Содержание

[ВВЕДЕНИЕ 2](#_Toc483143440)

[1.1 Паттерн проектирования 3](#_Toc483143441)

[1.2 IIS 5](#_Toc483143442)

[2 МОДЕЛЬ 7](#_Toc483143443)

[2.1 Контекст базы данных 7](#_Toc483143444)

[2.2 Схема базы данных. 7](#_Toc483143445)

[2.3 Внешнее взаимодействие с данными 9](#_Toc483143446)

[3 КОНТРОЛЛЕР 10](#_Toc483143447)

[3.1 Схема управляющих сигналов. 10](#_Toc483143448)

[3.2 Схема авторизации 13](#_Toc483143449)

[3.2.1 Схема работы SSL 15](#_Toc483143450)

[3.3 Взаимодействие с базой данных 16](#_Toc483143451)

[3.4 Методы оптимизации контроллеров 17](#_Toc483143452)

[3.4.1 Асинхронный подход 17](#_Toc483143453)

[3.4.2 Отложенные задачи 18](#_Toc483143454)

[4 ПРЕДСТАВЛЕНИЕ 19](#_Toc483143455)

[4.1 Реализация представления в проекте 19](#_Toc483143456)

[4.2. Методы оптимизации представления 20](#_Toc483143457)

[4.2.1 AJAX 20](#_Toc483143458)

[4.2.2 Кэширование 21](#_Toc483143459)

[4.2.3 Минификация кода 22](#_Toc483143460)

[5 Рейтинг Эло 24](#_Toc483143461)

[5.1 Возникновение алгоритма 24](#_Toc483143462)

[5.2 Вычисление рейтинга Эло 24](#_Toc483143463)

[6. Алгоритм Ярроу 26](#_Toc483143464)

[6.1 Компоненты алгоритма Ярроу 26](#_Toc483143465)

[6.1.1 Накопитель энтропии 26](#_Toc483143466)

[6.1.2 Механизм усложнения 27](#_Toc483143467)

[6.1.3 Механизм генерации 27](#_Toc483143468)

[6.1.4 Механизм управления усложнением 27](#_Toc483143469)

[7 Список использованных источников 28](#_Toc483143470)

# 

# ВВЕДЕНИЕ

Каждый человек, связанный с управлением, сталкивается с главной, и самой важной, проблемой в принятии решения - распределением ресурса. Вне зависимости от направленности деятельности компании, ресурс, каким бы он ни был, всегда будет ограничен. Ключом к развитию предприятия служит грамотная стимуляция (капиталовложение) в сегмент, приносящий большую прибыль организации, где под капиталом подразумеваются различные активы организации.

Все компании, помимо создания продукта, занимаются его реализацией, даже самый удобный, выгодный и перспективный проект должен быть правильно преподнесен: в нужное время, нужном месте, должном объеме. Отсюда вытекает необходимость в системе, которая могла бы дать оценку перспективности развития сегмента в регионе, спрогнозировать поведение объекта, выделить точки концентрации внимания компании.

Предъявленные требования могут быть реализованы, с применением модернизированной системы рейтингования Эло, несущей в себе принцип рассмотрения статистической модели исходов событий, и принятия математических приближений, позволяющих простейшие вычисления. Данная модель характеризуется быстротой изменения рейтинга в случае ввода объекта в систему, диверсификации ситуации в системе, ререлизе объекта через промежуток времени.

Выполнив пост-обработку выходных рейтингов, посредством нормирования, можно с высокой точностью распределить заложенный ресурс между сегментами, на него претендующими.

При реализации придержемся тенденции перевода всех ресурсов в web-среду, продиктовано это стремлением увеличить мобильность сервисов и приложений, развить устойчивость систем, обеспечить кроссплатформенность и значительно снизить издержки компании во всем «жизненном цикле» продукта. При реализации web-сервиса рассмотрим современные паттерны разработки и подходы к проектированию, различные Фреймворки и технологии.

В контексте работы, в качестве объекта будут рассмотрены отделы реализации продуктов страховой компании “Югория”.

## INTRODUCTION

Every person connected with management faces the main, and most important, problem in the solution - the distribution of the resource. Regardless of the direction of the company's activities, the resource, whatever it is, will always be limited. The key to the development of the enterprise, providing competent stimulation (investment) in the segment, bringing a greater profit to the organization, where the capital means the various assets of the organization.

All companies, besides creating, are engaged in its implementation, even the most convenient, profitable and promising project should be properly presented: at the right time, the right place, the proper volume. Refuse the need for a system that could assess the prospects for the development of the segment in the region, predict the behavior of the object, highlight points of concentration.

The presented requirements can be realized, taking into account the modernized rating system of Elo, which carries the principle of consideration of statistical models of outcomes of events, and the adoption of mathematical approximations that give the simplest calculations. This model is characterized by the rapidity of changes in the case of entering an object into the system, real-time real-time status diversification.

After executing post-control of the output ratings, with the help of the normalization, you can distribute the allocated resource between the segments that apply to it with high accuracy.

"Carrying out visits to other countries on the network, this desire to increase mobility of services and applications is dictated, to ensure the stability of systems, to ensure cross-platform and cost significance in the whole" life cycle ". When implementing a web service, modern development and approaches to design, various Frameworks and technologies are considered.

In the context of the work, the product sales departments of the insurance company «Yugoria» will be considered as the object.

# Паттерн проектирования

Данная работа выполнена на основе паттерна проектирования MVC (Model-View-Controller)-схема взаимодействия, при которой разделение данных приложения, интерфейса и управляющей логики производится таким образом, что конфигурирование и модификация каждого элемента может производиться независимо друг от друга.

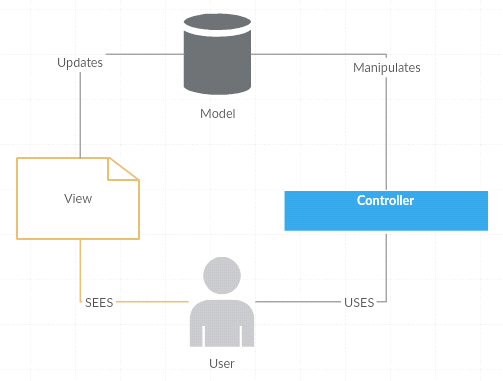


Схема 1.1.1 – Структура паттерна MVC

Компоненты:

* Модель представляет совокупность данных приложения и их описания, реагирующих на команды контроллера, изменяя своё состояние.
* Представление являет собой отображение данных модели пользователю, реагируя на модификацию модели или её состояния.
* Контроллер отвечает за интерпретацию действий пользователя, модифицируя модель или её состояние.

