Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Компьютерных систем и сетей
Кафедра Информатики

К защите допустить:

Заведующая кафедрой информатики

_____ Н. А. Волорова

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту
на тему:

СЕРВИС, ПРЕДОСТАВЛЯЮЩИЙ ВОЗМОЖНОСТЬ ПОЛУЧЕНИЯ НОВИНОК МУЗЫКАЛЬНОЙ ИНДУСТРИИ

БГУИР ДП 1-31 03 04 07 045 ПЗ

Студент	Т. С. Лавник
Руководитель	В. В. Шиманский
Консультанты:	
от кафедры информатики	В. В. Шиманский
по экономической части	К. Р. Литвинович
Нормоконтролёр	Н. Н. Бабенко

Рецензент

Минск 2017

РЕФЕРАТ

Ключевые слова: музыкальная публикация; артист; тип публикации; поиск артистов; отложенные задачи.

Дипломный проект выполнен на 6 листах формата А1 с пояснительной запиской на 59 страницах, без приложений справочного или информационного характера. Пояснительная записка включает 5 глав, 6 рисунков, 16 таблиц, 48 формул и 12 литературный источник.

Целью дипломного проекта является создание инструмента, позволяющего решать задачи поиска новых музыкальных публикаций удобным способом.

Для достижения цели дипломного проекта было разработано веб-приложение на фреймворке Ruby on Rails. Веб-приложение может быть использовано для получения новых публикаций интересующих пользователя артистов. В приложении используются алгоритмы, позволяющие периодически проверять различные сервисы на наличие обновлений, касающихся интересующих релизов артистов. Также в приложении присутствует email-рассылка, отправляемая пользователям в случае наличия обновлений.

В разделе технико-экономического обоснования был произведён расчёт затрат на создание ПО, а также экономии средств при приобретении программного продукта, получаемой пользователем. Проведённые расчёты показали экономическую целесообразность проекта.

АННОТАЦИЯ

на дипломный проект «Сервис, предоставляющий возможность получения новинок музыкальной индустрии» студента УО «Белорусский государственный университет информатики и радиоэлектроники» Лавник Т. С.

Ключевые слова: музыкальная публикация; артист; тип публикации; поиск артистов; отложенные задачи.

Дипломный проект выполнен на 6 листах формата А1 с пояснительной запиской на 59 страницах, без приложений справочного или информационного характера. Пояснительная записка включает 5 глав, 6 рисунков, 16 таблиц, 48 формул и 12 литературный источник.

Целью дипломного проекта является создание инструмента, позволяющего решать задачи поиска новых музыкальных публикаций удобным способом.

Для достижения цели дипломного проекта было разработано веб-приложение на фреймворке Ruby on Rails. Веб-приложение может быть использовано для получения новых публикаций интересующих пользователя артистов. В приложении используются алгоритмы, позволяющие периодически проверять различные сервисы на наличие обновлений, касающихся интересующих релизов артистов. Также в приложении присутствует email-рассылка, отправляемая пользователям в случае наличия обновлений.

Во введении производится ознакомление с проблемой, решаемой в дипломном проекте.

В первой главе производится обзор предметной области проблемы решаемой в данном дипломном проекте. Приводятся необходимые теоретические сведения, а также производится обзор существующих разработок.

Во второй главе производится краткий обзор технологий, использованных для реализации ПО в рамках дипломного проекта.

В третьей главе производится обзор реализованного ПО. Описываются его составные части и особенности. Приводятся результаты практических испытаний и производится сравнение с существующим ПО.

В четвертой главе производится технико-экономическое обоснование разработки.

В заключении подводятся итоги и делаются выводы по дипломному проекту, а также описывается дальнейший план развития проекта.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Музыка сопровождала человечество на протяжении всей истории. Вначале люди использовали только свой голос, потом появились музыкальные инструменты. До сих пор прогресс не стоит на месте. С появлением компьютерной техники и интернета появилось ещё больше возможностей для самовыражения: программное обеспечение для звукозаписи, веб-сайты для публикации своих произведений, сервисы для анализа предпочтений и тому подобное. Учитывая существование изобилия музыкальных ресурсов, которые предоставлют пользователям разнообразные возможности, становится трудно следить за новыми публикациями любимых исполнителей, и, хотя подобные сервисы уже присутствуют на мировом рынке, не все они являются удобными и логически законченными.

В связи с вышесказанным, целью данного диплома является создание сервиса, который будет предоставлять возможность удобного получения уведомлений о новинках музыкальной индустрии. Сервис должен быть оснащён понятным и дружелюбным интерфейсом, позволять пользователю удобно регистрироваться и наполнять данными свой профиль, указывая свои предпочтения в виде списка интересующих артистов.

