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 пользу 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 затем 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.

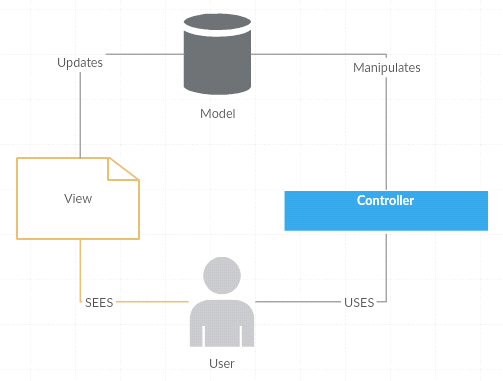
Выполнив пост-обработку выходных рейтингов, посредством нормирования, можно с высокой точностью распределить заложенный ресурс между сегментами, на него претендующими.

При реализации придержемся тенденции перевода всех ресурсов в web-среду, продиктовано это стремлением увеличить мобильность своих сервисов, приложений, развить устойчивость систем, обеспечить кроссплатформенность и значительно снизить издержки компании во всем «жизненном цикле» продукта.

В данной работе, в качестве объекта рассмотрены отделы реализации продуктов страховой компании “Югория”.

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.



Компоненты:

* Модель представляет совокупность данных приложения и их описание, 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:

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

Достоинства:

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

МОДЕЛЬ

Х. 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.

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

X Схема базы данных.

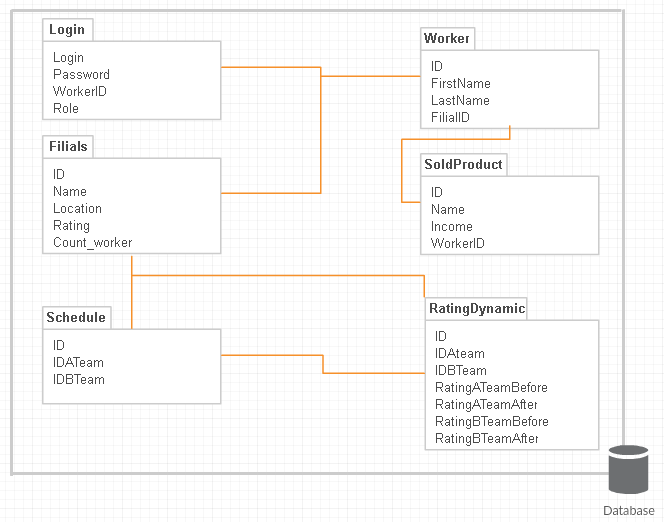
Таблица Филиалов компании(Filials) содержит: идентификационный номер филиала, его наименование, локацию, текущий рейтинг, количество работников.

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.

X.N Хранение данных

В качестве СУБД, используемой для административной работы с данными, применяется инструмент Microsoft SQL Server 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

Х. 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

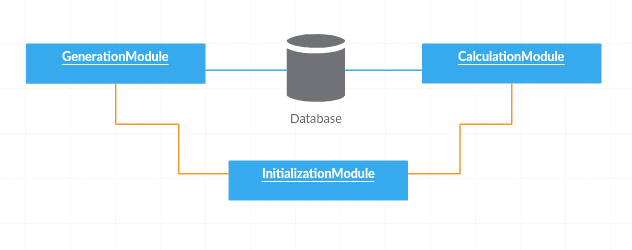
Этап обращения к базе данных считается узким местом во всех WEB-проектах. При работе с базой данных, lINQ-выражения проходят этап интерпретации, что само по себе занимает часть вычислительного и временного ресурсов, также обстоит дело с генерируемым кодом – часто он не является оптимальным. Поэтому при 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.

КОНТРОЛЛЕР

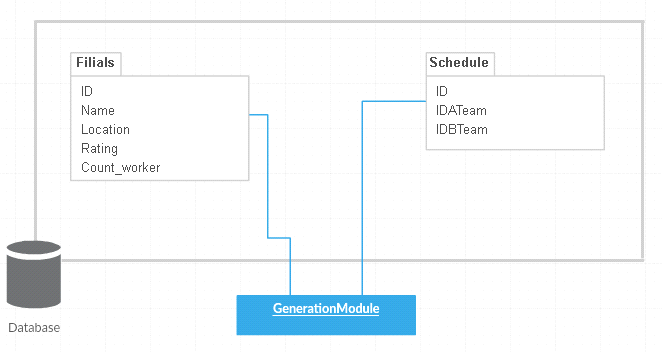
N.X Схема управляющих сигналов.

