|  |  |
| --- | --- |
|  | **Правительство Санкт-Петербурга Комитет по образованию Санкт-Петербургское государственное бюджетное профессиональное образовательное учреждение «Радиотехнический колледж»** |

|  |
| --- |
|  |

**КУРСОВАЯ РАБОТА (ПРОЕКТ)**Тема: \_\_\_\_проектирование и разработка сайта для ведения блогов\_\_\_  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
по профессиональному модулю (МДК, учебной дисциплине) \_\_\_\_\_\_\_  
\_\_\_МДК 05.02 Разработка кода информационных систем\_\_\_\_\_\_  
Специальность: Разработчик веб и мультимедийных приложений код: 09.02.07 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |
| --- |
| Выполнил(а):  студент(ка) \_\_\_\_\_2\_\_\_\_\_ курса  гр. №\_\_291\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_Новоселов Артём Сергеевич\_  (Ф.И.О.) Проверил:  руководитель: преподаватель  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Ф.И.О.) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Курсовая работа (проект) защищена  
 «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_\_г.  
 Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург 2020г.

[ВВЕДЕНИЕ 3](#_Toc1)

[ГЛАВА 1. ПРОЕКТНОЕ ОБСЛЕДОВАНИЕ 5](#_Toc2)

[1.2 ОБЗОР АНАЛОГОВ 6](#_Toc3)

[1.2.1 САЙТ ДЛЯ ВЕДЕНИЯ БЛОГОВ GHOST.ORG 6](#_Toc4)

[1.2.2 САЙТ ДЛЯ ВЕДЕНИЯ БЛОГОВ DEV.TO 7](#_Toc5)

[1.2.3 САЙТ ДЛЯ ВЕДЕНИЯ БЛОГОВ TUMBLR.COM 8](#_Toc6)

[1.2.4 СРАВНИТЕЛЬНЫЙ АНАЛИЗ 9](#_Toc7)

[2.1 ОПИСАНИЕ ТЕХНИЧЕСКИХ РЕШЕНИЙ ПРОЕКТА 12](#_Toc8)

[2.2 ОПИСАНИЕ РАЗРАБОТКИ ПРОЕКТА 12](#_Toc9)

[2.2.1 РАЗРАБОТКА БАЗЫ ДАННЫХ 13](#_Toc10)

[2.2.2 РАЗРАБОТКА WEB API 13](#_Toc11)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 14](#_Toc12)

**ВВЕДЕНИЕ**

В современном мире многие привычные отрасли начали миграцию в веб-пространство. Следствием этого стало такое понятие, как блоггинг. Сейчас он является альтернативой журналистике и писательству. **Актуальность темы моей курсовой работы** как раз обусловлена потребностью пользователей сети Интернет в надёжном и гибком сервисе, позволяющем вести свой собственный блог и читать чужие.

Таковая потребность возникает не только у людей, знакомых с технологиями, но и у тех, кто общается с ними на «Вы». К сожалению, не все современные системы могут угодить своим функционалом и тем, и другим.

Именно это я и поставил **целью** своей курсовой работы: разработать сервис, который в одинаковой мере удовлетворил оба вида пользователей. Для тех, кто мало знаком с современными информационными технологиями, система должна быть простой и понятной в использовании. Для более продвинутых пользователей сервис должен предоставить возможность полного её контроля и расширения в соответствии с их потребностями.

Для достижения данной цели мне необходимо решить следующие **задачи**:

1. проанализировать рынок доступных решений;
2. сделать выбор оптимальной технологии для разработки решения;
3. написание кода и запуск решения;
4. тестирование решения.

**Структура курсовой работы** следует из логики изложения основного  
материала. Таким образом, можно выделить следующие части: введение, основная часть, состоящая из 5-ти глав, заключение, список терминов, список использованных источников и литературы.

**Новизна**. В Интернете можно найти множество платформ для ведения блогов. Они отличаются между собой языком реализации, шириной функционала, возможностями расширения. Подробное рассмотрение некоторых популярных из них будет представлено в следующей главе курсовой работы. Сейчас я хотел бы остановиться на том, почему данное решение можно считать принципиально новым и отличным от аналогов.

Представленное в данной курсовой работе решение можно назвать многослойным: каждый последующий модуль опирается на предыдущие, но не зависит от последующих. Такая структура позволит обеспечить необходимую пользователю гибкость системы. Он сможет не только вносить изменения в модуль, но и целиком его заменить собственным.

Основной модуль представлен в качестве Web API, который выступает в качестве Headless CMS. Пользователь системы сможет даже построить собственное решение на его основе.

Помимо API решение имеет клиент с графическим интерфейсом, который позволит работать с сервисом и тем, кто мало знаком с IT-технологиями.

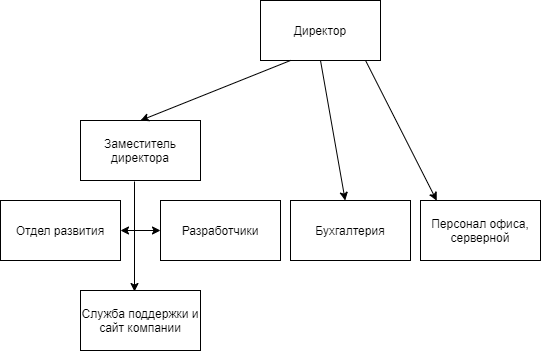
ГЛАВА 1. ПРОЕКТНОЕ ОБСЛЕДОВАНИЕ

**1.1 ОПИСАНИЕ** ПРЕДМЕТНОЙ **ОБЛАСТИ**

Информационная система, представленная в данном проекте является достаточно абстрактным явлением. Область её применения достаточно велика для существования «самой по себе». Сервис для создания и ведения блогов может быть использован во множестве ситуаций. Ввиду этого, для наглядности причин использования информационной системы, рассматриваемой в данной работе, приведён пример её внедрения в гипотетической компании.

Назовём нашу организацию «Облако». Она занимается предоставлением услуг облачного хостинга решений для Web. Во главе компании стоит директор. Его заместитель занимается вопросами работы с партнёрами, управлением отделами разработки и развития. Все продажи происходят автоматически: у компании уже есть сайт, на котором клиенты могут ознакомиться с услугами компании, сделать выбор необходимых услуг, совершить оплату. Служба поддержки клиентов оказывает помощь покупателям в случае возникновения у них вопросов и/или проблем. Так же в структуру можно включить бухгалтерию и персонал офиса, серверной.

Таким образом, получается следующая структура:



На сайте компании есть документация, в которой хранятся наиболее актуальные данные о платформе, необходимые клиенту для успешного и наиболее продуктивного её использования. Документация, к сожалению, не может выполнять функцию оповещения пользователей. В ней сложно отразить последние изменения кратко и обобщённо.

Такую проблему решает блог компании. Здесь организация сможет вести общение с клиентами, имея возможность уведомлять их об изменениях и обновлениях в работе сервиса, проводить краткие уроки, ненавязчиво рассказывать о преимуществах своего решения.

**1.2 ОБЗОР** АНАЛОГОВ

**1.2.1 САЙТ ДЛЯ ВЕДЕНИЯ БЛОГОВ GHOST.ORG**

Ghost.org можно без сомнений назвать решением «всё в одном». Сервис предоставляет полноценное веб-приложение для создания, ведения и управления блогом с возможностью расширения и подключения сторонних решений. Если быть точнее, они разработчики позаботились о создании целого ряда решений.