Для того, чтобы сервис был полезен пользователю, он должен быть удобен, привлекателен и реализовывать некоторую потребность. В этом состоит основа данного дипломного проекта: собирать информацию из разных сервисов в одном месте, тем самым упрощая этот для конечного пользователя. В некотором роде - это агрегат, который собирает интересы и возвращает ответ в виде удовлетворения этих интересов путём указания ссылок на публикации любимых артистов. Удобство должно быть выражено в дизайне интерфейса, который должен быть интуитивно ясен пользователю и помогать ему производить некоторые операции внутри сервиса. Также проект должен быть хорошо масштабируемым, чтобы можно было легко добавлять разные источники публикаций в свой арсенал.

1 ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Постановка задачи

Целью данного дипломного проекта является создание веб-сервиса, состоящего из серверной и клиентской частей с использованиям Ruby on Rails, AngularJS, Sidekiq, для получения уведомлений о новинках в музыкальной индустрии.

Для реализации данного проекта можно выделить следующие минимальные требования:

– Система должна быть компонентной, то есть разделённой на отдельные модули. Это облегчает разработку, внесение изменений и тестирование.

– Система должна поддерживать мультиязычность для охвата большей аудитории с возможностью добавления новых языков и редактирования старых.

– Система должна соответствовать современным требованиям к интерфейсам быть удобной для пользователя.

– Система должна быть выполнена в виде SPA-приложения с использованием асинхронных запросов.

– Система должна быть легко масштабируема для добавления новых источников публикаций.

– Система должна быть имплементирована с наличием авторизации, аутентификации, разных ролей, в том числе администраторской.

Для реализации вышеописанных требований необходимо решить следующие задачи:

– Изучить аналоги, присутствующие на рынке, для того, чтобы учесть ошибки и недостатки изучаемых сервисов, а также выделить сильные стороны и использовать их в разработке.

– Изучить предметную область, портрет целевой аудитории, чтобы лучше понимать нужды и правильно реализовать их в системе.

– Выбрать и изучить подходящие технологии для разработки сервиса.

– Создать архитектуру, которая будет удовлетворять требованиям разработки и реализовать её с использованием выбранных технологий.

1.2 Обзор существующих аналогов

В данном разделе будут рассмотрены приложения, которые по логике своей работы пересекаются с тематикой данной дипломной работы, а именно предоставляют пользователям возможность узнавать о новинках музыкальной индустрии. Некоторые из них является веб-сервисами, направленными на видеоконтент, другие - на музыкальный. И те, и другие выполняют свою задачу и направлены на свою целевую аудиторию. Рассмотрим несколько из вышеописанных сервис.

1.2.1 YouTube - Just Released Music Videos

YouTube - видеохостинговый сервис, предоставляющий пользователям загружать свои видео, делать их некоторую обработку и получать статистику просмотров. Также на YouTube есть комментарии, кнопки “мне нравится” и “мне не нравится”. Таким образом данный сервис имеет большую вовлечённость пользователей.

Однако в данной дипломной работе рассматривается отдельный сервис, который пересекается с её тематикой - это Just Released Music Videos. Эта площадка показывает людям последние публикации артистов. Другими словами, Just Released Music Videos демонстрирует пользователям список из только что издававшихся видео-роликов певцов, музыкантов, музыкальных коллективов. Данный сервис интегрирован в YouTube и является его частью.

Список видеороликов, представленный в Just Released Music Videos отсортирован в порядке убывания по дате публикации, то есть, чем новее видео, тем выше в этом списке оно поднимается, что логично для пользователя, который ожидает увидеть именно новинки музыкальной индустрии. Также напротив видео будет указана длительность композиции и небольшая картинка, которая является превью к видео.

У данного сервиса можно выделить следующие преимущества:

– позволяет пользователю не только посмотреть музыку, но и увидеть видеоклип;

– позволяет пользователю видеть количество просмотров, таким образом оценивая его популярность;

– есть возможность включения субтитров, что является встроенной функциональностью YouTube, таким образом позволяя пользователю сразу видеть текст композиции при условии его присутствия;