Недостатки концепции MVC:

* Независимая реализация блоков ведет к более сложной реализации их взаимодействия, что сказывается на ресурсоемкости.
* Необходимость разделения на блоки новых модулей при расширении функционала.
* Ограниченность в выборе хостинга

Достоинства:

* Глобальность архитектуры сервиса, что ведет к простоте ведения приложения, быстроте локализации проблемы, упрощению тестирования и отладки.

# IIS

Как было упомянуто ранее, конфигурацию данного паттерна проектирования и реализации поддерживает не каждый хостинг. Что бы обойти проблему размещения проекта, для систем ASP.NET необходимо использовать системную надстройку IIS. Internet Information Services – набор интегрированных в операционную систему виртуальных серверов, обеспечивающих размещение сервиса и поддерживающих работу с ним. Взаимодействие с сервисом подразумевает поддержку работы набора протоколов, в которые входят:

* HTTP
* HTPPS
* FTP
* POP3
* SMTP
* NNTP

Сервис IIS является специализированной надстройкой от Microsoft для размещения сервисов реализованных с применением технологий Фреймворка .NET.

Достоинствами данного сервиса в сравнении с решениями, такими как Apache, являются:

* Официальная поддержка
* Высокая безопасность системы

Недостатками:

* Моноплатформенность
* Производительность
* Темпы развития
* Решение платное

Работу IIS с http-запросом можно представить следующим образом:

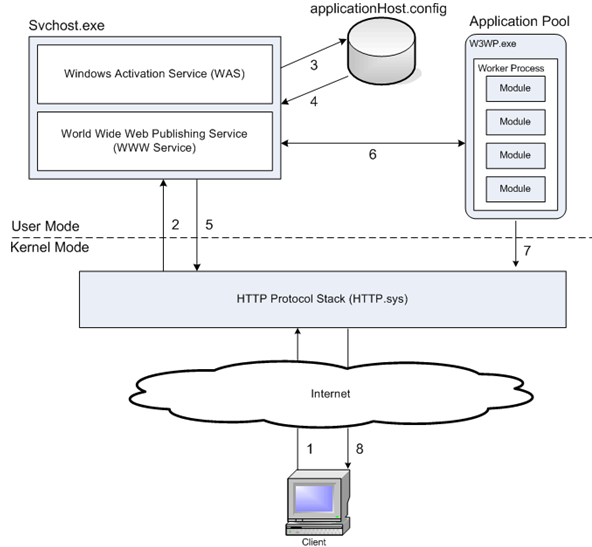


Схема 1.2.1 – Обработка http-запроса сервисом IIS

1. Браузер обращается к URL веб-сервера, на стороне сервера запрос перехватывает IIS драйвер **HTTP.SYS**.
2. **HTTP.SYS** обращается к **WAS**для получения информации из хранилища конфигурации.
3. Служба **WAS**запрашивает конфигурацию из хранилища IIS (applicationHost.config).
4. Конфигурационную информацию получает служба **W3SVC**(WWW-Service), эта информация содержит в себе данные о пуле приложений.
5. Служба **W3SVC**использует эту информацию для конфигурации **HTTP.SYS**.
6. Служба **WAS**запускает процесс **W3WP.exe** для пула приложений.
7. В процессе **W3WP.exe** работает приложение веб-сайта, которое, формирует и возвращает ответ драйверу **HTTP.SYS**.
8. **HTTP.SYS** отправляет ответ браузеру.

# 2 МОДЕЛЬ

# 2.1 Контекст базы данных

В контексте данной работы, модель системы являет собой совокупность данных и модели данных, то есть содержательную часть табличных данных – таблицы, их поля и связи в базе данных, содержание полей, и концептуальное описание конкретной предметной области - интерпретируемые языком разработки классы, зависимости и атрибуты.

Реализация содержательной части модели произведена на реляционной базе данных – наборе таблиц, состоящих из кортежей (совокупности строк и столбцов), хранящих в себе информацию, подлежащую четкой нормализации и типизации. Под нормализацией следует понимать минимизацию избыточности информации, путем структуризации базы данных. (НОРМАЛИЗАЦИЯ РАСПИСАТЬ)

Выбор реляционной базы данных был продиктован спецификациями проекта, а именно требованием к лаконичности базы данных, простоте и отсутствию необходимости в неформализованной сортировке (NoSQL решения). На данный момент реляционные базы данных используются в 93% проектов со схожими предъявляемыми требованиями.

Плюсы реляционных баз данных:

* Эта модель данных отображает информацию в наиболее простой для пользователя форме
* Основаны на развитом математическом аппарате, который позволяет достаточно лаконично описать основные операции над данными.
* Позволяет создавать языки манипулирования данными не процедурного типа.
* Манипулирование данными на уровне выходной БД и возможность изменения.

Минусы:

* Трудоемкость разработки структуры БД

# 2.2 Схема базы данных.

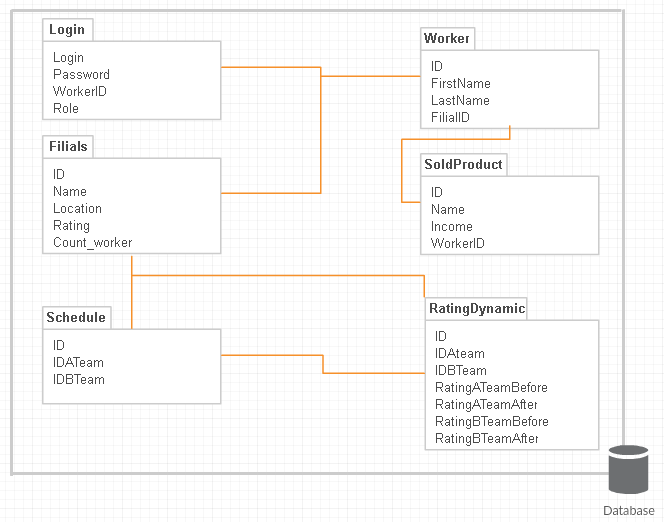


Схема 2.2.1 – Таблицы данных

Таблица Филиалов компании(Filials) содержит: идентификационный номер филиала, его наименование, локацию, текущий рейтинг, количество работников.