Одним из них является Headless CMS, предоставляемая разработчикам как self-hosting платформа для управления контентом. Код является открытым: любой желающий может использовать его для разработки собственных решений на его основе.

Вторым можно считать упрощенную версию для малознакомых с IT-технологиями. Разработчики Ghost.org решают за конечного пользователя вопросы хостинга блога, взаимодействия с кодом (в данном случае оно не требуется: платформа представляет собой веб-приложение с многофункциональным интерфейсом), хранения данных. В таком случае доступ к блогу осуществляется через сервера Ghost.org. На них же хранится информация. Данное решение имеет существенный минус для пользователя - плата за аккаунт.

Преимущества платформы:

* простота для пользователя;
* ряд решений, позволяющий выбрать в точности подходящее в ситуации;
* множество возможностей для оформления.

Недостатки платформы:

* платформа разработана на node.js, который уступает по скорости и производительности некоторым другим технологиям (согласно https://dotnet.microsoft.com/apps/aspnet ASP.NET Core обеспечивает в разы большую производительность);
* за использование хостинга Ghost.org предусмотрена плата по подписке.

Ghost.org - прекрасное решение, которым угодит многим пользователям, но когда встаёт вопрос производительности, оно сильно уступает более продвинутым технологиям.

1.2.2 САЙТ ДЛЯ ВЕДЕНИЯ БЛОГОВ DEV.TO

Dev.to - веб-приложение, которое может быть использовано для ведения блога. Их идея - создание сообщества, поэтому структура сайта больше напоминает социальную сеть. Сервис позволяет читать блоги интересных вам авторов, следить за их активностью на сайте, комментировать записи, создать и вести собственный блог.

Серверная часть сервиса находится в открытом доступе и может быть использована в создании других решений на её основе. Однако разработчики предупреждают о том, что работа над ней пока не закончена, и не гарантируют, что решение будет одинаково хорошо функционировать в разных условиях.

Преимущества:

* простой интерфейс, привычный для пользователей социальных сетей;
* сервис предоставляет услуги бесплатно;
* не требуются особые навыки для использования.

Недостатки:

* почти нет возможностей к персонализации;
* платформа представляет из себя в большей степени агрегатор блогов, то есть присутствует конкуренция за читателя;
* платформа нерасширяема: нельзя подключить сторонние приложения.

Из выявленных преимуществ и недостатков сервиса можно сделать следующий вывод. Платформа более всего подходит для создания не просто блога, но сообщества единомышленников. Простота сервиса сильно выделяет его на фоне конкурентов для пользователей, кто мало знаком или совсем не знаком с разработкой, но оставляет невозможным расширение возможностей.

1.2.3 САЙТ ДЛЯ ВЕДЕНИЯ БЛОГОВ TUMBLR.COM

Данная платформа выступает, в первую очередь, как социальная сеть. Пользователь платформы, зарегистрированный в сети, может создать необходимое ему количество страниц, которые выступают в качестве хранилища для его постов.

Разработчики позаботились о возможности создания пользователем собственного дизайна для страницы, хостинга на платформе, смены доменного имени. Решение может существовать как в качестве части социальной сети, так и отдельно от неё.

Преимущества:

* привычность платформы, достигаема за счёт её схожести с социальной сетью;
* возможность персонализации блога;
* имеются мобильные и веб-версии приложения;
* большой ряд файлов, доступных для прикрепления к постам;
* услуги, предоставляемые сервисом, бесплатны.

Недостатки:

* многие пользователи отмечают сложность платформы для новичков;
* код сервиса закрыт;
* общедоступность платформы создаёт проблему разносторонности пользователей - слишком большая целевая аудитория;
* нет возможности подключить сторонние приложения.

Из вышесказанного следует, что платформа является наиболее популярной и простой для пользователя. К сожалению, именно это и создаёт некоторые проблемы с ней. За счёт простоты решения возникает ограничение её расширяемости, а популярность сильно повышает уровень конкуренции внутри платформы.

1.2.4 СРАВНИТЕЛЬНЫЙ АНАЛИЗ

Все приведённые в данной статье технологии можно несомненно считать самостоятельными и достойными решениями. Но у каждого есть недостатки и преимущества. Для наибольшей объективности в оценке сервисов, выделим основные критерии. Самыми важными можно назвать простоту использования, расширяемость, дизайн, возможность изменения дизайна, потенциал к изменению (возможность построения принципиально новой информационной системы на основе этой), производительность, скорость работы. Оценка будет производиться по пятибальной шкале, где «1» - функция реализована плохо, а «5» - аспект реализован в полной мере и удовлетворяет потребность. Для тех пунктов, которые в системе не представлены, будет использована оценка «0».

|  |  |  |  |
| --- | --- | --- | --- |
| Критерий | ghost.org | dev.to | tumblr.com |
| Простота использования | 4 | 5 | 4 |
| Расширяемость | 5 | 3 | 3 |
| Дизайн | 5 | 4 | 4 |
| Возможность изменения дизайна | 5 | 2 | 5 |
| Потенциал к изменению | 5 | 0 | 0 |
| Производительность | 3 | 3 | 3 |
| Скорость | 4 | 3 | 3 |

В ходе сравнения прояснилось, что лидером среди аналогов выступает ghost.org. Это достигается за счёт предоставления пользователю целого ряда решений.

1.3 РАЗРАБОТКА СИСТЕМНОГО И ТЕХНИЧЕСКОГО ПРОЕКТОВ

Если исходить из анализа аналогов, решение должно быть модульным, гибким и иметь потенциал к полной кастомизации. Такие требования ведут к тому, что архитектура приложения должна быть многослойной. Таким образом, для пользователя будут обеспечены возможности как к использованию готового решения, так и к созданию своего на основе имеющегося.

Ввиду вышесказанного, приложение может иметь следующую архитектуру в упрощённом виде:



Доступ к базе данных осуществим через API и общедоступный клиент. Второй так же имеет доступ к API, как и пользователи, которые смогут создать на базе API собственные клиентские приложения или же сменить базу данных.

ГЛАВА 2. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1 ОПИСАНИЕ ТЕХНИЧЕСКИХ РЕШЕНИЙ ПРОЕКТА

Языком программирования для реализации этого приложения был выбран C#. Согласно сайту Майкрософт, C# - это простой современный объектно-ориентированный язык программирования со статической типизацией. Данный ЯП достаточно прост в изучение (имеется ввиду базовый синтаксис), а на его основе построены решения для множества технологий. Одно из таких - платформа .NET, в число возможностей которой входит огромный функционал. В него включён фреймворк для разработки веб-приложений ASP.NET Core, который отличается гибкостью (на его основе можно создавать как MVC решения, так и Web API), относительной простотой (существует чёткая структура проекта, что сильно упрощает разработку) и производительностью.

В вопросе базы данных выбор пал на NoSQL БД MongoDB. Основным критерием её выбора была простота работы с ней. Данные в БД хранятся в форме отдельных BSON-документов (расширенный JSON). Такое решение сильно упрощает работу с данными. Делает её быстрой и простой, сохраняя при этом высокую производительность.