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

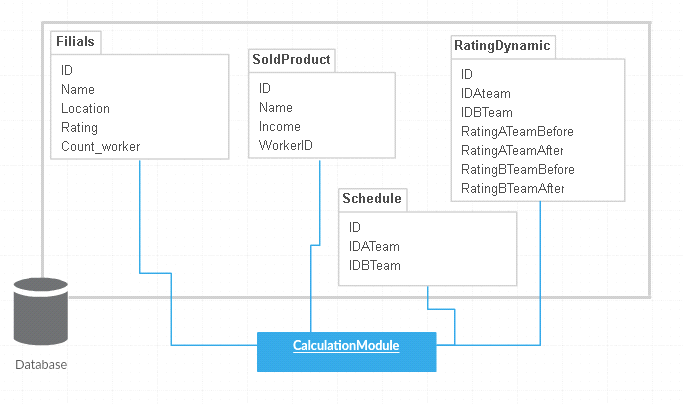


Блок инициализации включает в себя контроллер системного времени, отвечающий за трассировку событий (временных точек), и контроллер активации блоков генерации и расчета.

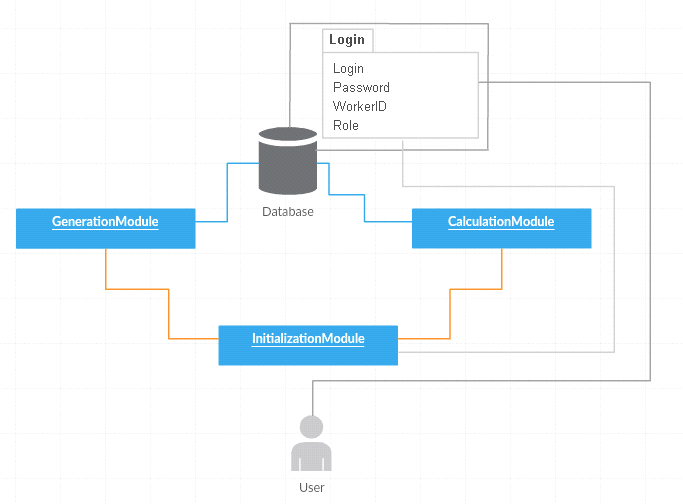
Блок генерации пар команд - алгоритм, работа которого, заключается в выборке филиалов из базы данных для создания «игрового» расписания, алгоритм базируется на псевдослучайной основе, для наработки практического опыта был взят алгоритм Ярроу.



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.

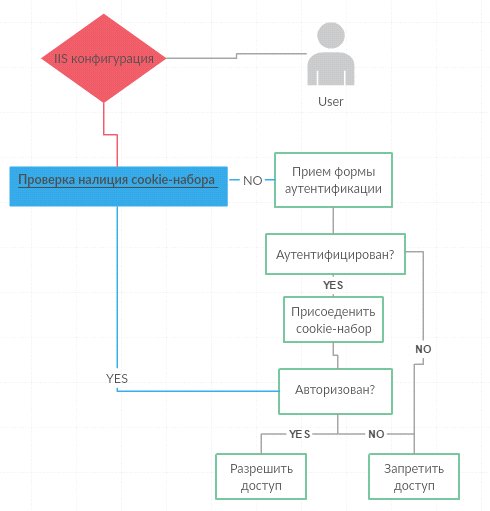


Данная система существует в автономном режиме, но так же существует инструмент ручного обращения к системе, отсюда вытекает необходимость создания системы аутентификации пользователя инициализирующего контроллер.



Аутентификация в системе

X.N Схема авторизации



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

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.

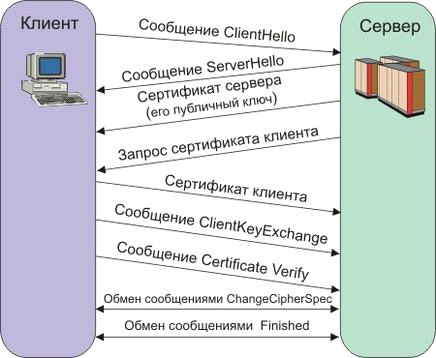
Но есть вероятность, что злоумышленник перехватит незашифрованный трафик, извлечет из него cookie-набор (который уже зашифрован) и воспользуется им в своих целях. Расшифровывать его ему не понадобится - достаточно просто снабдить им собственные пересылаемые запросы. Противостоять такой атаке повтором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

Х. Схема работы 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.

Х. ПРЕДСТАВЛЕНИЕ

Х. 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.