Таблица работников(Worker) хранит: идентификационный номер агента, имя, фамилию, идентификационный номер филиала, к которому он относится.  
Таблица реализованных продуктов(SoldProduct) имеет: идентификационный номер операции, наименование продукта, итоговая прибыль с реализации, идентификационный номер агента-инициализатора продажи.

Таблица RM расписания(Schedule) располагает: идентификационным номером события, идентификационным номером первой команды, идентификационным номером второй команды.

Таблица динамики рейтинга(RatingDynamic) содержит: идентификационный номер состоявшегося сравнения, идентификационный номер первой команды, идентификационный номер второй команды, рейтинг первой команды до, рейтинг первой команды после, рейтинг второй команды до, рейтинг второй команды после.

Таблица аутентификации(Login) хранит в себе: логин, пароль, идентификационный номер агента, роль агента, отвечающая за правомерность использования контроллеров.

# 2.3 Внешнее взаимодействие с данными

В качестве СУБД, используемой для административной работы с данными, применяется инструмент Microsoft SQL 2012. Данная СУБД является внешним инструментом управления реляционными базами данных, позволяющим обеспечить поддержку хранения данных и взаимодействие с пользователями и администраторами.

Отметим основные плюсы и минусы данной СУБД:

Плюсами являются:

* Масштабируемость
* Производительность
* самоадминистрирование
* Интегрированность с платформой .NET

Минусами:

* Моноплатформенность

Так же необходимость в СУБД продиктована важным функционалом – безопасным резервированием и восстановлением. В цикле разработки и ведения продукта, связанным с вычислением и хранением данных, создание резервных копий базы минимизирует вероятность её утраты.

BACUP данных системы включает в себя несколько этапов:

* Копирование и перенос данных
* Шифрование
* Сжатие

# 2.3.1 Этапы резервного копирования

Первым этапом происходит копирование данных из табличного представления и их перенос в заданную защищенную от потери данных директорию – сменный носитель информации, удаленный сервер по протоколу ftp, или же иное хранилище данных.

Второй этап резервирования шифрование - оно проводится в целях защиты данных от несанкционированного доступа во время хранения информации. Шифрование реализовано с использованием системы RSA

Генерация ключей происходит следующим образом:

1. Выбираются два простых числа p,q по 1024 бита
2. Вычисляется их модуль

Формула 2.3.1.1 - Модуль простых чисел

1. Вычисляется значение функции Эйлера для модуля

Формула 2.3.1.2 –Функция Эйлера

1. Выбирается целое число E взаимно простое со значением функции Эйлера
2. Вычисляется значение числа D, мультипликативно обратное к числу E

Формула 2.3.1.3 – Вычисление мультипликативно обратного числа

Таким образом, мы имеем ключ для шифрования данных, основанного на криптографически-стойкой односторонней функции 2.3.1.4

Формула 2.3.1.4 – Шифрование данных

Где:

* S – зашифрованные данные
* M – исходные данные
* E - взаимно простое со значением функции Эйлера число
* N – модуль простых чисел

Благодаря свойству мультипликативности, появляется возможность обратного преобразования зашифрованных данных, которое происходит по формуле 2.3.1.5

Формула 2.3.1.5 – Дешифрование данных

Где:

* S – зашифрованные данные
* M – исходные данные
* D - мультипликативно обратное к числу E
* N – модуль простых чисел

Третьим этапом идет сжатие данных, данный этап проводится с целью экономии места, так как базы данных могут иметь большой объем и процесс резервирования может быть частым. Алгоритмом сжатия служит GZIP. Алгоритм кодирования GZIP основан на двух постулатах сжатия:

* Алгоритме LZ77 – принцип работы которого, основан на замене повторных вхождений данных на ссылки (повторные цепочки заменяются на первый экземпляр)
* Алгоритме Хаффмана – принципе кодирования символов с переменной длинной, где ни одно кодовое слово не является префиксом другого (чем чаще вхождение «символа», тем короче его код, коды мало коррелируют)

Преимущества алгоритма

* Компромисс между скоростью и степенью сжатия
* Аппаратная поддержка
* Кроссплатформенность
* Интегрированная браузерная поддержка

# 3 КОНТРОЛЛЕР

# 3.1 Схема управляющих сигналов.

Система управления данными разбита на три подмодуля, несущих в себе определенные функции. Сделано это для упрощения тестирования, интеграции и обслуживания.

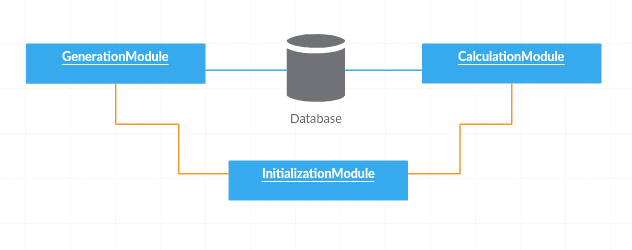


Схема 3.1.1 – Модули управления данными

Блок инициализации включает в себя контроллер системного времени, отвечающий за трассировку событий (временных точек), и контроллер активации блоков генерации и расчета.

Блок генерации пар команд - алгоритм, работа которого, заключается в выборке филиалов из базы данных для создания «игрового» расписания, алгоритм базируется на псевдослучайной основе, для наработки практического опыта был взят алгоритм Ярроу.

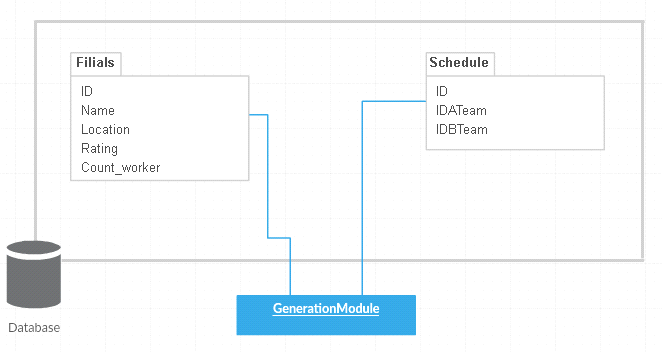


Схема 3.1.2 – Структура модуля генерации

Блок расчета рейтинга - основывается на системе расчета рейтинга Эло, при инициализации модуля происходит выборка данных о рейтинге филиалов на момент инициализации, выборка данных о количестве продаж конкретных филиалов за отчетный период, по алгоритму расчета Эло вычисляется новый рейтинг, далее происходит запись и обновление в таблиц.

