Правительство Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Санкт-Петербургский государственный университет»

Программная инженерия

Евтушенко Владислав Валерьевич

Создание информационной системы для управления рисками предприятия

Выпускная квалификационная работа

Научный руководитель:
ст. преп. Немешев М.Х.

Рецензент:
техн. дир. ООО «Системы КМ» Петров А.Г.

Санкт-Петербург
2017

SAINT-PETERSBURG STATE UNIVERSITY

Software engineering

Evtushenko Vladislav

Creating information system for risk management of enterprise

Graduation Thesis

Scientific supervisor:
assistant M.H. Nemeshev

Reviewer:
CEO of «Systems CM» A.G. Petrov

Saint-Petersburg
2017

Оглавление

Введение

Существуют предприятия, управление которыми регулируется международными стандартами ISO[1]. Применение этих стандартов обеспечивает уверенность потребителей, инвесторов, общества и других заинтересованных лиц в том, что организация способна выполнить как условия контракта, так и применяемые законодательные и другие нормативные требования.

Стандарты для систем менеджмента развиваются, и возникает проблема перехода на их новые версии. В 2015 году часть стандартов, в том числе ISO 9001:2015 и ISO 14001:2015, стала включать в себя новое требование – внедрение риск-ориентированного подхода в управление предприятием [1,2]. Это дополнительно увеличило спрос на системы для управления рисками.

Существует большое разнообразие систем для управления рисками: начиная от структурированных документов Microsoft Office и заканчивая крупными программными комплексами компании SAP[2].

Но в то же время, у существующих решений можно выделить две проблемы. Они относительно дорогие по причине большого набора функций, что не подходит для небольших компаний с ограниченным бюджетом. Также немалая часть решений рассматривает именно финансовые риски, что может быть неактуально для ряда отраслевых специалистов, например, экологов.

В компании «Системы КМ»[3], занимающейся в том числе и услугами по внедрению управления рисками на предприятиях, имеется свой подход и основанный на нем прототип информационной системы. Прототип реализует общие принципы управления рисками, но не обладает всей функциональностью, необходимой конечным пользователям. Данная работа ведется в рамках создания информационной системы на базе этого прототипа и с учетом недостатков существующих решений.

[1] Международная организация по стандартизации (ISO), URL: https://www.iso.org (дата обращения 17.05.2017)
[2] Программный продукт Risk Management компании SAP, URL: https://www.sap.com/products/risk-management.html (дата обращения 17.05.2017)
[3] Компания «Системы КМ», URL: http://systemskm.ru (дата обращения 17.05.2017)

1. Постановка задачи

Целью данной бакалаврской работы является создание информационной системы для управления рисками предприятия.

Для достижения этой цели был сформулирован следующий набор задач.

- Анализ предметной области и существующих решений.
- Разработка архитектуры системы для управления рисками предприятия.
- Реализация системы для управления рисками предприятия.
- Проведение миграции данных из предыдущего решения.

2. Предметная область

2.1. Обзор существующих решений

Перед решением поставленной задачи был произведен анализ предметной области и проанализированы существующие решения. Цель обзора заключалась в рассмотрении сильных и слабых сторон существующих решений, а также в изучении возможностей взять какое-либо из открытых решений за основу разрабатываемой в рамках работы системы.

Некоторые из этих решений будут рассмотрены далее.

2.1.1. SAP Risk Management

SAP Risk Management – программа для управления рисками, которая позволяет согласовывать управление рисками с факторами ценности бизнеса, получать надежную информацию о возникновении рисков, реагировать на возникающие риски и возможности, прогнозировать последствия незапланированных событий [3]. С программой поставляется ряд вспомогательных инструментов, в том числе набор для построения графиков. Пример пользовательского интерфейса SAP Risk Management показан на рис. 1.

Для своей работы программа требует операционную систему семейства Windows и веб-браузер Microsoft Internet Explorer. Существуют десктопная и веб-версии программы.

В этой программе не было найдено каких-либо слабых сторон, кроме цены, которая справедливо обосновывается большой функциональностью, гарантией качества и другими положительными факторами.

Стоимость программы начинается от нескольких тысяч долларов в год и вычисляется индивидуально для каждого предприятия-клиента. Также компания SAP за отдельную плату предлагает обучающие семинары по использованию своих продуктов.

Схожие по цене и функциональности решения есть и у ряда других компаний, например, SAI Global[4].

[4] Компания SAI Global, URL: https://www.saiglobal.com (дата обращения 17.05.2017)