Для создания клиента приложения используется Blazor WebAssembly. Данная технология позволяет разрабатывать браузерную часть решения на языке C#, оставляя возможность взаимодействия с JavaScript-кодом. При этом отмечается прирост в производительности в сравнении с более привычными технологиями браузерной разработки. К сожалению, пока данная технология поддерживается не всеми браузерами, но огромная аудитория (подавляющее большинство) поддерживаемых позволяет пойти на такую жертву.

Bootstrap 4 позволит избежать прямой работы со стилями во время разработки клиентского приложения. Данное решение позволит сэкономить время и усилия, затраченные на разработку.

2.2 ОПИСАНИЕ РАЗРАБОТКИ ПРОЕКТА

**2.2.1 РАЗРАБОТКА БАЗЫ ДАННЫХ**

Для ускорения разарботки приложения на первых этапах была выбрана упрощённая модель базы данных. Её схематическое представление:

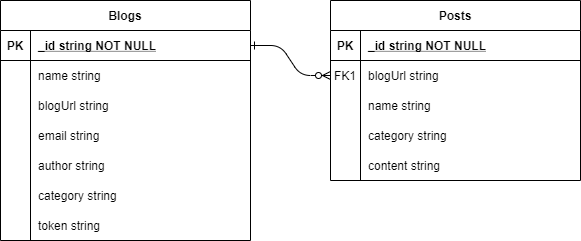


Таблица «Blogs» содержит в себе записи блогов со следующими данными: идентификатор, имя, URL-идентификатор (blogUrl), адрес электронной почты автора, имя автора, категория контента блога, индивидуальный токен, обеспечивающий доступ к блогу.

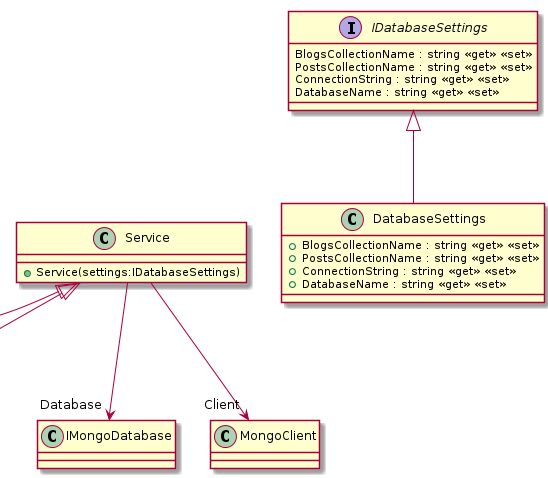
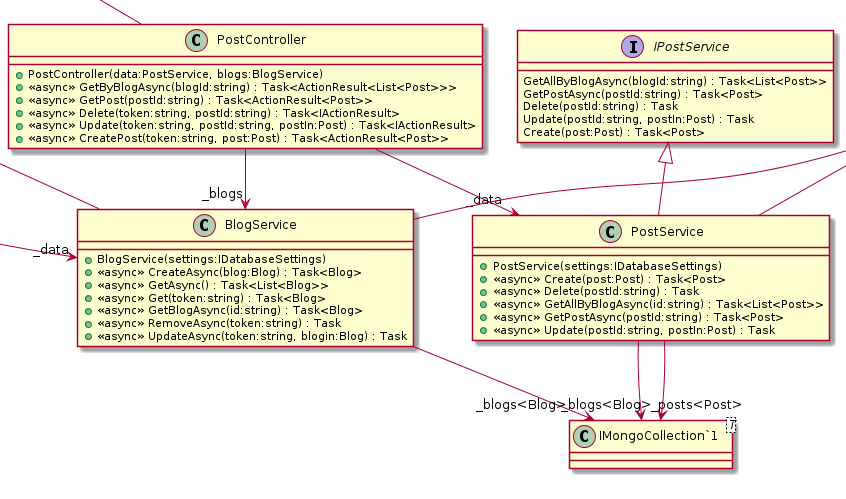
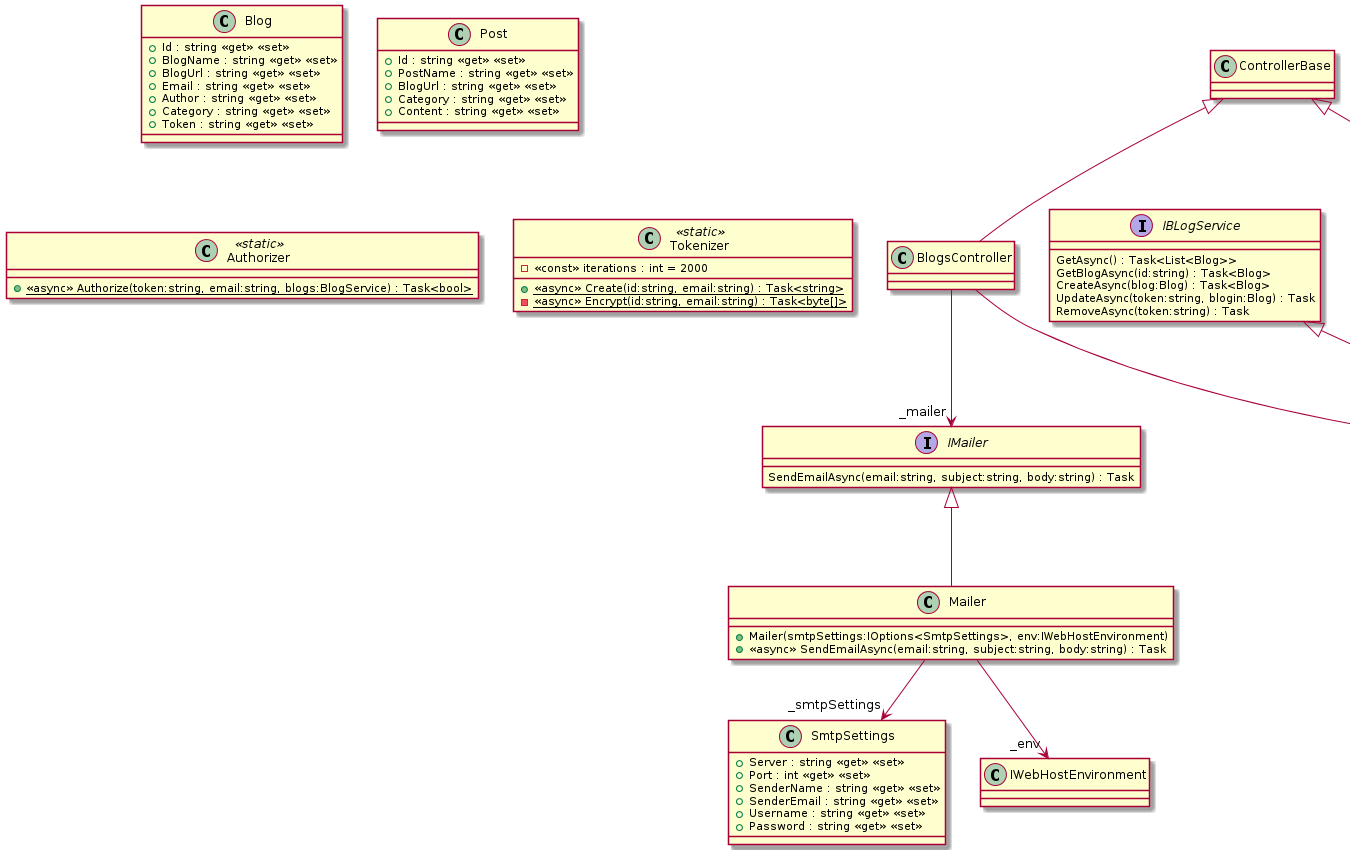
Таблица «Posts» содержит в себе записи постов блогов со следующими данными:

индентификатор, название, URL-идентификатор блога, категория содержимого, и, собственно, само внутренее содержимое.