Х. Методы оптимизации представления

Х. AJAX

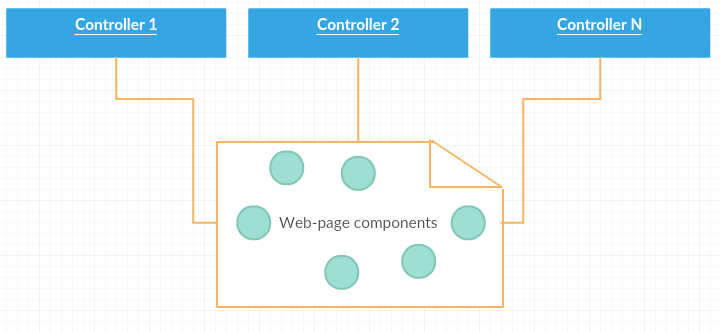
Для уменьшения серверной нагрузки был реализован подход к построению интерактивных пользовательских интерфейсов веб-приложений AJAX(Asynchronous JavaScript and XML), подход фонового клиент-серверного обмена данными без полной перегрузки веб-страницы.

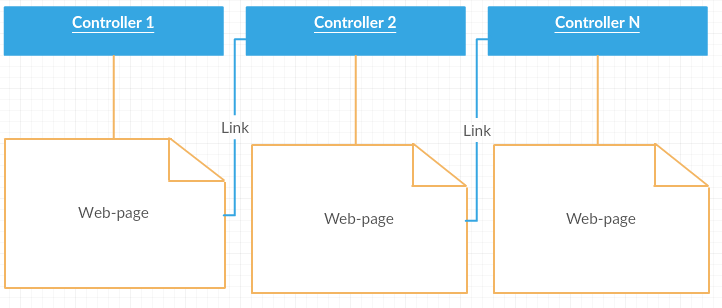
Преимущества использования:

* 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





Х. Кэшированние

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.

Х. 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.

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 GZIP

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

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

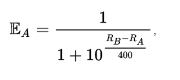
* Вероятность искажения разметки и методов

Математические выкладки

Работа контроллеров IntializatioModule и GenerationModule являет собой реализацию алгоритма рейтингования Эло и алгоритма генерации Ярроу.

Х. Вычисление рейтинга Эло

Для получения рейтинга Эло применяется вычисление математического ожидания количества очков, которое наберёт первый игрок в партии со вторым игроком (исходя из модели, математическое ожидание равно сумме вероятности выигрыша первого игрока и половины вероятности ничьей):



где:

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

Новый рейтинг игрока A рассчитывается по формуле:



где:

* K — коэффициент, значение которого равно 10 для сильнейших игроков (рейтинг 2400 и выше), 20 - для игроков с рейтингом меньше чем 2400 и 40 - для новых игроков (первая игра);
* Sa - фактически набранное игроком A количество очков (1 очко за победу, 0,5 — за ничью и 0 — за поражение);
* R`a - новый рейтинг игрока A.

Модификация данного алгоритма состоит в том, что для оптимизации необходимо набранные филиалами очки подвергать нормировке, что позволит

1.1 Компоненты алгоритма Ярроу.

Алгоритм Ярроу является композиций из 4 независимых частей:

Накопитель энтропии.

Механизм усложнения.

Механизм генерации.

Механизм управления усложнением.

Накопитель энтропии

Накопление энтропии (англ. entropy accumulation) — это процесс, при котором ГПСЧ получает новое неугадываемое внутреннее состояние[8]. В данном алгоритме энтропия накапливается в двух пулах (англ.)русск.: быстром и медленном[9]. В данном контексте под пулом понимается хранилище инициализированных и готовых к использованию битов. Быстрый пул обеспечивает частые усложнения ключа. Это гарантирует, что компрометация ключа имеет невысокую продолжительность. Медленный пул обеспечивает редкие, но существенные усложнения ключа. Это необходимо для того, чтобы гарантировать получение безопасного усложнения даже в тех случаях, когда оценки энтропии сильно завышены. Входные выборки попеременно посылаются в быстрый и медленный пулы.

Механизм усложнения

Механизм усложнения соединяет накопитель энтропии с механизмом генерации. Когда механизм управления усложнением определяет, что усложнение необходимо, то механизм усложнения, используя информацию из одного или сразу двух пулов (англ.)русск., обновляет ключ, который используется механизмом генерации. Таким образом, если злоумышленник не знает текущий ключ или пулы, то ключ будет ему неизвестен после усложнения. Также, возможно, усложнению потребуется большое количество ресурсов, чтобы минимизировать успех атаки на основе угадывания входных значений.

Чтобы сгенерировать новый ключ, усложнение из быстрого пула использует текущий ключ и хеши всех входов быстрого пула с момента последнего усложнения ключа. Как только это будет выполнено, оценки энтропии (англ.)русск. для быстрого пула обнулятся.

Усложнение из медленного пула использует текущий ключ и хеши входов быстрого и медленного пулов. После генерирования нового ключа оценки энтропии для обоих пулов сбрасываются в ноль[13].