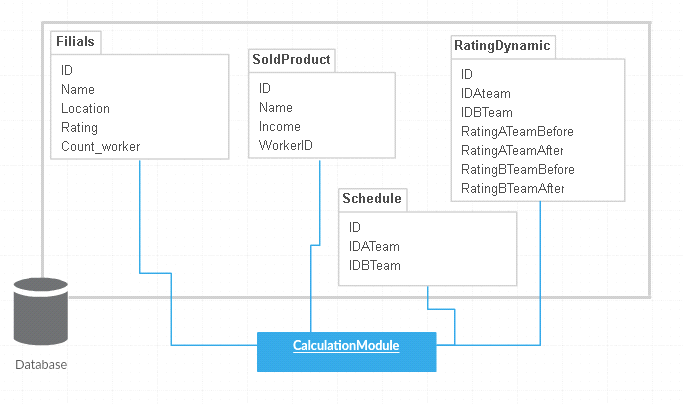


Схема 3.1.3 – Структура модуля расчетов

Данная система существует в автономном режиме, но так же существует инструмент ручного обращения к системе, отсюда вытекает необходимость создания системы аутентификации пользователя инициализирующего контроллер.

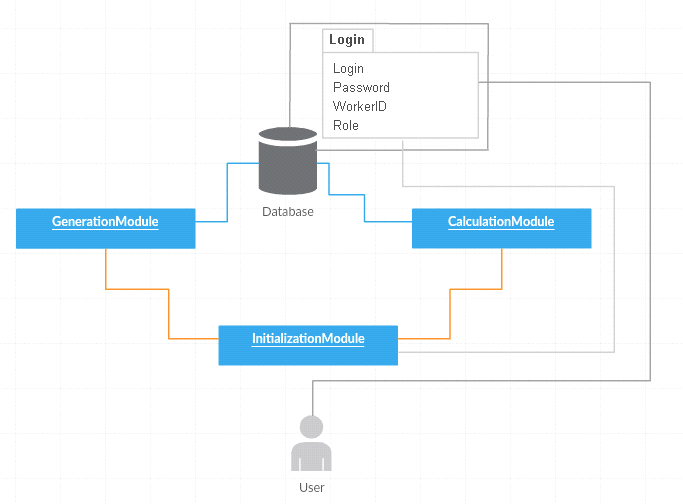


Схема 3.1.3 – Структура управления доступом

# 3.2 Схема авторизации

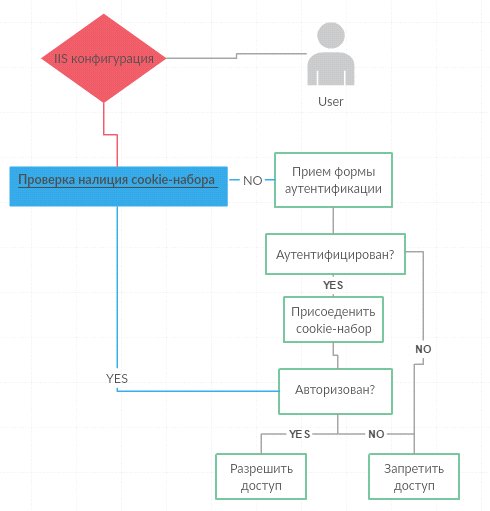


Схема 3.2.1 – Проверка аутентификации в приложении

Реализована данная система на основе фильтров аутентификации, они срабатывают до любого другого фильтра и выполнения метода, для любого контроллера системы, имеющего атрибуты проверки.

В качестве элемента авторизации, в системе применяется аутентификация с помощью WEB-форм. Посредством аутентификации с помощью форм платформа ASP.NET создает cookie-набор безопасности для зарегистрированных пользователей, обслуживает их и автоматически поддерживает контекст безопасности для последующих запросов.

Аутентификация с помощью форм основана на билетах (также называемых маркерами). Это значит, что когда пользователь регистрируется, он получает так называемый билет с базовой информацией о себе. Информация сохраняется в зашифрованном cookie-наборе, который присоединяется к ответу, так что автоматически отправляется в каждом последующем запросе.

Когда пользователь запрашивает контроллер ASP.NET, который не доступен анонимным пользователям, исполняющая среда ASP.NET проверяет, имеется ли билет аутентификации с помощью форм. Если нет, производится автоматическая переадресация на страницу входа.

Плюсами данной конфигурации являются:

* Полный контроль над кодами аутентификации
* Браузерная кроссплатформенность
* Контроль над frontend частью формы, а соответственно дополнительной валидацией на стороне клиента и повышением защиты от SQL-инъекций

Минусы системы :

* Необходимость поддержки каталога удостоверений пользователей, в отличии, например, от Windows-autentification и сервисов сторонней авторизации.
* Необходимость ведения защиты от вмешательства в сетевой трафик.

Аутентификацию сконфигурирована так, чтобы она шифровала и подписывала cookie-наборы, в результате существенно затрудняя злоумышленнику задачу получения информации от нее. Так же cookie-наборы не содержат никакой важной информации, и потому они не включают пароли, введенные при аутентификации.

Но есть вероятность, что злоумышленник перехватит незашифрованный трафик, извлечет из него cookie-набор (который уже зашифрован) и воспользуется им в своих целях. Расшифровывать его ему не понадобится - достаточно просто снабдить им собственные пересылаемые запросы. Противостоять такой атаке повтором можно только за счет применения технологии SSL для всех контроллеров.

Технология SSL (Secure Sockets Layer - уровень защищенных сокетов) позволяет шифровать коммуникации через HTTP. Протокол SSL поддерживается широким кругом браузеров и гарантирует, что передача информации между клиентом и веб-сервером не может быть расшифрована злоумышленниками. Технология основана идее наличия специализированного сертификата.