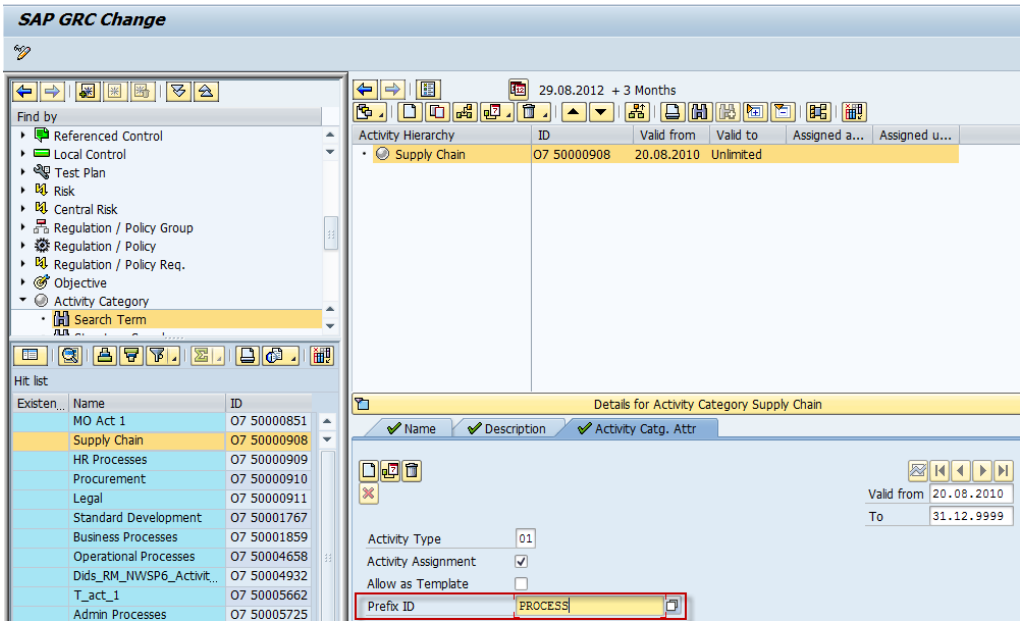

Рис. 1: SAP Risk Management – Управление действиями, связанными с рисками

2.1.2. Eramba

Eramba[5] – программа для управления рисками с широким набором функций.

Имеются две версии программы. Первая бесплатная и с открытым исходным кодом, но с ограниченным набором функций, вторая версия обладает всеми доступными функциями и распространяется по платной подписке.

Программа создана с помощью Apache HTTP Server, PHP, MySQL и ряда других технологий. Для работы требуется операционная система семейства Linux или другие, позволяющие запустить Linux внутри виртуальной машины.

Взять за основу версию Eramba с открытым исходным было нецелесообразно по следующим причинам:

- Пользовательский интерфейс программы не соответствует требованиям, предъявляемым к разрабатываемому в рамках этой работы решению. Приведение интерфейса к необходимому виду потребовало бы полной его переработки и привело бы к слишком большим издержкам.
- Несмотря на наличие документации, Eramba имеет множество функций и модулей, крупное изменение или создание новых компонентов требует слишком много времени и ресурсов. Изменения та-

[5] Программа Eramba, URL: http://www.eramba.org (дата обращения 17.05.2017)

кого уровня обычно делают сами разработчики программы в рамках платной подписки.

2.1.3. Структурированные документы и таблицы

Для управления рисками можно использовать структурированные документы и таблицы, работать над которыми можно совместно. Существует большое количество офисных приложений, таких как Microsoft Office, Google Docs/Google Sheets, LibreOffice, для которых можно найти готовые к использованию шаблоны.

Сильные стороны такого решения: привычный многим пользователям интерфейс и относительная простота – не нужно использовать отдельный сервис, в котором потребуется разобраться.

Слабой стороной является ограниченная функциональность - нельзя выйти за рамки приложения для работы с таблицами или текстом.

Такой подход к управлению рисками широко распространен.

2.1.4. Выводы

После проведения анализа существующих решений были сделаны следующие выводы.

- Большинство решений в рамках предметной области проприетарные. Существует большое разнообразие как по цене, так и по функциональности.
- Проприетарные решения, обладающие достаточной функциональностью и удобным пользовательским интерфейсом, имеют неприемлемую для ряда компаний стоимость.
- Не было найдено решений с открытым исходным кодом, которые целесообразно было бы взять за основу создаваемой в рамках работы системы.

2.2. Подход к управлению рисками предприятия

Существует множество подходов к управлению рисками компаний, также существуют и стандарты, касающиеся этой деятельности, например, семейство стандартов ISO 31000[6]. Далее будет описан упрощенный процесс управления рисками предприятия, на основе которого разрабатывается описываемая в работе система. Упрощенная схема этого процесса изображена на рис. 3.

[6] Международный стандарт ISO 30001 – Risk Management, URL: https://www.iso.org/iso-31000-risk-management.html (дата обращения 17.05.2017)

Управление рисками начинается со структурирования информации о деятельности предприятия. Ставятся цели, далее в рамках каждой цели рассматриваются производственные процессы, которые влияют на ее достижение. Процессы могут раскладываться на операции, операции на производственные аспекты. Для каждого аспекта определяется диапазон его нормального значения и уровень приоритета. Затем рассматриваются меры управления этими аспектами, которые обеспечивают нахождение аспектов в нормальном диапазоне; определяются последствия невыполнения этих мер. Риск будет вычисляться, в том числе, по уровням приоритета аспектов и значимости последствий невыполнения мер управления этим аспектом. Процесс структурирования информации в нотации UML диаграмм классов показан на рис. 2.

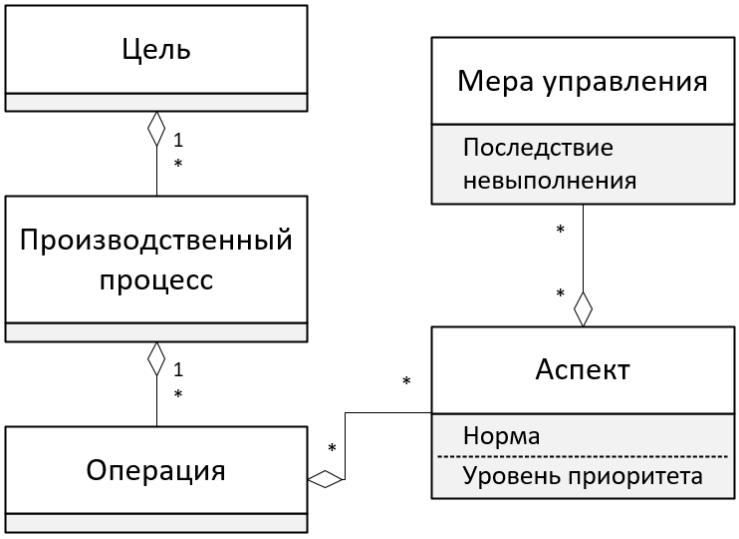

Рис. 2: Структурирование информации о деятельности предприятия

После структурирования информации о деятельности компании, теоретические риски вычисляются автоматически. Далее проводятся внутренние проверки оценки эффективности применяемых на предприятии мер управления. После чего вычисляется реальный риск. Затем возможно получить отчеты по рискам и на основе их разработать программу антирисковых мероприятий. После применения подобных мер возможно потребуется актуализировать информацию о деятельности предприятия и далее в той же последовательности продолжать процесс управления рисками.