Такая структура базы данных достаточно примитивна, но и она позволит создать полноценное работоспособное приложение. Упрощение следует из выделенного ранее принципа расширяемости - ядро приложения должно быть простым, чтобы предоставить в дальнейшем потенциал к модификациям.

**2.2.2 РАЗРАБОТКА WEB API**

Ниже вы можете видеть кодовую структуру приложения представленную в виде UML-диаграммы классов.



Классы «Blog» и «Post» представляют из себя модели для работы с соответствующими элементами базы данных. Их поля соответствуют полям объектов БД.

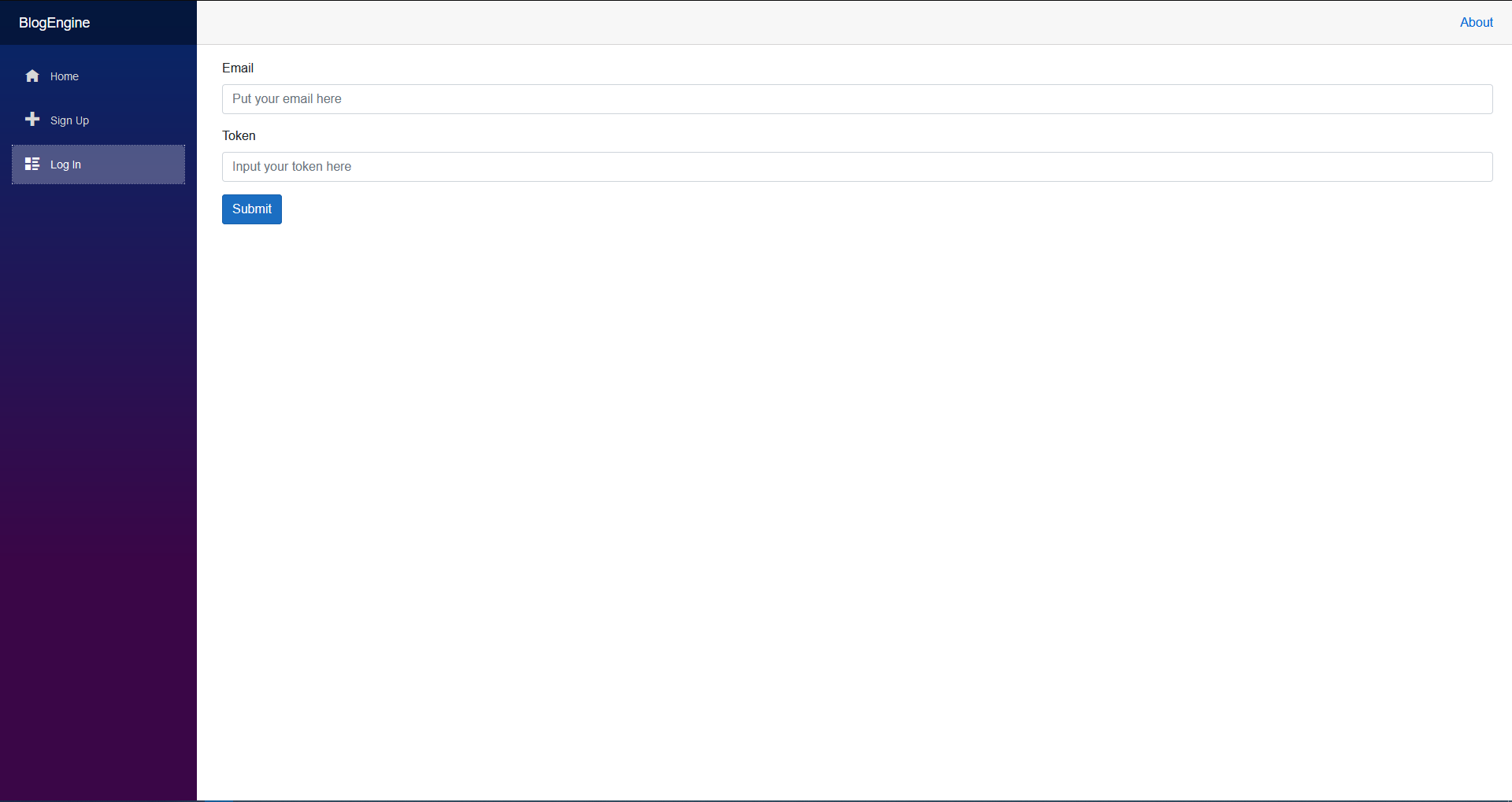
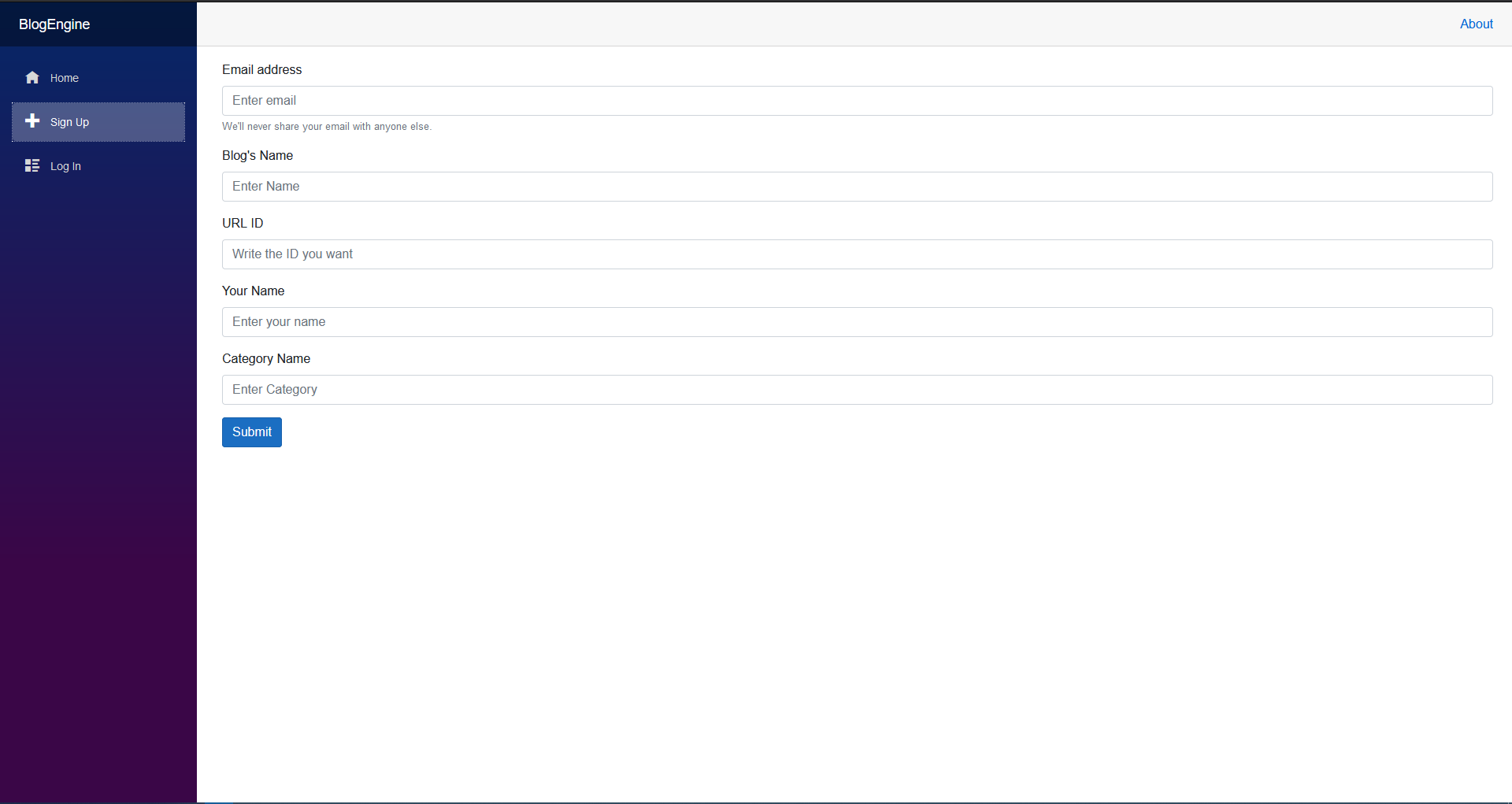
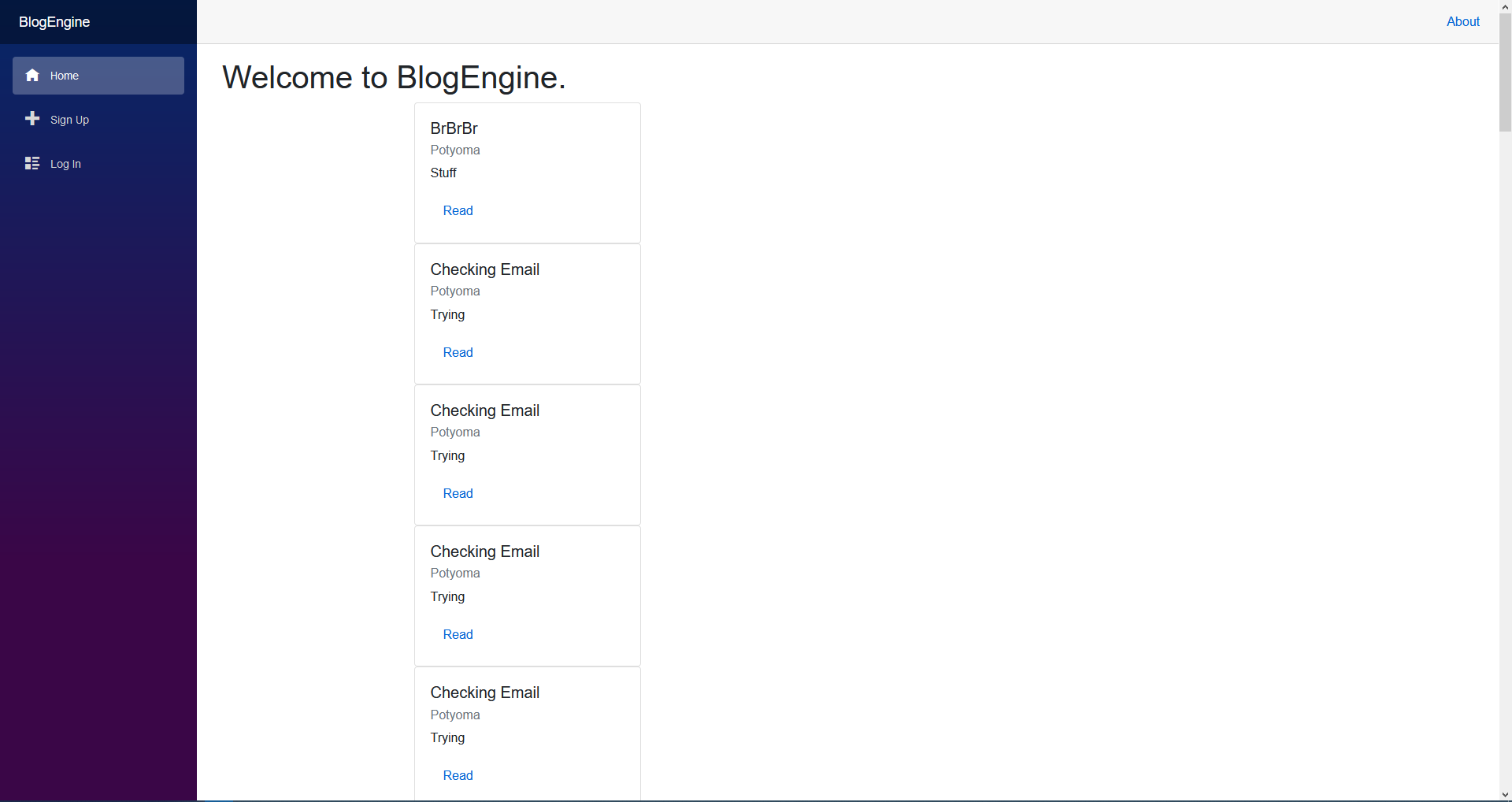
Контроллеры позволяют управлять ответами на HTTP-запросы: предоставлять данные или же управлять ими в БД.

