Программа обучения к стажировке

Какие вещи мы **не затрагиваем** в процессе обучения, описаны здесь, чтобы понять на что пока точно не стоит концентрировать свое внимание:

* WinForms -приложения – desktop приложения
* WPF-приложения – desktop приложения
* Xamarin – мобильные кроссплатформенные приложения
* Unity – игровой движок
* ASP.NET Web Pages и Web Forms – альтернативных подход к построению web приложений
* ASP.NET MVC – затрагиваем частично, здесь можно почитать, но не сильно углубляться в детали

Инструменты:

* Visual Studio 2017
* C# 6 и 7, .Net Framework 4.6.2
* Git и SourceTree
* MS SQL Management Studio
* Resharper
* Postman
* Fiddler (редко и опционально)
* Google Chrome Developer Tools

Какой стек технологий мы **используем**, чтобы почитать детали:

* ASP.NET WebAPI 2 – как инфраструктурный web framework
* OWIN и как реализация Katana – для модели обработки запросов
* IIS 8 и выше – для хостинга
* Autofac – DI контейнер
* Hangfire – для фоновой обработки
* Entity Framework – как ORM для взаимодействия с БД
* MS SQL Server 2016 и выше – для сервера БД
* NUnit, Moq - для тестирования
* RESTSharp – для взаимодействия с внешними сервисами
* SignalR – как технология оповещения клиента о событиях со стороны сервера
* Swagger (OpenAPI) – как спецификация описания API
* AutoMapper – как библиотека для маппинга одних сущностей в другие, например из разных слоев
* FluentValidator – местами для валидации
* NLog – для логирования
* ElasticSearch – как хранилище для оптимального поиска
* ELK стек (ElasticSearch, Kibana, Curator) – для хранения, просмотра и очистки логов

Следующие концепции **используются**:

* Layered архитектура – подход к разделению архитектуры на слои, она же гексагональная архитектура
* Domain Drivement Development (DDD) – как подход к проектированию доменной модели
* Command Query Responsibility Separation (CQRS) – для организации потоков данных
* Event Sourcing – местами для организации хранения
* REST (на самом деле RESTish) – как подход к проектированию API
* SOLID принципы – для построения расширяемой и поддерживаемой модели
* GOF патттерны (фабрики, visitor-ы, декораторы, builder-ы) – для организации расширяемого и поддерживаемого кода
* Паттерны корпоративных приложений (PoEAA) (репозиторий, UnitOfWork, ORM) – для организации расширяемого и поддерживаемого приложения, тестируемости, транзакционности

1. Блок предварительной подготовки

Блок предназначен для подготовки перед началом стажировки, плюс первая неделя.

* 1. Основы ООП

**Цель:** получить понимание концепций Объектно Ориентированного Программирования, на которых строится подавляющее большинство современных приложений. Архитектура и проектирование и более сложные концепции часто опираются на ООП.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тема | Рекомендуемое время освоения | Мотивация | Комментарий | Ссылки на материалы |
| Понимание зачем ООП (Вступление Брюс Эккеля из "Философия Java") | 3 часа | Основа, тем более большинство систем строятся в соответствии с ООП парадигмой |  | Одним из лучших описаний ООП и концепций, откуда все взялось это глава «Введение в объекты» в книге Брюса Эккеля «Философия Java».  Вот, например, тут есть эта глава, правда в немного неудобном виде:  <http://programming-lang.com/ru/comp_programming/ekkel/0/j2.html> |
| Понятие классов, объектов |
| Понятие трактовки о 3 принципов ООП: инкапсуляция, наследование, полиморфизм |
| Понятие композиции, агрегации объектов | <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-aggregation-vs-composition/>  <https://www.geeksforgeeks.org/association-composition-aggregation-java/> |
| Основные паттерны: factory, singleton, decorator | 1 день | Служат основными строительными блоками при проектировании, позволяет мыслить на более архитектурном уровне | Можно изучить основные паттерны, самое главное понять какие проблемы они решают и где действительно лучше их применять | <https://metanit.com/sharp/patterns/>  Первоисточник:  <https://www.ozon.ru/context/detail/id/148946316>  Адаптированный под .NET, лично рекомендуемый вариант:  <https://www.ozon.ru/context/detail/id/31789305/> |