Механизм генерации

Механизм генерации дает на выходе последовательность псевдослучайных чисел. Она должна быть такой, чтобы злоумышленник, не знающий ключа генератора, не смог отличить ее от случайной последовательности бит.

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 (англ. Backtracking) после компрометации ключа.

Механизм управления усложнением

Для того, чтобы выбрать время усложнения, механизм управления должен учесть различные факторы. К примеру, слишком частая смена ключа делает более вероятной атаку итеративного угадывания. Слишком медленная, напротив, дает больше информации злоумышленнику, скомпрометировавшему ключ. Поэтому механизм управления должен уметь находить золотую середину между этими двумя условиями.

По мере поступления выборок в каждый пул, сохраняются оценки энтропии для каждого источника. Как только эта оценка для любого источника достигает предельного значения, происходит усложнение из быстрого пула. В подавляющей части систем это случается множество раз в час. Усложнение из медленного пула происходит, когда оценки для любых {\displaystyle k} k из {\displaystyle n} n источников в медленном пуле превышают пороговую отметку.

1.2 Рейтинг Эло

1.2.1

1.2.2 Вычисление рейтинга Эло

Вычисляется математическое ожидание количества очков, которое наберёт игрок A в партии с B (оно равно сумме вероятности выигрыша игрока A и половины вероятности ничьей):

{\displaystyle \mathbb {E} \_{A}={\frac {1}{1+10^{\frac {R\_{B}-R\_{A}}{400}}}}} {\mathbb {E}}\_{A}={\frac 1{1+10^{{{\frac {R\_{B}-R\_{A}}{400}}}}}},

где:

{\displaystyle \mathbb {E} \_{A}} {\displaystyle \mathbb {E} \_{A}} — математическое ожидание количества очков, которое наберёт игрок {\displaystyle A} A в партии с {\displaystyle B} B;

{\displaystyle R\_{A}} R\_{A} — рейтинг игрока {\displaystyle A} A;

{\displaystyle R\_{B}} R\_{B} — рейтинг игрока {\displaystyle B} B.

Новый рейтинг игрока {\displaystyle A} A рассчитывается по формуле:

{\displaystyle R\_{A}^{\prime }=R\_{A}+K\cdot (S\_{A}-\mathbb {E} \_{A})} R\_{A}^{\prime }=R\_{A}+K\cdot (S\_{A}-{\mathbb {E}}\_{A}),

где:

{\displaystyle K} K — коэффициент, значение которого равно 10 для сильнейших игроков (рейтинг 2400 и выше), 20 (было 15) — для игроков с рейтингом меньше чем 2400 и 40 (было 30) — для новых игроков (первые 30 партий с момента получения рейтинга ФИДЕ), а также для игроков до 18 лет, рейтинг которых ниже 2300;

{\displaystyle S\_{A}} S\_{A} — фактически набранное игроком {\displaystyle A} A количество очков (1 очко за победу, 0,5 — за ничью и 0 — за поражение);

{\displaystyle R\_{A}^{\prime }} {\displaystyle R\_{A}^{\prime }} — новый рейтинг игрока {\displaystyle A} A.

СРАНЕНИЕ ОТЕЧЕСТВЕННЫХ РЕШЕНИЙ

В данный момент, большая часть, включая крупные, отечественные компании не ведут рейтинговый учет своих реализующих отделений, соответственно отсутствуют меры справедливого поощрения и наказания, не стимулируются отделения с высокой (относительно других) продуктивностью работы. (незакончено)

Литература

* *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
* *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
* *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. — М.: [« HYPERLINK "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"Вильямс HYPERLINK "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"»](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. — 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
* *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
* *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
* *Роберт Виейра.* Программирование баз данных Microsoft SQL Server 2005. Базовый курс = Beginning Microsoft SQL Server 2005 Programming. — М.: [« HYPERLINK "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"Диалектика HYPERLINK "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"»](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), 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
* *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
* Максим Никитин. [Закончилась ли эпоха реляционных СУБД?](http://www.cnews.ru/reviews/free/marketBD/articles/articles2.shtml) // CNews.ru, 2010 г. ССЫЛКА НА СТАТИСТИКУ
* *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. — [ISBN HYPERLINK "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%B3/9785845915023" 978-5-8459-1502-3](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%B3/9785845915023). АЯКС
* *Ноубл, Дж., Андерсон, Т., Брэйтуэйт, Г., Казарио, М., Третола, Р.* Flex 4. Рецепты программирования. — БХВ-Петербург, 2011. — С. 548. — 720 с. ORM
  + 1. *Julia Lerman.* Programming Entity Framework. — 2nd Edition. — [O’Reilly](https://ru.wikipedia.org/wiki/O%E2%80%99Reilly_Media), 2010. — 920 p. —[I 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