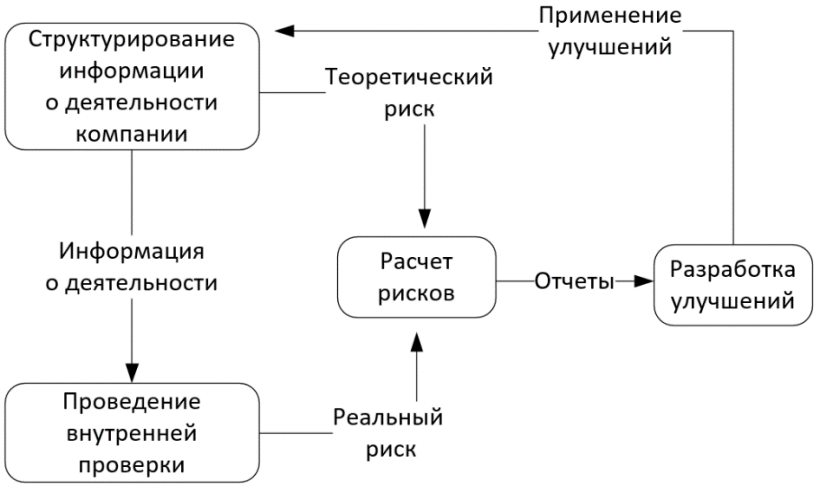

Рис. 3: Процесс управления рисками (упрощенный)

3. Разработанное решение

3.1. Требования

Для разрабатываемого в рамках работы решения было составлено техническое задание, которое, в том числе, включает в себя функциональные и нефункциональные требования.

Функциональные требования.

- Возможность управления рисками с помощью веб-приложения.
- Возможность проведения внутренних проверок с помощью Android приложения.
- Возможность синхронизации данных между Android приложением и веб-приложением.
- Возможность генерации отчетных документов по заданным шаблонам.
- Авторизация и аутентификация одного пользователя.
- Поддержка многих пользователей и их ролей в системе на уровне архитектуры.

Нефункциональные требования.

- Работа с данными, собранными до внедрения системы в эксплуатацию.
- Работа мобильного приложения при возможном отсутствии подключения к локальной сети или сети Интернет.

3.2. Архитектура системы

Система состоит из серверной, двух клиентских и одной сервисной частей, как показано на рис. 4.

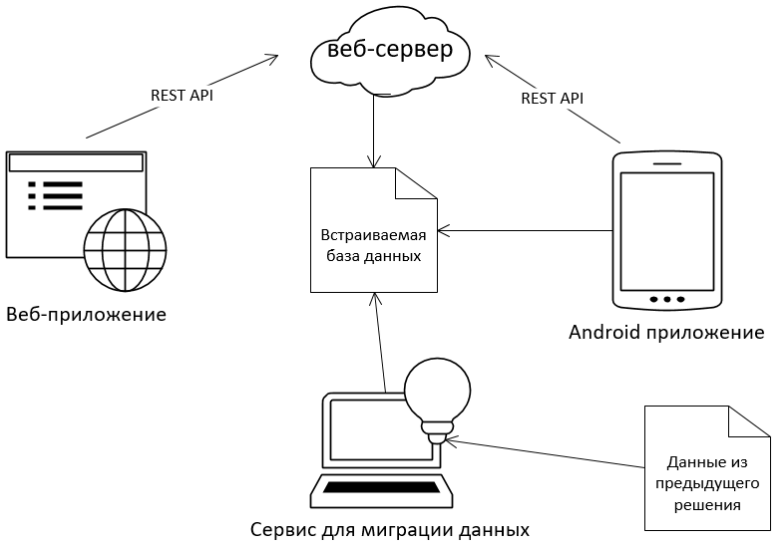

Рис. 4: Общая архитектура системы

В зависимости от запроса мобильного или веб приложения, серверная часть выдает данные, сериализованные в формате JSON, сообщения о возможных ошибках или файлы. Во время обработки запросов веб-сервер записывает/считывает данные из встраиваемой базы данных, если это необходимо. Серверная часть может размещаться в локальной сети предприятия или в облачном сервисе. Для коммуникации с клиентами по протоколу HTTP или HTTPS веб-сервер имеет интерфейс прикладного программирования (API), который создан в соответствии с архитектурным стилем REST[7]. Выбор протокола между HTTP и HTTPS обуславливается средой размещения веб-сервера, например, в случае закрытой локальной сети использование HTTPS будет излишним.

Веб-приложение является программой, выполняемой в веб-браузере пользователя. Для сохранения и загрузки данных, а также для получения ключа авторизации посылается соответствующий Ajax запрос на сервер с сериализованными в формате JSON данными. Для авторизации к HTTP/HTTPS запросам добавляется заголовок, содержащий ключ авторизации веб-приложения.

Мобильное приложение работает под управлением операционной системы Android. Для отправки результатов проведенной проверки (синхронизации данных в одну сторону), а также для получения ключа авторизации посылается HTTP или HTTPS запрос на сервер, содержащий сериализованные в формате JSON данные. Актуальная версия базы данных скачивается с веб-сервера по

[7] Архитектурный стиль Representational State Transfer (REST), впервые был описан Реем Томасом, URL: http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (дата обращения 17.05.2017)

протоколу HTTP или HTTPS (синхронизация данных в другую сторону). Для авторизации к запросам добавляется заголовок, содержащий ключ авторизации Android приложения.