Сертификат содержит:

* название организации и адрес держателя
* открытый ключ держателя, который будет использоваться для реализации ключа SSL-сеансов для шифрования коммуникаций
* даты проверки сертификата
* серийный номер сертификата

# 3.2.1 Схема работы SSL

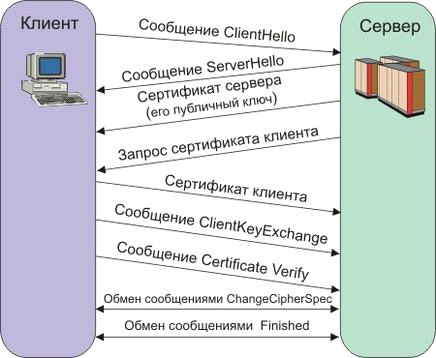


Схема 3.2.1.1 – Клиент-серверный обмен SSL

* Клиент отправляет запрос на соединение с сервером.
* Сервер подписывает свой сертификат и отправляет его клиенту
* Клиент проверяет, издан ли сертификат CA, которому он доверяет.
* Клиент сравнивает информацию в сертификате с информацией, присланной сайтом (включая доменное имя и его открытый ключ). Клиент также проверяет, допустим ли сертификат стороны сервера, не был ли он отменен и издан ли он заслуживающим доверия CA. Затем клиент принимает соединение.
* Клиент сообщает серверу, какие ключи шифрования он поддерживает для коммуникаций.
* Сервер выбирает наибольшую подходящую длину ключа и информирует клиента.
* Используя указанную длину ключа, клиент случайным образом генерирует симметричный ключ шифрования. Он будет использован в период транзакции между сервером и клиентом. Это гарантирует оптимальную производительность, поскольку симметричное шифрование намного быстрее асимметричного.
* Клиент шифрует ключ сеанса, используя открытый ключ сервера (из сертификата), и затем отправляет серверу зашифрованный ключ сеанса.
* Сервер принимает зашифрованный ключ сеанса и расшифровывает его, используя свой защищенный ключ. После этого и клиент, и сервер имеют общий секретный ключ, и могут применять его для шифрования коммуникаций в период, пока длится сеанс.

# 3.3 Взаимодействие с базой данных

Взаимодействие с данными WEB-сервиса осуществляется посредством LINQ-запросов (Language Integrated Query) - запросов среды разработки к базе данных. Работа LINQ-запросов основана на интерпретации и преобразовании компиляционной средой LINQ-выражений в SQL-запросы, которые осуществляют взаимодействие с базами данных, используя объектно-ориентированную технологию работы с данными ADO.NET (Active Data Object). ADO.NET - представляет собой частный случай технологии ORM (object-relational mapping), позволяющей абстрагироваться от написания SQL кода. Она выполняет обязанности по связыванию базы данных с концептуальной моделью данных, используемой в объектно-ориентированных языках программирования, создавая виртуальную базу объектов в проекте.

Технология предоставляет широкий функциональный спектр:

* Осуществление двухстороннего преобразования работы с классовыми терминалами в табличное представление и наоборот
* Интерфейсное обеспечение CRUD-операций (Create, Read, Update, Delete) , подразумевающих полный контроль над базой данных
* Поддержка паттерна Code First, представляющего собой возможность создания и обновления табличной базы данных согласно составу и содержанию программных классов.

На данном этапе разработки был выбран функциональный Фреймворк технологии ADO – Entity Framework.

К преимуществам данной технологии можно отнести следующее:

* Ориентированность на программистов, а не на администраторов БД
* Интегрированность со средой
* Возможность осуществлять миграции (изменение структуры БД)
* Возможность генерации не статичных запросов

Но есть и существенный недостаток:

* Скорость работы с базой данных

Этап обращения к базе данных считается узким местом во всех WEB-проектах. При работе с базой данных, lINQ-выражения проходят этап интерпретации, что само по себе занимает часть вычислительного и временного ресурсов, также обстоит дело с генерируемым кодом – часто он не является оптимальным. Поэтому при рационализации работы сервиса необходимо выявлять места с необходимостью быстроты реакции БД, объемными или часто повторяющимися участками кода, сложных запросов.

В данной работе при извлечении количества и принадлежности продажи продукта генерируется объемный SQL код с большим количеством итераций, посему было принято решение о внедрении Фреймворка Dapper - технологии обращения к данным, обходящей шаги интерпретации, запросы формируются в строковом эквиваленте, что позволяет снизить расход ресурсов системы. Но при этом, Фреймворк лишен остального функционала, поэтому работа с данными в сервисе построена на симбиозе c Entity Framework.

# 3.4 Методы оптимизации контроллеров

# 3.4.1 Асинхронный подход

WEB-сервис предполагает множественный доступ к ресурсу, отсюда вытекает необходимость в высокой производительности и минимизации занятия рабочего ресурса, это реализуемо посредствам распараллеливания и контроля работы пользователей и процессов. Данная среда и язык разработки позволяют реализовать асинхронный подход к работе.

Такой подход обеспечивает лучшее быстродействие системы посредствам отделения потоковой группы для реализации того или иного вида работ, не останавливая при этом основной ствол выполнения программы.

Благодаря асинхронности, есть возможность обхода "узких" мест таких как: многотабличные и многостроковые запросы, вызов функций и методов, не влияющих на прямую функциональность сервиса, но необходимых, выполнение операций прослушивания очередей, переопределение и конвертация переменных классов.

Но из плюсов подхода вытекают и его минусы, а прежде всего отсутствие контроля ликвидности ведения фоновых процессов. Если, при выполнении части запроса, будет выдано исключение, то оно будет проигнорировано и средой разработки, и исполняемым кодом, соответственно будет нарушена целостность и правильность данных с последующей некорректностью работы.

Большую долю ошибок в асинхронном режиме работы вызывает искажение данных в результате асинхронных изменений или же клинчей (взаимных/самоблокирования).

Для устранения этого минуса были введены ограничения доступа к рабочему ресурсу:

* Семафоры - объекты кода, ограничивающие количество рабочих потоков входящих в закрытую область.
* Мьютексы - одноместные семафоры.

# 3.4.2 Отложенные задачи

В цикле работы любого WEB-проекта есть задачи, исполнение которых подразумевает исполнение в отложенном режиме, или же выполнение в режиме расписания, для реализации этих функций в системе был реализован Фреймворк HangFire.

