность на платформу Windows;
* Язык бесплатен только для небольших фирм, индивидуальных программистов, стартапов и учащихся. Крупной компании покупка лицензионной версии этого языка обойдётся в круглую сумму.

Источники литературы

<https://xn--80adfdbscmorebdjpezh9nvd.xn--p1ai/shop/product/postanovka-zadachi-na-proektirovaniepostanovka-zadachi-na-proektirovanie/>

<https://habr.com/ru/company/luxoft/blog/312188/>

<http://plantuml.com/guide>

https://plantuml-editor.kkeisuke.com/

Приложение 1. Листинг сценариев использования на языке plantUML

@startuml

left to right direction

skinparam actorStyle awesome

package ДействияНезарегистрированногоПользователя{

usecase "Просмотр контента" as contentView

usecase "Регистрация" as registration

}

:Незарегистрированный пользователь:--(contentView)

:Незарегистрированный пользователь:--(registration)

package ДействияЗагеристрированногоПользователя{

usecase "Авторизация" as authorization

usecase "Управление учетной записью" as accountManagement

usecase "Добавление контента" as contentAddition

}

:Зарегистрированный пользователь :-- authorization

:Зарегистрированный пользователь :-- accountManagement

:Зарегистрированный пользователь :-- contentAddition

:Зарегистрированный пользователь :-- contentView

package ДействияАдминистратора{

usecase "Проверка достоверности информации" as checkAddedContent

usecase "Управление контентом" as contentManagement

usecase "Выдача прав администратора" as roleManaging

}

authorization -- Администратор

contentView -- Администратор

checkAddedContent -- Администратор

contentManagement -- Администратор

accountManagement -- Администратор

contentAddition -- Администратор

roleManaging --Администратор

@enduml