Сервис для миграции данных не входит в конечное решение для пользователя, но является неотъемлемым инструментом во время разработки и внедрения системы в эксплуатацию. Роль сервиса заключается в переносе данных из предыдущего решения в новое, разрабатываемое в рамках работы.

3.3. Компоненты

3.3.1. Веб-сервер

Используемые технологии

Для реализации веб-сервера было решено использовать следующие технологии.

Основным языком программирования был выбран C# [4], он удобен и надежен для создания серверных частей веб-приложений, для него существует множество полезных инструментов, большое количество информационных источников и одна из лучших документаций MSDN[8].

В качестве веб-фреймворка используется ASP NET Core[9] [5]. Эта технология появилась в 2015 году на базе ASP NET. Ключевые отличия: кроссплатформенность и открытый исходный код, а также ряд полезных и удобных нововведений. Для механизмов аутентификации и авторизации была выбрана технология JWT[10][6], которая поддерживается ASP NET Core.

Для размещения веб-приложения возможно использование IIS[11] в случае операционной системы семейства Windows, Nginx[12] в случае Linux, а также программного обеспечения Docker[13].

---

[8] Microsoft Developer Network (MSDN) – ресурс Microsoft для разработчиков, содержащий множество информации о продуктах компании, URL: https://msdn.microsoft.com (дата обращения 17.05.2017)
[9] ASP NET Core – фреймворк для разработки от компании Microsoft, URL: https://docs.microsoft.com/en-us/aspnet/core (дата обращения 17.05.2017)
[10] JSON Web Token (JTW) – технология для генерации ключей доступа, описанная в RFC 7519, URL: https://jwt.io (дата обращения 17.05.2017)
[11] Internet Information Services (IIS) – веб-сервер от компании Microsoft, предназначенный для работы под управлением семейства ОС Windows, URL: https://www.iis.net (дата обращения 17.05.2017)
[12] Nginx – кроссплатформенный веб-сервер, оригинальным автором которого является Игорь Сысоев, URL: https://www.nginx.com (дата обращения 17.05.2017)

В качестве встраиваемой базы данных была выбрана SQLite[14][7], которая может использоваться, как и веб-сервером, так и Android приложением. Для хранения данных о пользователях, а также журналов работы веб-сервера используется система управления базами данных Microsoft SQL Server[15][8]. Для решения задач объектно-реляционного отображения используется фреймворк Entity[16] [9].

Для генерации отчетных документов в формате Microsoft Excel 2010 используется фреймворк EPPlus Core[17].

Функциональность

Веб-сервер обладает следующей функциональностью.

- Аутентификация и авторизация пользователей.
- Управление состоянием системы.
- Генерация документов в формате Microsoft Excel 2010 по заданным шаблонам.
- Синхронизация данных между мобильным приложением и веб-приложением.
- Обработка запросов на сохранение, получение и изменение данных от мобильного приложения и веб-приложения.
- Передача файлов мобильному приложению и веб-приложению
- Размещение статических файлов веб-приложения.

Веб-сервер регистрирует пользователей в системе, используя фреймворк Identity[18], входящий в состав ASP NET Core, создавая при этом новую запись в базе данных. Дальнейшая аутентификация и авторизация вносят изменения в созданную запись с помощью Identity и технологии JTW. Авторизация требуется для использования большинства функций сервера клиентами.

---

[13] Docker – платформа для запуска приложений кроссплатформенно, URL: https://www.docker.com (дата обращения 17.05.2017)
[14] SQLite – встраивая база данных с открытым исходным кодом, URL: https://www.sqlite.org (дата обращения 17.05.2017)
[15] Microsoft SQL Server – СУБД компании Microsoft, URL: https://www.microsoft.com/en-us/sql-server/sql-server-2016 (дата обращения 17.05.2017)
[16] EntityFramework – фреймворк для объектно-реляционного отображения на платформе .NET, URL: https://docs.microsoft.com/en-us/ef (дата обращения 17.05.2017)
[17] EPPlus Core – неофициальный перенос фреймворк EPPlus с платформы .NET Framework на платформу .NET Core, URL: https://github.com/VahidN/EPPlus.Core (дата обращения 17.05.2017)
[18] Indentity – часть фреймворка ASP NET Core, для авторизации и аутентификации пользователей, URL: https://www.asp.net/identity (дата обращения 17.05.2017)

Веб-сервер управляет состоянием всей системы. Всего два состояния: общее управление рисками и внутренняя проверка. На время внутренней проверки сервер запрещает изменения данных веб-приложением, оставляя только возможность просмотра.

Архитектура

Основным шаблоном проектирования был выбран Model-View-Controller [10]. В случае веб-сервера в роли View выступают модели представления данных, сериализованные в формате JSON. Для каждой функции веб-сервера, а также для каждой таблицы встраиваемой базы данных был создан отдельный контроллер. Контроллеры используют вспомогательные сервисы и модели, содержащие бизнес-логику приложения. Упрощенная диаграмма классов представлена на рис. 5.

Для работы со встраиваемой базой данных используются архитектурные приемы: Репозиторий и UnitOfWork [10]. Для работы с базой данных, содержащей данные о пользователях, используются вспомогательные классы фреймворка Indentity.