С помощью данной технологии, являющей собой планировщик задач, была создана система ведущая хранение, учет и обязанности по выполнению необходимых работ, так же система отвечает за реинициализацию задачи и функцию логирования, в случае экстренного завершения.

В контексте проекта, на Фреймворк легла нагрузка по отложенной инициализации модулей расчета и генерации, фоновому логированнию и работа с почтовым клиентом.

Работа Фреймворка основана на открытых API, через которые он общается с приложением. При поступлении рабочей заявки, в базу данных вносится её контекст и атрибуты, отражающие период, частоту инициализаций. В случае аварийного завершения задачи, происходит лог-выписка и выполнение задачи откладывается на заданный администратором тайм-аут.(ЛИТЕРАТУРА?)

# 4 ПРЕДСТАВЛЕНИЕ

# 4.1 Реализация представления в проекте

В данной работе представление, согласно паттерну проектирования, отвечает:

* За получение необходимых данных от контроллера и отправку их пользователю;
* Предобработку введённых данных пользователем данных;
* Передачу параметров и влияние на состояние модели, через отправку ей сообщений и вызов контроллера.

Frontend часть реализована на комбинации Html, CSS, JavaScript и интерпретируемого в html языка C#. Представление, как сказано ранее, формализует данные клиент-серверного обмена, при этом в обмене участвуя. В зависимости от реализации, назначения и насыщенности данными доля frontend-нагрузки на трафик может сильно варьироваться, доходя до ста процентов, что сказывается на быстродействии. (МАЛОВАТО ДОПИЛИТЬ)

# 4.2. Методы оптимизации представления

# 4.2.1 AJAX

Для уменьшения серверной нагрузки был реализован подход к построению интерактивных пользовательских интерфейсов веб-приложений AJAX(Asynchronous JavaScript and XML), подход фонового клиент-серверного обмена данными без полной перегрузки веб-страницы.

Преимущества использования:

* Экономия трафика
* Уменьшение нагрузки на сервер
* Ускорение реакции интерфейса
* Почти безграничные возможности для интерактивной обработки

Недостатки:

* Отсутствие интеграции со стандартными инструментами браузера
* Усложнение проекта
* Требуется включенный JavaScript в браузере

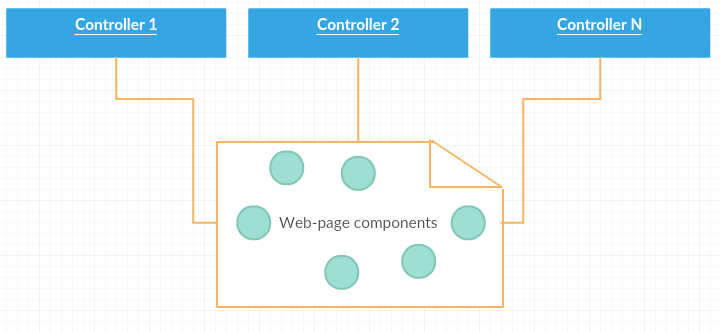


Схема 4.2.1.1– Структура моно станичного построения

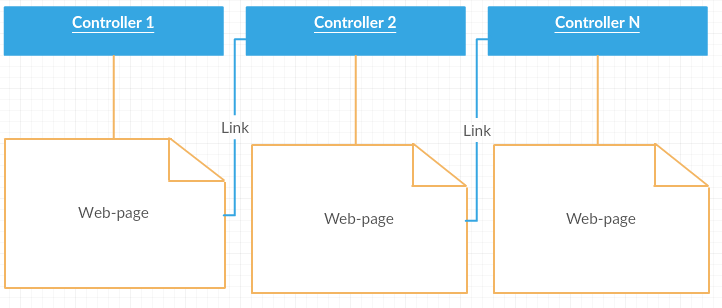


Схема 4.2.1.1– Структура много станичного построения

# 4.2.2 Кэширование

WEB-кэширование – это подход к проектированию сервисов, при котором закладывается возможность сохранения в памяти копии генерируемой информации, требующей выделения большой производительности или же времени.

В контексте данной работы были реализованы два подхода к кэшированию:

* Кэширование вывода – процесс генерации и сохранения web-страницы, отправляемой пользователю.
* Кэширование данных - этап выборки и сохранения результатов из базы данных.

Использование кэширования ведет к повышению таких параметров как производительность и масштабируемость за счет снижения времени и обхода узких мест (обращение к базе данных), соответственно.

При эксплуатации технологии следует иметь в виду, что рентабельно и логично сохранять данные и разметку отвечающие ряду параметров:

* Приемлемая статичность информации
* Отсутствие требований к динамичному изменению
* Востребованность в многоразовых обращениях

Для разнесения нагрузки на память, в работе предусмотрена возможность кэширования, как на стороне сервера (наиболее частые запросы, разметка и обращения), так и на стороне пользователя (конкретизированные выборки данных).

Преимущества использования подхода:

* Снижение времени обработки запросов
* Снижение вычислительной нагрузки на сервис

Недостатки:

* Увеличение трудоемкости разработки
* Требовательность к объему памяти сервера
* Увеличение нагрузки на клиентский браузер

При создании сервиса, на базе AJAX создана система валидации формы отправки запроса на инициализацию контроллера и системы аутентификации на стороне пользователя.(можно вставить рисуночек)

# 4.2.3 Минификация кода

Под минификацией кода, в контексте данной работы, подразумеваются методы сжатия и объединения frontend-части приложения. Как было обозначено ранее, код разметки и обработки действий является неотъемлемой частью трафика клиент-серверного обмена, а часто занимает большую часть, и если трафик с годами снизил ресурсную стоимость, то требования к скорости работы сервисов сильно возросли, поэтому необходимость минимизации объема информационного обмена все ещё актуальна.

В качестве инструмента минификации была выбрана утилита YUI-compressor с последующим архивированием GZIP.

Достоинства инструментария:

* Открытость проекта
* Результативность компрессии

Принципы данной минификации основаны на уменьшении объема кода (удаление пробелов, комментариев, переносов строки), оптимизации сборок (удаление неиспользуемых библиотек и частей библиотек), оптимизации названий (сокращение названий функций и переменных) а так же методы компрессии и декомпрессии.

При реализации приложений нужно обосновать необходимость минификации, часто скорость обмена с сервером не перекрывает затраты на сжатие/декомпрессию, так же следует учитывать вероятность ошибок при оптимизации кода.