* 1. Основы C#

**Цель:** изучить основные конструкции языка, его возможности, привыкнуть к нему.

**Литература:** наверное лучший и точный список, по моему личному опыту - <http://sergeyteplyakov.blogspot.com/2014/04/best-books-on-cnet.html>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тема | Рекомендуемое время освоения | Мотивация | Комментарий | Ссылки на материалы |
| Установить Visual Studio | 2 часа | Основной инструмент разработки | Необходимо привыкнуть к инструменту и его возможностям | Для личных нужд можно Community версию:  <https://visualstudio.microsoft.com/ru/downloads/?rr=https%3A%2F%2Fwww.google.com%2F> |
| Основные операторы, структура программ | 1 неделя | Основные структурные блоки приложения |  | <https://metanit.com/sharp/tutorial/>  <https://professorweb.ru/my/csharp/charp_theory/level1/index.php>  <https://metanit.com/sharp/algoritm/>  Отдельно для многопоточности, очень классный раздел:  <http://www.albahari.com/threading/>  Отдельно по алгоритмам:  <https://metanit.com/sharp/algoritm/>  Про асинхронность:  <https://blogs.msdn.microsoft.com/ruericlippert/2010/12/16/c-5-4-10/> |
| Системы типов и классов в C# |
| Циклы и итераторы |
| Работа с исключениями | Основа обработки ошибок |
| Делегаты и события | Элементы для функционального программирования |
| Интерфейсы и абстрактные классы | Наиболее важная тема. Основной фундамент для проектирования и построения полиморфного поведения |
| Рефлексия типов | Понимание механизма взаимодействия с информацией о типах и членах классов | Можно понять принцип, особо углубляться не стоит. Редко используется, если не заниматься метапрограммированием(Например, DI-контейнеры) |
| Generic-и | Основные структуры данных и понимание типизации на их примере |  |
| Коллекции |
| Работа с потоками (Stream-ы), работа с System.IO | Понимание работы с ресурсами, потоками, с файловой системой и сетью | Лучше поиграться с примерами, чтобы «руки привыкли» |
| Понятие сериализации и десериализации |  |  |
| Консольное приложение чтения из файлов с сериализацией и десериализацией | 4 часа | Для закрепления основ |  |  |
| LINQ | 4 часа | Основной механизм написания запросов для вытаскивания данных из различных источников и комбинирования | Достаточно понять принцип и какие возможности предлагаются и как и где посмотреть. Тема важная, лучше тестовые примеры самим попробовать и воспроизвести |
| Задача: Написать тестовую Query на LINQ | 3 часа | Очень важно для закрепления навыков |  |
| Основы многопоточности и асинхронности | 1 день | Работа с многопоточностью, типовые проблемы и каких избежать | Вопреки нашему многопоточному миру, не так часто используется в web разработке, поэтому достаточно понятийного понимания |
| Понимание работы CLR, памяти и GC | 4 часа | Понимание сущности работы экосистемы .NET и CLR | Вещь интересная, но чисто для понимания механизмов, для практики редко применяются. Поэтому можно изучить на понятийном уровне. |
| Работа со строками и датами | 2 часа | Оперирование строками и датами – часто встречающиеся операции | Достаточно ознакомиться с возможностями, чтобы понимать что именно и как подсмотреть |

* 1. Основы SQL

**Цель:** мы для взаимодействия с БД используем Entity Framework, однако понимание как работают реляционные СУБД и как строятся запросы – один из must-have навыков, так как он дает прояснение многим другим концепциям.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тема | Рекомендуемое время освоения | Мотивация | Комментарий | Ссылки на материалы |
| Понимание основ реляционных БД, схемы, таблицы, строки | 3 дня | Самое популярный тип хранения данных |  | <https://youtu.be/jb_gkVQqikA>  <https://professorweb.ru/my/sql-server/2012/level1/>  <https://metanit.com/sql/sqlserver/> |
| Основные операторы и команды: SELECT, UPDATE, INSERT, DELETE |
| Операторы фильтрации и сортировки: WHERE и ORDER BY |
| Операторы соединения: JOIN-ы и их виды |
| Агрегация: GROUP BY, HAVING и агрегирующие функции |
| Подзапросы в JOIN и WHERE |
| Основы оптимизации, планы запросов | 3 часа | Позволяет понять как работают запросы, проблемы и пути их исправления | Скорее дополнительная тема, для ознакомления | <https://www.essentialsql.com/what-is-a-query-plan/>  <https://developer.rackspace.com/blog/understanding-a-sql-server-execution-plan/> |