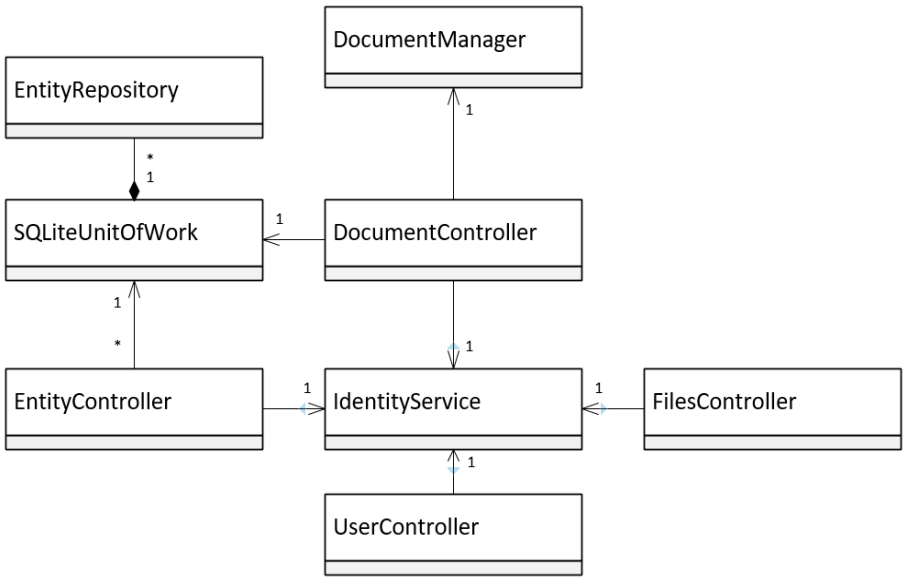

Рис. 5: Упрощенная диаграмма классов веб-сервера.

3.3.2. Сервис миграции данных

Используемые технологии

Предыдущее решение использовало для хранения данных приложение Microsoft Access[19]. Поэтому для создания сервиса миграции данных самым простым и удобным способом было продолжать использовать стек технологий Microsoft. Еще одной причиной выбора .NET является LINQ [11] - набор функций и часть синтаксиса языка C#, которые позволяют удобно работать с базами данных, используя синтаксис близкий к языку SQL.

Для реализации объектно-реляционного отображения был выбран фреймворк Entity в совокупности с библиотекой SQLiteCodeFirst[20], которая расширяет возможности первого. К сожалению, библиотека не обеспечивала желаемый тип работы с зависимыми таблицами в SQLite, а именно запрещение удаления из главной таблицы при наличии зависимых записей. Эта возможность была реализована самостоятельно.

Entity не поддерживает напрямую объектно-реляционного отображение из MicrosoftAccess, поэтому была использована свободно распространяемая утилита Microsoft SQL Server Migration Assistant[21]. Она позволила перенести данные сначала в Microsoft SQL Server, а уже затем выполнить объектно-реляционное отображение.

Проблемы и их решения

Просто перенести данные из Microsoft Access в SQLite было недостаточно по ряду причин.

- Местами была нарушена целостность данных, например, в некоторых таблицах отсутствовали первичные ключи, в других были ссылки на несуществующие записи.
- Было избыточное количество таблиц и полей.

---

[19] Microsoft Access – программа из пакета Microsoft Office, предназначенная для работами с базами данных, URL: https://products.office.com/en/access (дата обращения 17.05.2017)

[20] SQLiteCodeFirst – фреймворк для объектно-реляционного отображения в контексте базы данных SQLite, URL: https://github.com/msallin/SQLiteCodeFirst (дата обращения 17.05.2017)

[21] Microsoft SQL Server Migration Access – инструмент для миграции данных из Microsoft Access в Microsoft SQL Server, URL: https://docs.microsoft.com/en-us/sql/ssma/sql-server-migration-assistant (дата обращения 17.05.2017)

- Схема базы данных слабо отражала бизнес-логику системы в целом.

Поэтому возникла необходимость перепроектирования схемы базы данных и восстановления целостности данных.

- Новая схема проектировалась в соответствии с третьей нормальной формой.
- Новая схема соответствует бизнес-логике системы на должном уровне.
- Новая схема не включает в себя избыточные поля или таблицы.
- Целостность данных обеспечивается набором триггеров.

Архитектура

Для работы с базами данных были использованы архитектурные приемы Репозиторий и UnitOfWork. Для миграции каждой отдельной таблицы был написан ряд правил, который включил в себя класс MigrationPerformer. Для генерации скриптов создания и удаления триггеров используется класс TriggerGenerator, который, в том числе, использует класс ReflectionHelper для получения метаинформации о схеме базы данных SQLite. Архитектура в нотации UML диаграмм классов представлена на рис. 6.