Достоинства использования минификации

* Снижение объема кода
* Увеличение скорости клиент-серверного обмена

Недостатки

* Вероятность искажения разметки и методов
* Трудность трассировки ошибок Frontend-части на стендах

# 5 Рейтинг Эло

# 5.1 Возникновение алгоритма

Работа контроллера CalculationModule являет собой реализацию алгоритма рейтингования Эло, разработанного американским физиком и шахматистом Апардом Эло.

Алгоритм был предложен в 1960-е годы международной шахматной федерации ФИДЕ, в качестве метода расчета относительной силы игроков в соревновательных видах спорта.

# 5.2 Вычисление рейтинга Эло

Все новоприбывшие субъекты системы получают начальный рейтинг равный 1400. При расчете рейтинга Эло, применяется вычисление математического ожидания количества очков, которое предположительно наберёт первый игрок при игре со вторым игроком (исходя из модели, математическое ожидание равно сумме вероятности выигрыша первого игрока и половины вероятности ничьей):

Формула 5.2.1 – Вычисление математического ожидания

где:

* Ea - математическое ожидание количества очков, которое наберёт игрок A в партии с игроком B;
* Ra - рейтинг игрока A;
* Rb - рейтинг игрока B.

После расчета математического ожидания, вычисляется новый рейтинг игрока A согласно формуле:

Формула 5.2.2 – Вычисление рейтинга филиала

где:

* K — коэффициент изменения рейтинга;
* Sa - набранное игроком A количество очков;
* R`a - новый рейтинг игрока A.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Текущий рейтинг игрока | >2400 | 2000<x<2400 | x<2000 | Первые 10 игр |
| Значение коэффициента | 10 | 20 | 30 | 40 |

Таблица 5.2.1 - Зависимость коэффициентов от текущего рейтинга

|  |  |  |
| --- | --- | --- |
| Победа | Ничья | Поражение |
| 1 | 0.5 | 0 |

Таблица 5.2.2 – Набираемые в ходе игры очки

Следует отметить, что в отличие от спортивных состязаний, при реализации продукта, филиалы находятся в заведомо не равных условиях: разнятся объемы населения и специфика регионов, поэтому, появилась необходимость в усовершенствовании алгоритма расчетов. В качестве модификации алгоритма, была предложена спецификация, состоящая в нормировке набранных филиалами очков.

Таким образом, Sa из формулы х рассчитывается:

*Формула 5.2.3 – Вычисление количества очков*

Где:

* S`a – итоговое количество очков;
* Sa – очки, набранные игроком A;
* Sb – очки, набранные игроком B;

Это позволило нивелировать разницу в условиях филиалов, уйти от факта большего числа продаж к их доле в соревновании, и в конечном счете преобразовать систему расчета рейтинга к разности математического ожидания и доли продаж в состязании.

В контексте данной работы, в качестве очков, полученных филиалами, выступает прибыль компании от реализации продуктов.(рост, система)

# 6. Алгоритм Ярроу

# 6.1 Компоненты алгоритма Ярроу

Алгоритм Ярроу был разработан в 1999 году Брюсом Шнайнером, Джоном Келси и Нилсом Фергусом, разработчиками из компании по обеспечению WEB-безопасности.

Алгоритм является криптографически-стойким генератором псевдослучайных чисел с высоким показателем энтропии. В своей основе он несет четыре независимых компонента:

* Накопитель энтропии.
* Механизм усложнения.
* Механизм генерации.
* Механизм управления усложнением.

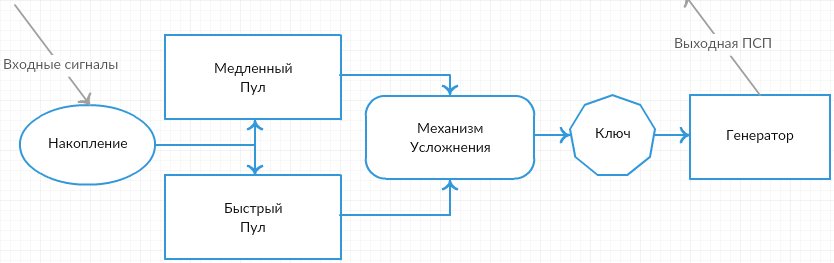


Рисунок 6.1.1 – Схема ГПЧ

# 6.1.1 Накопитель энтропии

Накопление энтропии — это процесс изменения внутреннего состояния ГСПЧ. В данном алгоритме энтропия накапливается в двух пулах: быстром и медленном. Быстрый пул обеспечивает частые усложнения ключа. Это гарантирует, что компрометация ключа имеет невысокую продолжительность. Медленный пул обеспечивает редкие, но существенные усложнения ключа, для обеспечения безопасного усложнения в случаях низкой энтропии.

# 6.1.2 Механизм усложнения

Механизм усложнения, используя информацию из пулов, обновляет ключ, который используется механизмом генерации. Таким образом, если злоумышленник не знает текущий ключ или пулы, то ключ будет ему неизвестен после усложнения. Чтобы сгенерировать новый ключ, усложнение из быстрого пула использует текущий ключ и хеши всех входов быстрого пула с момента последнего усложнения ключа. Усложнение из медленного пула использует текущий ключ и хеши входов быстрого и медленного пулов.

# 6.1.3 Механизм генерации

Механизм генерации, в качестве выходной последовательности дает битовый набор – псевдослучайных последовательность чисел. Она должна отвечать главному требованию - не знающий ключа генератора, не должен отличить выходную последовательность от случайной последовательности бит.

Для этого механизм генерации должен обладать следующими свойствами:

* стойкостью к криптографическим атакам;
* производительностью;
* способностью генерировать очень длинную последовательность сигналов без усложнения;
* стойкостью к атакам перебором с возвратом.

# 6.1.4 Механизм управления усложнением

Механизм управления усложнением отвечает за смену ключа. Слишком частая смена, делает более вероятной атаку итеративного перебора, медленная - дает больше информации злоумышленнику, получившему ключ. При поступлении выборок в пулы, сохраняются оценки энтропии для каждого источника. Как только оценка источника достигает предельного значения, происходит усложнение из быстрого пула. Усложнение из медленного пула происходит, когда оценки для любых k из n источников в медленном пуле превышают пороговую отметку.