Пространство имён «Utilities» содержит в себе классы для дополнительной манипуляции данными внутри приложения.

Рассмотрим, как код работает на примере процедуры регистрации блога. Контроллер получает данные пользователя POST-запросом от клиента. Далее он обращается к «BlogService», который создаёт новый объект в базе данных. Токен для объекта генерируется классом «Tokenizer». Алгоритм представлен в коде проекта (см. Приложение 1). После чего «BlogService» вызывает класс «Mailer» для создания электронного письма и отправки его на указанный адрес. После данных процедур контроллер отправляет клиенту новый блог-объект в качестве ответа.

2.2.3 РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ

Наибольшее испытание в разработке клиента для приложение - проектирование простого и понятного пользовательского интерфейса. Далее представлены получившиеся в ходе разработки прототипы пользовательского интерфейса.



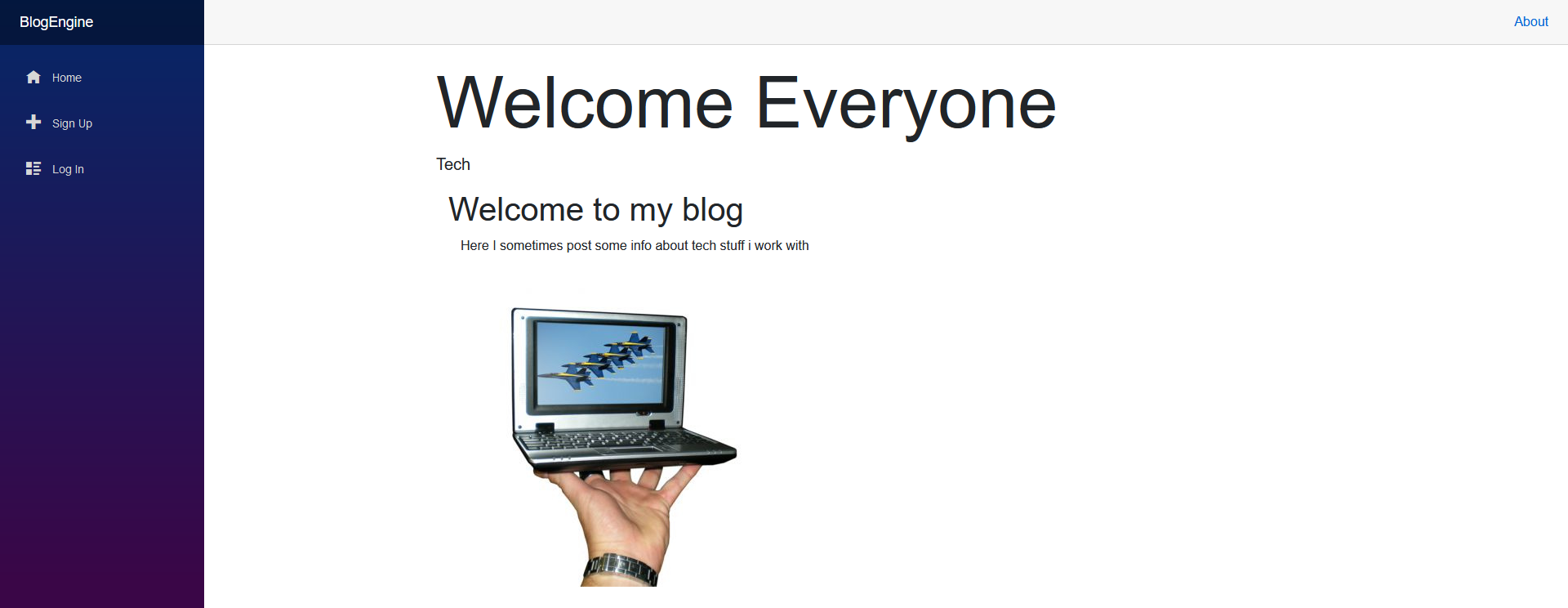
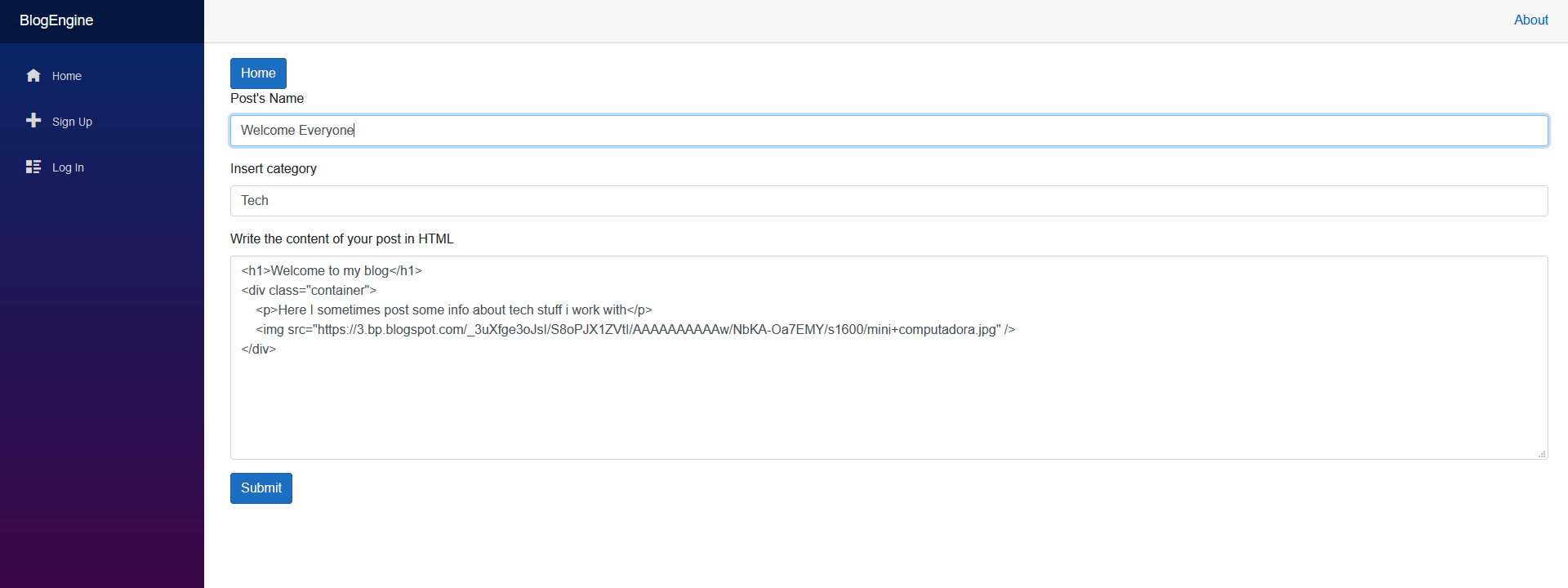
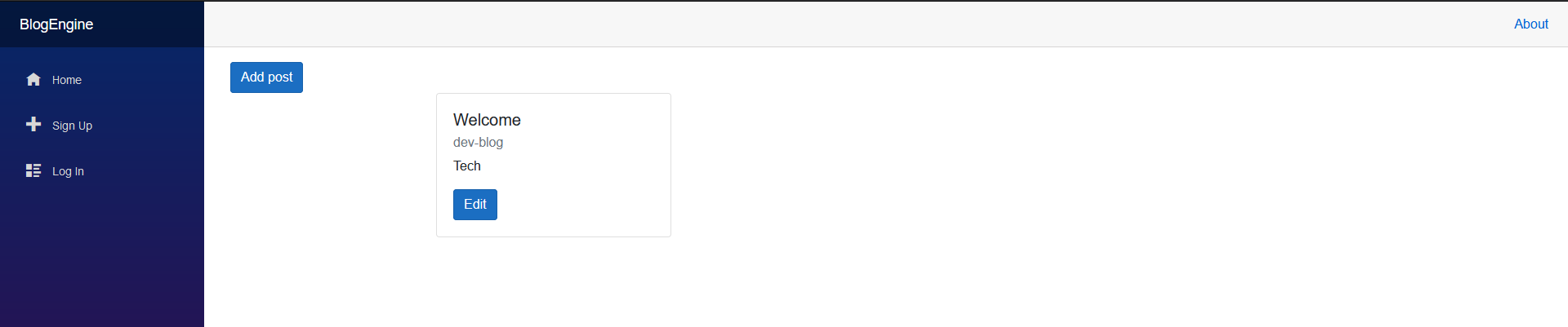
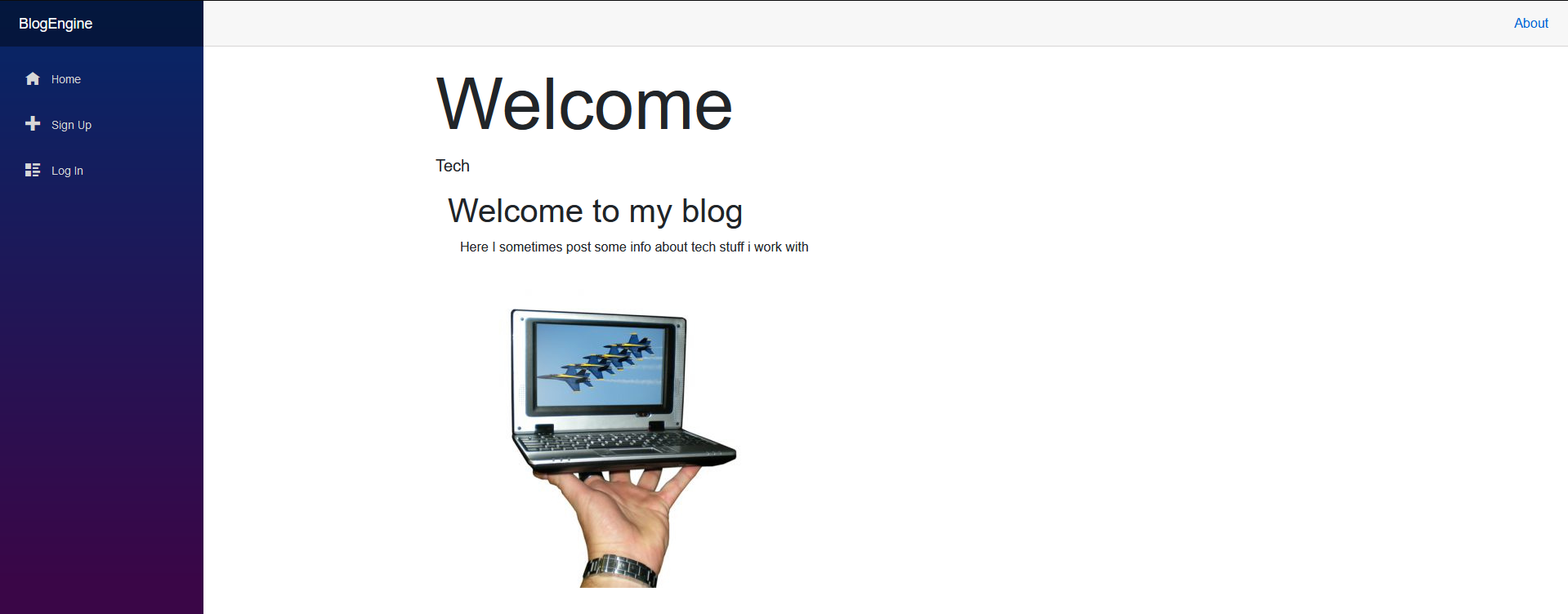
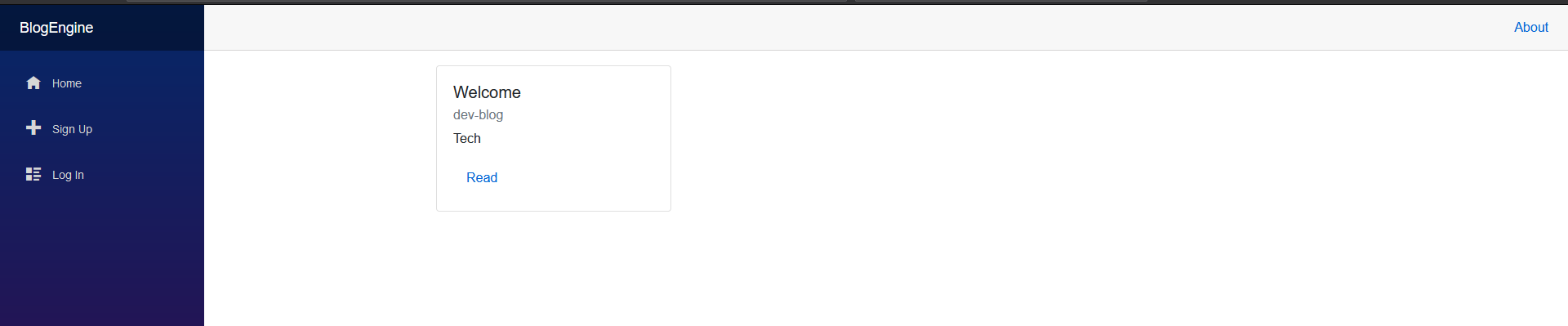
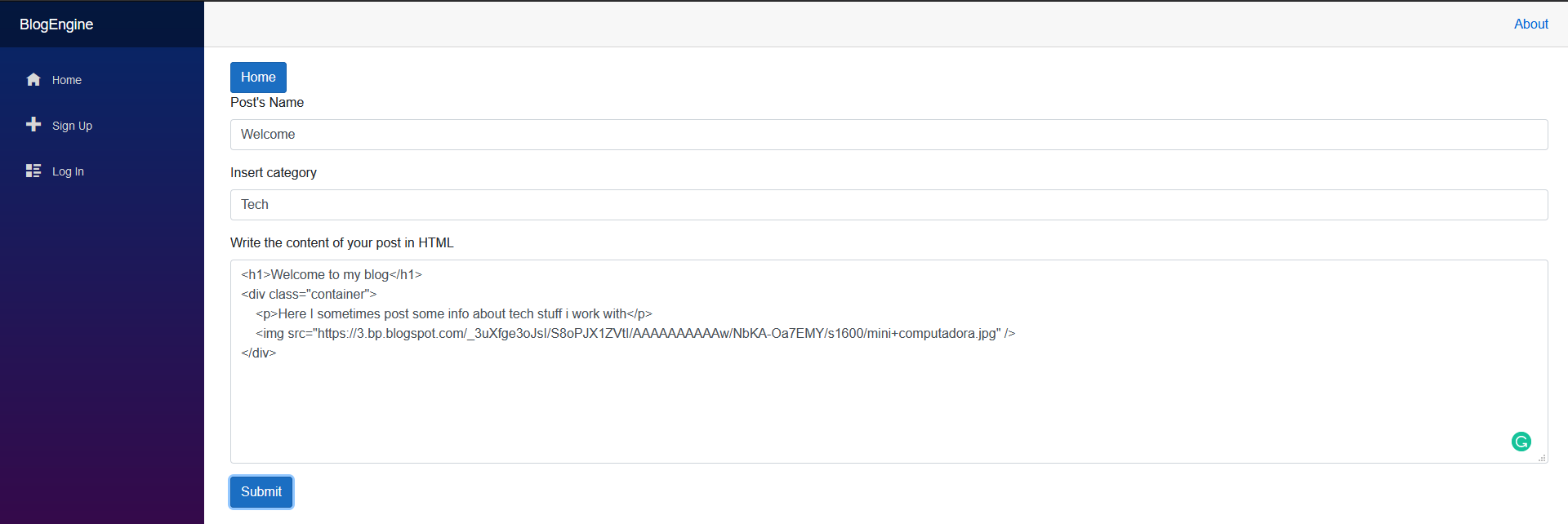
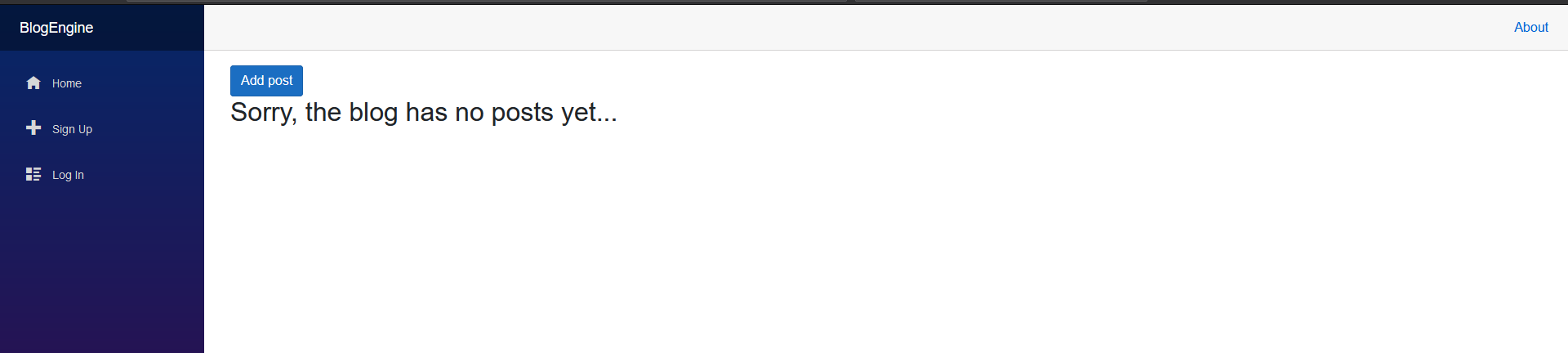
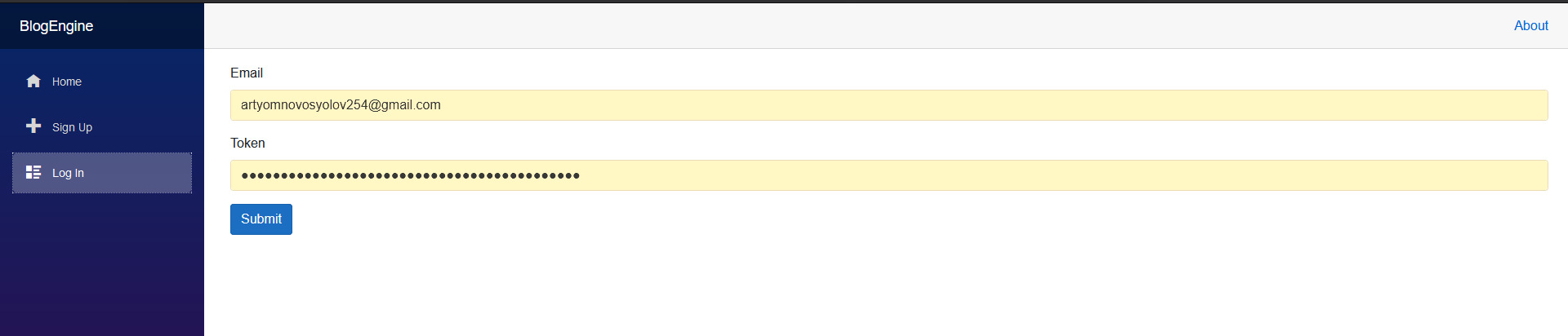
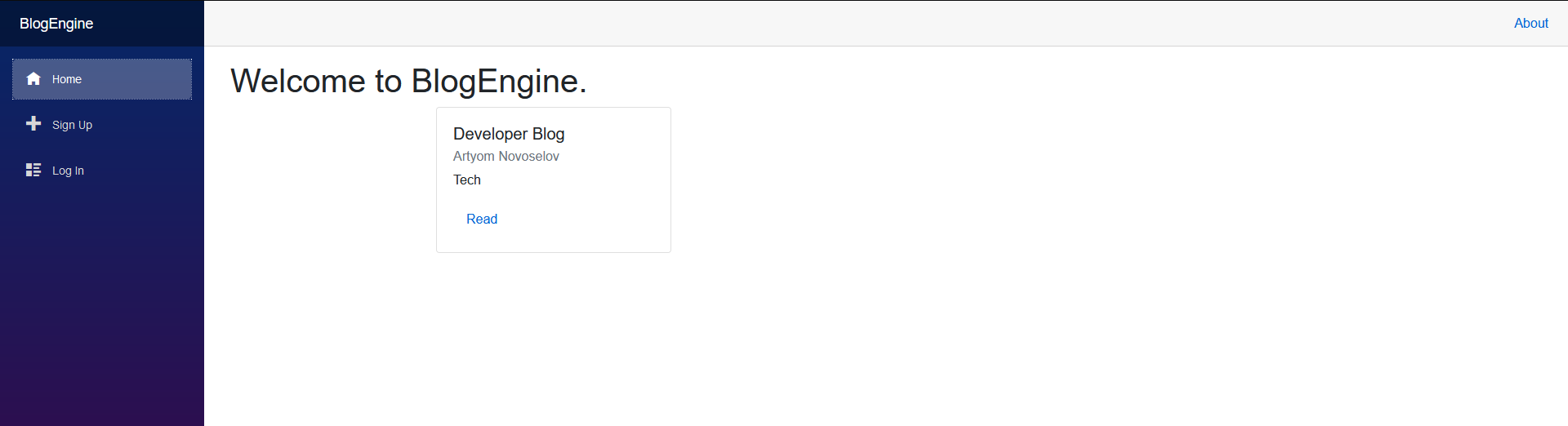
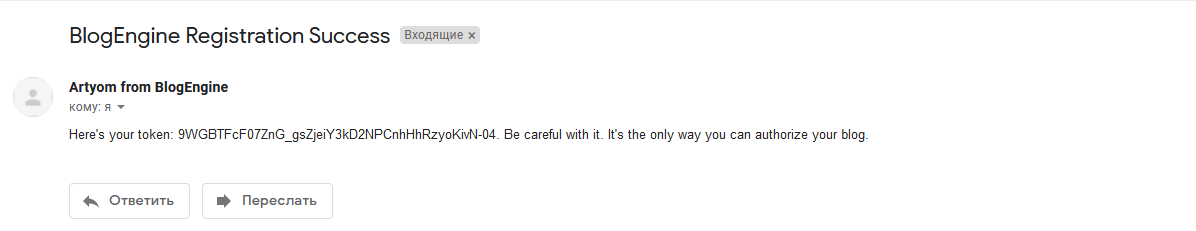
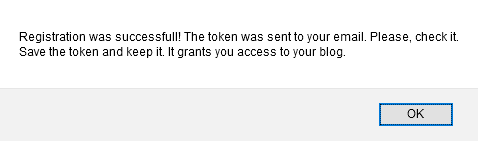
На первом можно видеть главную страницу сервиса, которая содержит боковое меню с переходами на другие разделы сервиса. В основной области окна располагается список блогов, доступных к чтению.

Второе окно представляет собой простую форму регистрации нового пользователя. По полю на каждый необходимый для создания аккаунта пункт. кнопка «Submit» выделяется на фоне остального интерфейса для удобства пользователя.

Третье изображение демонстрирует страницу входа в аккаунт. Она не имеет сильных отличий от страницы регистрации, за исключением количества полей.

Дальнейшая разработка приложения основывалась на повторение дизайн-элементов в HTML-коде и добавлении в код страниц методов для совершения HTTP-запросов к API.

2.2.4 ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ



**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. dotnet.microsoft.com
2. https://dotnet.microsoft.com/apps/aspnet
3. forem.com
4. https://github.com/forem
5. dev.to/about
6. ghost.org
7. https://github.com/TryGhost/Ghost
8. tumblr.com
9. https://www.tumblr.com/about
10. https://dotnet.microsoft.com/languages
11. https://webassembly.org/roadmap/
12. https://gs.statcounter.com/browser-market-share