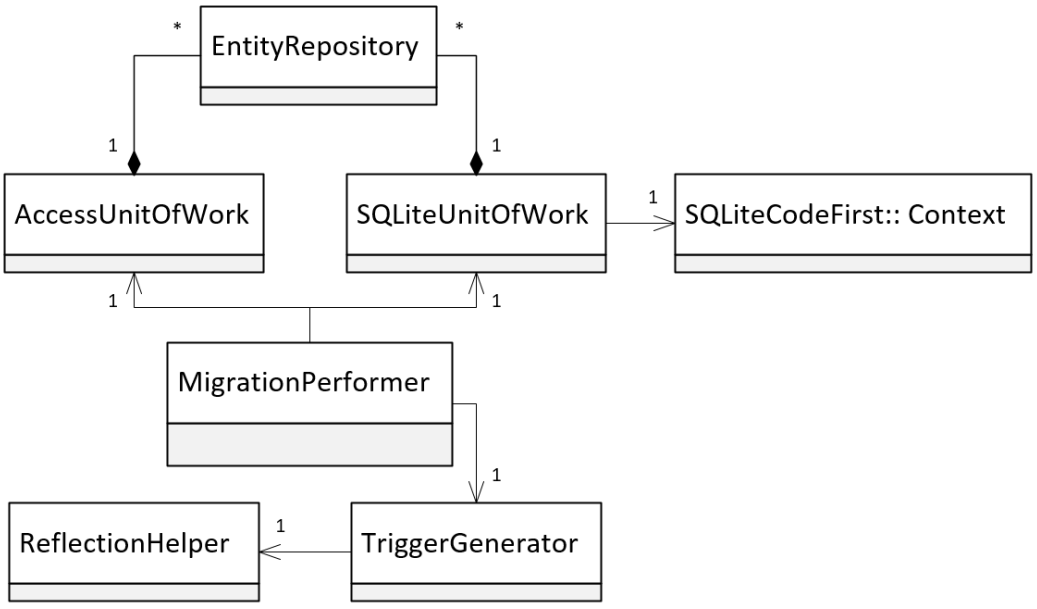

Рис. 6. Упрощенная диаграмма классов сервиса миграции данных

3.3.3. Android приложение

Используемые технологии

Основным языков программирования был выбран Kotlin[22], который обеспечивает быструю и удобную разработку и полностью совместим с Java.

Для хранения данных используются две встраиваемые базы данных SQLite. Для работы с базами данных используется объектно-реляционное отображение, которое обеспечивается фреймворком ActiveAndroid[23].

Для коммуникации с веб-сервером используются фреймворки OkHttp[24] и Retrofit 2[25], которые используют модели для каждого HTTP запроса, что помогает избежать ряда возможных ошибок, связанных с неправильным использованием типов параметров запросов.

Функциональность

Использование приложения начинается с аутентификации пользователя по средствам ввода логина и пароля. В случае успеха мобильное приложение получает ключ авторизации, который записывается в одну из двух баз данных, в ту, в которой хранится системная информация. Далее, при последующих обращениях к веб-серверу мобильное приложение будет включать ключ авторизации в заголовки запросов.

Синхронизация данных с веб-сервером происходит в две стороны. После успешной аутентификации пользователь попадает на экран синхронизации данных от сервера к мобильному приложению. В это время мобильное приложение делает асинхронный HTTP/HTTPS GET запрос к веб-серверу на проверку наличия новой версии базы данных, и в случае наличия такой скачивает ее. После проверки пользователь из пункта меню выбирает пункт «Завершить проверку» и снова попадает на экран синхронизации. Теперь мобильное приложение будет

[22] Kotlin – язык программирования, созданный и поддерживаемый компанией JetBrains, URL: https://kotlinlang.org (дата обращения 17.05.2017)
[23] ActiveAndroid – фреймворк для объектно-реляционного отображения в контексте базы данных SQLite на платформе Android, URL: http://www.activeandroid.com (дата обращения 17.05.2017)
[24] OkHTTP – фреймворк для работы с HTTP/HTTPS запросами на платформе Java (том числе Android), URL: http://square.github.io/okhttp (дата обращения 17.05.2017)
[25] OkHTTP – фреймворк для работы с HTTP/HTTPS запросами на платформе Java (том числе Android), URL: http://square.github.io/retrofit (дата обращения 17.05.2017)

асинхронно отправлять веб-серверу изменения в данных, связанных с результатами проверки в форме соответствующих POST, PUT и DELETE HTTP/HTTPS запросов. Экран синхронизации представлен на рис. 7А.

После проведения синхронизации от веб-сервера к мобильному приложению пользователь переходит к последовательности экранов-фильтров, позволяющих последовательно находить меру управления, эффективность которой оценивается во время проверки. Экран-фильтр представлен на рис. 7Б.

После выбора нужной меры управления пользователь переходит к списку найденных несоответствий, замечаний и предложений. Он может просматривать, изменять, добавлять и удалять эти сущности. В это время приложение будет асинхронно выполнять соответствующие действия над базой данных, скачанной с веб-сервера и записывать результаты действий в таблицу системной базы данных. Эти результаты будут использоваться при синхронизации данных от клиента к серверу, с их помощью система определяет, какие именно записи и как изменились. Экран добавления найденного в процессе проверки несоответствия представлен на рис. 7В.

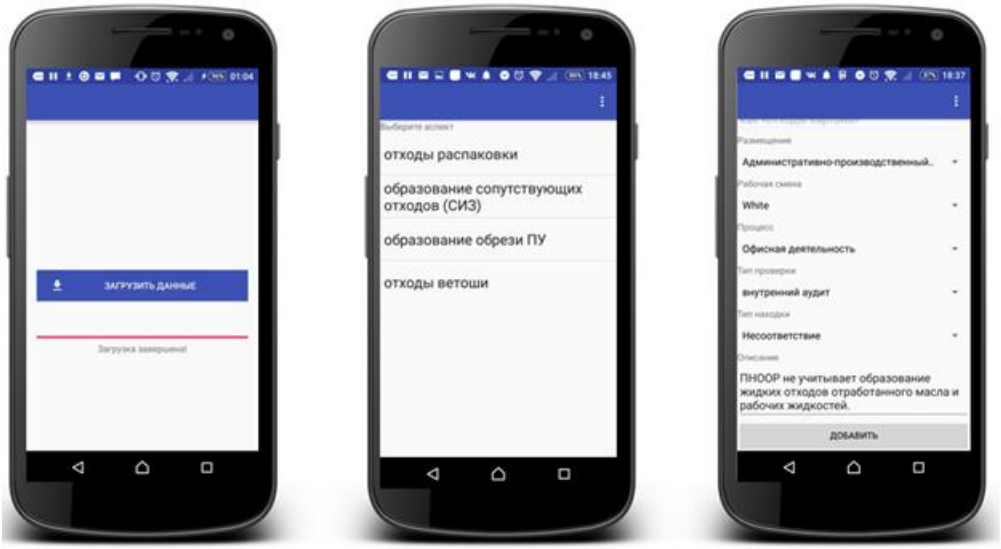

Рис 7: Пользовательские интерфейсы мобильного приложения: А – экран синхронизации данных от веб-сервера к клиенту; Б – экран фильтр, для поиска проверяемой меры управления; В – экран добавления найденного несоответствия

Архитектура

В приложении на каждый тип экрана приходится один класс Activity, всего типов экрана три: экран для синхронизации данных с веб-сервером, экран-фильтр для поиска проверяемой меры управления, экран для просмотра, изменения, удаления и добавления найденных несоответствий, замечаний и предложений по улучшениям.