* 1. Основы Web

**Цель:** получить понимание работы web приложений, серверов, сетевого стека, а также на основе этого как строятся приложения в .NET Framework мире на примере ASP.NET приложения.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тема | Рекомендуемое время освоения | Мотивация | Комментарий | Ссылки на материалы |
| Вводная лекция по работе web-а | 1 день | Понимание как работает web на концептуальном уровне | Основа основ, хоть и примерные | По личному опыту:  <https://www.youtube.com/watch?v=tiRIWEnvsPI>  Бегло посмотреть, детали и код из этого можно пропустить, основное – цикл запросов серверов: <https://youtu.be/E0t2CYo0qSk>  Тоже полезно:  <https://youtu.be/PsLzEAsphbM> |
| Модель OSI | 3 часа | Понимание сетевого стека, за счет чего и как работает | Более теоретизированно, глубокие знания не обязательны | По личному опыту:  <https://youtu.be/i16XviL8YMY>  и:  <https://professorweb.ru/my/csharp/web/level1/1_2.php> |
| Основы ASP.NET приложений | 3 дня | Прикладные навыки написания ASP.NET приложений | Основная база | Практика:  <https://metanit.com/sharp/aspnet_webapi/>  <https://docs.microsoft.com/ru-ru/aspnet/web-api/overview/getting-started-with-aspnet-web-api/tutorial-your-first-web-api>  Дополнительно:  <https://metanit.com/sharp/articles/mvc/13.php>  <https://metanit.com/sharp/articles/mvc/10.php>  <https://habr.com/ru/post/189086/> |
| Задание: Написать простое Web Api приложение | 1 день | Практические навыки взаимодействия с ASP.NET и IIS | Просто написать простое WebAPI приложение, поиграться с ним и научиться его хостить на IIS | - |
| COMET: WebSockets, LongPooling, Server Sent Events  SignalR | 3 дня | Механизмы двусторонней коммуникации сервера и клиента, механизм часто используемый в чатах, играх и других приложениях с малой задержкой | - | Основное:  <https://metanit.com/sharp/mvc5/16.1.php>  <https://docs.microsoft.com/ru-ru/aspnet/signalr/overview/getting-started/introduction-to-signalr>  Дополнительно, в JS код можно особо не вчитываться, чиать пояснения про COMET:  <https://learn.javascript.ru/ajax> |
| OWIN и Katana | 1 день | Основная технология обработки запросов в современных ASP.NET приложениях, позволяет уйти от зависимости от IIS | Важно понять принцип построения и понятия Middleware | <https://metanit.com/sharp/mvc5/11.1.php>  Цикл статей:  <https://andrey.moveax.ru/post/owin-introduction-part1>  А также:  <https://habr.com/ru/post/202018/> |

1. Блок предварительной подготовки. Часть 2. Работа с данными.

Блок предназначен познакомиться с основными взаимодействия с РСУБД из .NET приложения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тема | Рекомендуемое время освоения | Мотивация | Комментарий | Ссылки на материалы |
| DAL | 4 дня | Основы взаимодействия с БД | Просто ознакомиться, попробовать тестовые примеры | <https://professorweb.ru/my/ADO_NET/base/level1/ado_net_index.php> |
| Основы ADO.NET |
| Основы Entity Framework | Основной механизм взаимодействия с БД | Необходимо достаточно хорошо ознакомиться | <https://metanit.com/sharp/entityframework/> |
| Navigation Properties и Foreign Keys |
| Code First и миграции |

1. Блок проектирования

Блок предназначения для понимания современных подходов и приемов к проектированию web приложения

* 1. Работа с API

Здесь будет больше очного объяснения, но для начала можно почитать:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тема | Рекомендуемое время освоения | Мотивация | Комментарий | Ссылки на материалы |
| REST | 3 часа | Основной подход к построению API |  | Не совсем правильная статья, имеющая много неточностей, но для начинающих и пояснения примерного концепта:  <https://habr.com/ru/post/38730/> |
| Аутентификация | 1 день | Must have часть любой системы |  | <https://habr.com/ru/company/dataart/blog/311376/>  <https://habr.com/ru/company/mailru/blog/343288/>  <https://habr.com/ru/company/dataart/blog/262817/> |

* 1. Основы проектирования

Здесь будет больше очного объяснения, но для начала можно почитать

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тема | Рекомендуемое время освоения | Мотивация | Комментарий | Ссылки на материалы |
| Dependency injection и зависимости | 1 день | Основной подход к управлению зависимостями |  | <https://habr.com/ru/post/434380/>  <https://habr.com/ru/post/176007/>  <https://medium.com/@chrisverwijs/a-step-by-step-guide-to-using-ninject-for-dependency-injection-in-c-68a125bd7fa4> |
| SOLID | 3 дня | Принципы, которые помогают строить расширяемую системы |  | <http://sergeyteplyakov.blogspot.com/2014/10/solid.html>  Там же цикл статей по принципам отдельно |
| Архитектура: layered и другие типы | 1 день | Основы архитектуры |  | <https://martinfowler.com/bliki/PresentationDomainDataLayering.html>  <https://habr.com/ru/company/it-grad/blog/276297/> |
| Паттерны корпоративных приложений | 2 дня | Основные паттерны | Достаточно почитать про Repository, QueryObject, POCO(POJO), DTO, Unit Of Work, Domain Model | <https://martinfowler.com/eaaCatalog/> |

1. Блок тестового приложения

Блок предназначен для написания тестового приложения как эмуляция корпоративного приложения, выявления проблем различных подходов и его эволюционирования через эти подходы.

Будем писать приложение, находить в нем потенциальные проблемы, исправлять и доводить его до копии Enterprise приложения.

1. Блок смежных областей

Блок предназначен для освоения практик тестирования и DevOps

* 1. Основы тестирования

**Литература:** Р. Ошуроув «Искусство автономного тестирования с примерами на С#» (The Art Of Unit Testing Second Edition With Examples In C#)

* 1. Основы DevOps

Тут будем изучать на практике. Разве что можно ознакомиться с понятием CI и попробовать Jenkins с Powershell