# 7 Список использованных источников

1. Адам Фримен, Джозеф C. Раттц-мл. LINQ: язык интегрированных запросов в C# 2010 для профессионалов = Pro LINQ: Language Integrated Query in C# 2010. — М.: [«Вильямс»](https://ru.wikipedia.org/w/index.php?title=%D0%92%D0%B8%D0%BB%D1%8C%D1%8F%D0%BC%D1%81_(%D0%B8%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE)&action=edit&redlink=1), 2011. — С. 656. — [ISBN 978-5-8459-1701-0](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:%D0%98%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8_%D0%BA%D0%BD%D0%B8%D0%B3/9785845917010). LINQ
2. Адам Фримен. ASP.NET MVC 4 с примерами на C# 5.0 для профессионалов, 4-е издание = Pro ASP.NET MVC 4, 4th edition. — М.: [«Вильямс»](https://ru.wikipedia.org/w/index.php?title=%D0%92%D0%B8%D0%BB%D1%8C%D1%8F%D0%BC%D1%81_(%D0%B8%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE)&action=edit&redlink=1), 2013. — 688 с. — [ISBN 978-5-8459-1867-3](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:%D0%98%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8_%D0%BA%D0%BD%D0%B8%D0%B3/9785845918673). ПАТТЕРН
3. Джесс Чедвик и др. ASP.NET MVC 4: разработка реальных веб-приложений с помощью ASP.NET MVC = Programming ASP.NET MVC 4: Developing Real-World Web Applications with ASP.NET MVC.  2013. — 432 с. — [ISBN 978-5-8459-1841-3](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:%D0%98%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8_%D0%BA%D0%BD%D0%B8%D0%B3/9785845918413). ПАТТЕРН
4. Сергей Рогачев. [Обобщённый HYPERLINK "http://rsdn.org/article/patterns/generic-mvc.xml" Model-View-Controller](http://rsdn.org/article/patterns/generic-mvc.xml) // rsdn.org. — 2007. ПАТТЕРН
5. Роберт Э. Уолтерс, Майкл Коулс. SQL Server 2008: ускоренный курс для профессионалов = Accelerated SQL Server 2008. — М.: [«Вильямс»](https://ru.wikipedia.org/w/index.php?title=%D0%92%D0%B8%D0%BB%D1%8C%D1%8F%D0%BC%D1%81_(%D0%B8%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE)&action=edit&redlink=1), 2008. — С. 768. — [ISBN 978-5-8459-1481-1](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:%D0%98%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8_%D0%BA%D0%BD%D0%B8%D0%B3/9785845914811). SQL
6. Роберт Виейра. Программирование баз данных Microsoft SQL Server 2005. Базовый курс = Beginning Microsoft SQL Server 2005 Programming 2007. — С. 832. — [ISBN 0-7645-8433-2](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:%D0%98%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8_%D0%BA%D0%BD%D0%B8%D0%B3/0764584332). SQL
7. К. Дж. Дейт. Введение в системы баз данных = Introduction to Database Systems. — 8-е изд. — М.: [Вильямс](https://ru.wikipedia.org/w/index.php?title=%D0%92%D0%B8%D0%BB%D1%8C%D1%8F%D0%BC%D1%81_(%D0%B8%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE)&action=edit&redlink=1), 2006. — С. 1328. — [ISBN 0-321-19784-4](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:%D0%98%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8_%D0%BA%D0%BD%D0%B8%D0%B3/0321197844). РЕЛЯЦИОНКА
8. Стивен Хольцнер. Ajax Библия программиста = Ajax Bible. — М.: [Диалектика](https://ru.wikipedia.org/w/index.php?title=%D0%94%D0%B8%D0%B0%D0%BB%D0%B5%D0%BA%D1%82%D0%B8%D0%BA%D0%B0_(%D0%B8%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE)&action=edit&redlink=1), 2009. — С. 553 АЯКС
9. Станек Уильям Р. Internet Information Services (IIS) 7.0. Справочник администратора. — СПб.: [Русская редакция](https://ru.wikipedia.org/w/index.php?title=%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%B0%D1%8F_%D1%80%D0%B5%D0%B4%D0%B0%D0%BA%D1%86%D0%B8%D1%8F&action=edit&redlink=1), 2009. — 528 с. — [ISBN 978-5-7502-0383-3](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:%D0%98%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8_%D0%BA%D0%BD%D0%B8%D0%B3/9785750203833). ИИС
10. Ноубл, Дж., Андерсон, Т., Брэйтуэйт, Г., Казарио, М., Третола, Р. Flex 4. Рецепты программирования. — БХВ-Петербург, 2011. — С. 548. — 720 с. ORM
11. Julia Lerman. Programming Entity Framework. — 2nd Edition. — [O’Reilly](https://ru.wikipedia.org/wiki/O%E2%80%99Reilly_Media), 2010. — 920 p. — [SBN 0-596-80726-0](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:%D0%98%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8_%D0%BA%D0%BD%D0%B8%D0%B3/0596807260). Entuty Dapper
12. Шахматы: энциклопедический словарь / гл. ред. [А. Е. Карпов](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D0%BF%D0%BE%D0%B2,_%D0%90%D0%BD%D0%B0%D1%82%D0%BE%D0%BB%D0%B8%D0%B9_%D0%95%D0%B2%D0%B3%D0%B5%D0%BD%D1%8C%D0%B5%D0%B2%D0%B8%D1%87). — М.: [Советская энциклопедия](https://ru.wikipedia.org/wiki/%D0%91%D0%BE%D0%BB%D1%8C%D1%88%D0%B0%D1%8F_%D0%A0%D0%BE%D1%81%D1%81%D0%B8%D0%B9%D1%81%D0%BA%D0%B0%D1%8F_%D1%8D%D0%BD%D1%86%D0%B8%D0%BA%D0%BB%D0%BE%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F_(%D0%B8%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE)), 1990. — С. 136. — 624 с. — 100 000 экз. — [ISBN 5-85270-005-3](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:%D0%98%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8_%D0%BA%D0%BD%D0%B8%D0%B3/5852700053). ЭЛО
13. Фергюсон, Н., Шнайер, Б. Практическая криптография. — Издательский дом “Вильямс”, 2004. — 432 с. — ISBN 5–8459–0733–0. ЯРРОУ