За синхронизацию данных с сервером отвечают соответствующие классы, которые работают асинхронно. Класс SyncActivity коммуницирует с этими классами с помощью средства операционной системы Local Broadcast.

Работа с базой данных осуществляется через объектно-реляционное отображение таблиц баз данных в соответствующие классы. А также с применением архитектурных приемов: Репозиторий и UnitOfWork.

Упрощенная диаграмма классов в нотации UML представлена на рис. 8.

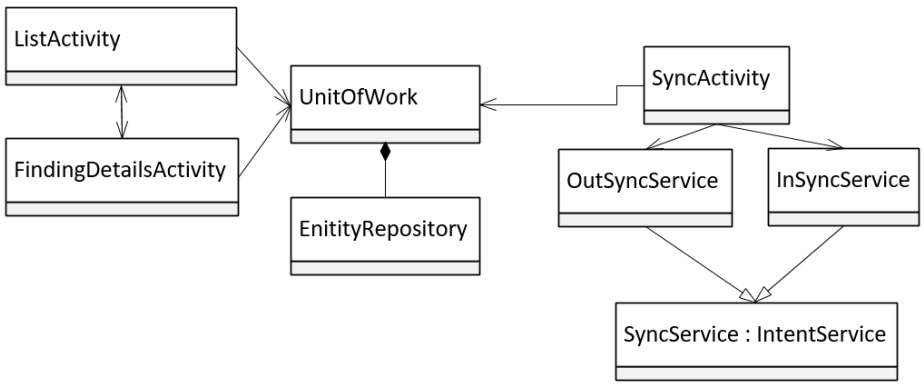

Рис. 8: Упрощенная диаграмма классов Android приложения.

3.3.4. Веб-приложение

Используемые технологии

Для создания веб-приложения в качестве основного веб-фреймворка был выбран Angular[26] [12], который позволяет удобно и быстро создавать веб-приложения.

[26] Angular – платформа для разработки мобильных и веб-приложений, URL: https://angular.io (дата обращения 17.05.2017)

В качестве основного языка программирования был выбран TypeScript[27] [13], среди преимуществ которого можно выделить статическую типизацию и поддержку Объектного Ориентированного Программирования. К тому же, он поддерживается Angular по умолчанию.

Для создания пользовательских интерфейсов был выбран фреймворк Bootstrap, позволяющий создавать довольно красивые и адаптивные веб-страницы, не особо углубляясь при этом в CSS и веб-дизайн.

Для обработки действий пользователя и коммуникации с веб-сервером используется набор библиотек RxJS[28][14], позволяющий удобно работать с событиями и асинхронными операциями.

Функциональность

Открыв веб-приложение в веб-браузере, пользователь попадает на страницу входа в приложение. В случае успешного входа, пользователь переходит к основному разделу веб-приложения. В это время веб-приложение делает запрос к веб-серверу на аутентификацию, в случае успеха, веб-приложение получает ключ авторизации, который оно хранит в Web Storage[29] и будет использовать его в дальнейшем при авторизации, добавляя его в заголовок каждого запроса к веб-серверу.

Далее пользователь имеет возможность переключаться между вкладками соответствующим производственным аспектам, мерам управления этими аспектами, найденным несоответствиям мер управления, ожидаемым результатам и антирисковыми действиями. Это изображено на рис. 9A. Для каждой из этих четырех сущностей определены: добавление, изменение и удаление (рис. 9D). В случаях добавления и изменения пользователю откроется соответствующая всплывающая форма. В это время веб-приложение асинхронно отправляет веб-серверу соответствующие запросы, содержание таблиц обновляется автомати-

[27] TypeScript – язык программирования, являющийся надмножеством JavaScript, URL: https://www.typescriptlang.org (дата обращения 17.05.2017)
[28] RxJS – библиотека для асинхронного программирования на языке JavaScript, URL: http://reactivex.io/rxjs (дата обращения 17.05.2017)
[29] Web Storage – технология, позволяющая веб-сайтами хранить их данные на стороне клиента, URL: https://www.w3.org/TR/webstorage (дата обращения 17.05.2017)

чески по мере изменения данных на сервере, используя одну из возможностей Angular – data binding[30].

У каждой из четырех сущностей есть свойства, например, у производственных аспектов есть родительское свойство – производственная операция. Добавление, изменение и удаление подобных свойств работают по аналогии с теми же действиями для каждой из четырех сущностей.

Для поиска мер управлений по производственным аспектам и другим видам поиска между вкладками используется система фильтров, изображенная на рис. 9B. Также для каждой из вкладок имеется возможность выборки по значениям некоторых свойств, что изображено на рис. 9E. Подобный поиск осуществляется на стороне клиента.

При применении выборки от вкладки к вкладке или по свойствам одной из вкладок данные при последующем добавлении заполняются автоматически выбранными из фильтров значениями. Также выбранные с помощью фильтров сущности используются как входные параметры для генерации отчетных документов веб-сервером и последующего скачивания их веб-приложением.

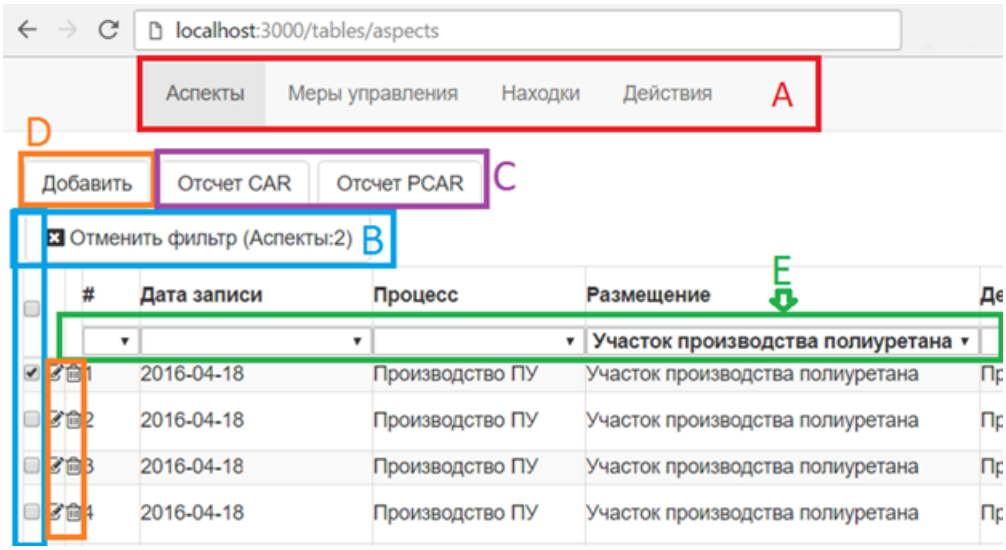

Рис. 9: A – главные вкладки, соответствующие производственным аспектам, мерам управления, находками невыполнения мер управления и антирисковыми действиями соответственно; B – фильтры, применяемые для выборки от вкладки к вкладке; C – кнопки для генерации отчетных документов; D – ви-

[30] Data binding – набор функций Angular, которые позволяют связывать свойства HTML объектов с соответствующими переменными в Angular напрямую, URL: https://angular.io/docs/ts/latest/guide/template-syntax (дата обращения 17.05.2017)

джеты для открытия форм добавления, изменения и удаления соответствующих сущностей; E – фильтры по значению полей сущностей

Архитектура

Angular позволяет создавать одностраничные веб-приложения в терминах компонент. Компонента – это логически обособленная часть приложения, которая состоит из HTML, CSS и Typescript кода. Решение состоит из компоненты, отвечающей за навигацию между вкладками, компоненты отвечающей за фильтры и так далее. Компоненты вкладываются друг в друга и в финальную HTML страницу включается одна родительская компонента.

Помимо компонент используются и другие архитектурные элементы, такие как, например, сервисы, которые доступны сразу нескольким компонентам с помощью Dependency Injection. Для примера в решении используется глобальный сервис для работы с данными при посредничестве веб-сервера или глобальный сервис, который хранит состояния всех выбранных фильтров.

Для работы с данными используются архитектурные приемы: Репозиторий и UnitOfWork [10], которые абстрагируют запросы к веб-серверу на получение или изменение данных.

3.4. Внедрение системы

Внедрение описываемой в рамках работы системы для управления рисками в информационную структуру первого предприятия-клиента осуществлялось вместе с оказанием консалтинговых услуг в этой же области другим специалистом компании «Системы КМ».

Разработка велась поэтапно с валидацией каждого из этапов специалистом-консультантом, а также конечным пользователем – сотрудником компании-клиента.

По завершении разработки была составлена спецификация программы, создан раздел справки и проведен семинар по использованию системы. В настоящее время созданная система проходит пилотное испытание.

4. Результаты

В ходе работы была создана система для управления рисками предприятия, которая получила практическое применение. Более детально, были получены следующие результаты.

- Проанализированы существующие решения и сделан вывод о недостатках этих решений.
- Разработана архитектура системы для управления рисками предприятия.
- Реализована система для управления рисками предприятия.
- Проведена миграция данных из предыдущего решения.

Список литературы

[1] ISO 9001:2015 «Quality management systems. Requirements». – 5th edition. – International Organization for Standardization, 2015 – Clause A.4.

[2] ISO 14001:2015 «Environmental management systems. Requirements with guidance for use». – 3rd edition. – International Organization for Standardization, 2015 – Clause 6.1.

[3] SAP Risk Management, product website. – https://www.sap.com/cis/solution/lob/finance/enterprise-risk-compliance.html (дата обращения 17.05.2017)

[4] Jeffrey Richter. CLR Via C#. – 4th Edition – Microsoft Press, 2015 – P. 896

[5] Adam Freeman. Pro ASP NET Core MVC – Apress, 2017 – P.804

[6] S. Donaldson, S. Siegel, C.K. Williams, A. Aslam. Enterprise Security – Apress, 2017 – P. 484

[7] Allen Grant, Owens Mike. The Definite guide to SQLite – Apress, 2010 – P. 322

[8] Mazumdar Pranab, Agarwal Sourabh, Benerjee Amit. Pro SQL Server on Microsoft Azure – Apress, 2016 – P. 205

[9] Z. Hirani, L. Tenny, N. Gupta, B. Driscoll, R. Vettor. Entity Framework 6 Recipes – Apress, 2013 – P. 501

[10] Матанит, веб-ресурс для разработчиков. – https://metanit.com (дата обращения 17.05.2017)

[11] Joseph Rattz, Adam Freeman. Pro LINQ – Apress, 2010 – P.777

[12] Adam Freeman. Pro Angular – Apress, 2017 – P.643

[13] Steve Fenton. Pro TypeScript – Apress, 2014 – P. 212

[14] Oren Fahri. Reactive Programming with Angular and ngrx – Apress, 2017 – P. 144