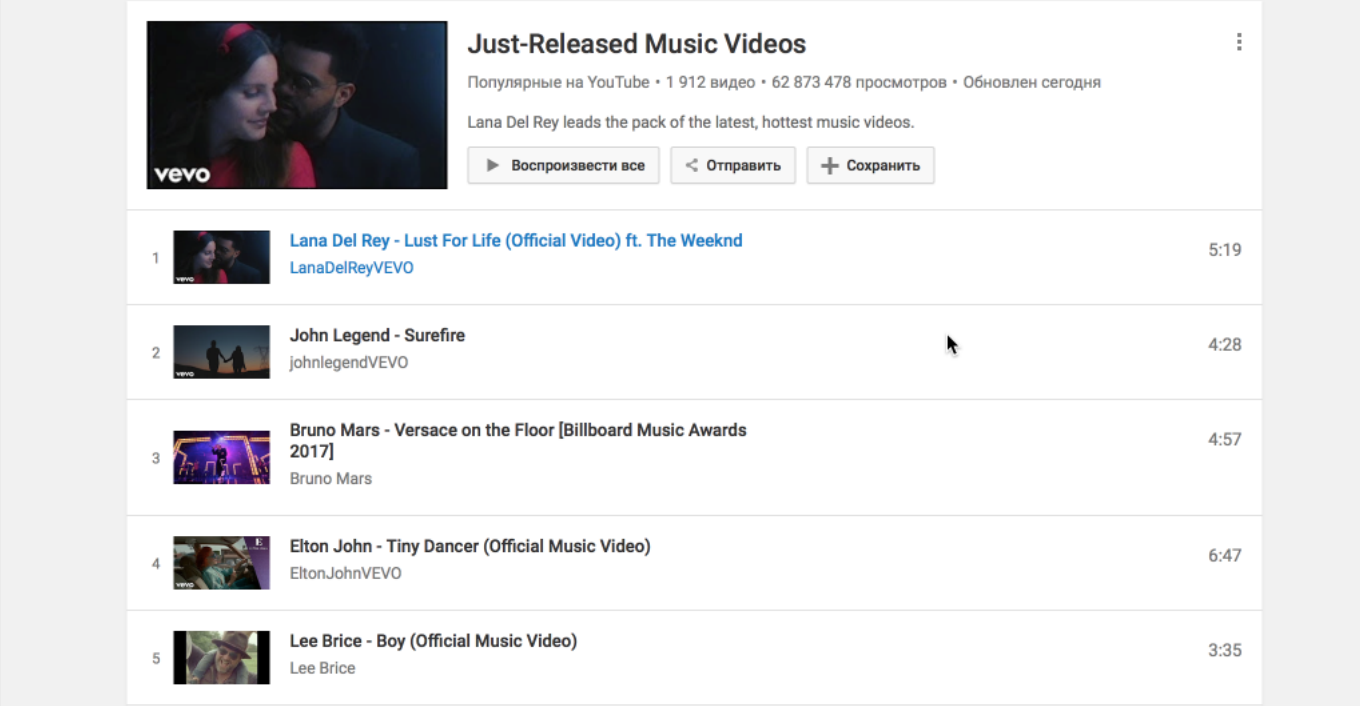


Рисунок 1.1 – Изображение сервиса YouTube - Just Released Music Videos

Однако кроме преимуществ можно выделить следующие недостатки:

– артисты, представленные в списке данного сервиса, никак не связаны с тем, что интересно пользователю, тем самым создавая лишнюю информационные данные;

– сервис не является самостоятельным, а интегрированным в YouTube;

– нет возможности получения прямых уведомлений на почту или в нотификациях переносимого гаджета;

1.2.2 AllMusicAllMusic представляет собой американский сервис, который включает в себя крупную онлайновую музыкальную базу данных. В AllMusic представлена информация о жанрах музыки, музыкантах и коллективах, в том числе новые публикации и профессиональные рецензии. Данный сервис является обладателем огромного музыкального архива, который насчитывает около шести миллионов композиций. Также на сайте имеется возможность оценки публикаций пользователями. AllMusic предоставляет пользователю довольно широкий спектр возможностей. Например, пользователь может подписаться на обновления, чтобы получать письма с уведомлением о новой публикации. Также есть возможность получать индивидуальные музыкальные рекомендации. Для этого необходимо заполнить артистов, которые нравятся пользователю, в поля для ввода и далее будут предложены альбомы артистов по каким-либо параметрам схо-

жих с введёнными ранее. Я считаю, что это весьма полезная функциональность для того, чтобы найти новую музыку. AllMusic обладает стильным, выдержанным, но, на мой взгляд, слегка загроможденным дизайном. Сервис предоставляет широкую функциональность, и по этой причине пользователю, заинтересованному лишь в получении музыкальных новинок, может мешать наличие множества разнообразных разделов. Сервис перестаёт быть сконцентрированным на одной вещи. Это, в зависимости от интересов конечного пользователя, может как оттолкнуть, так и быть интересным.

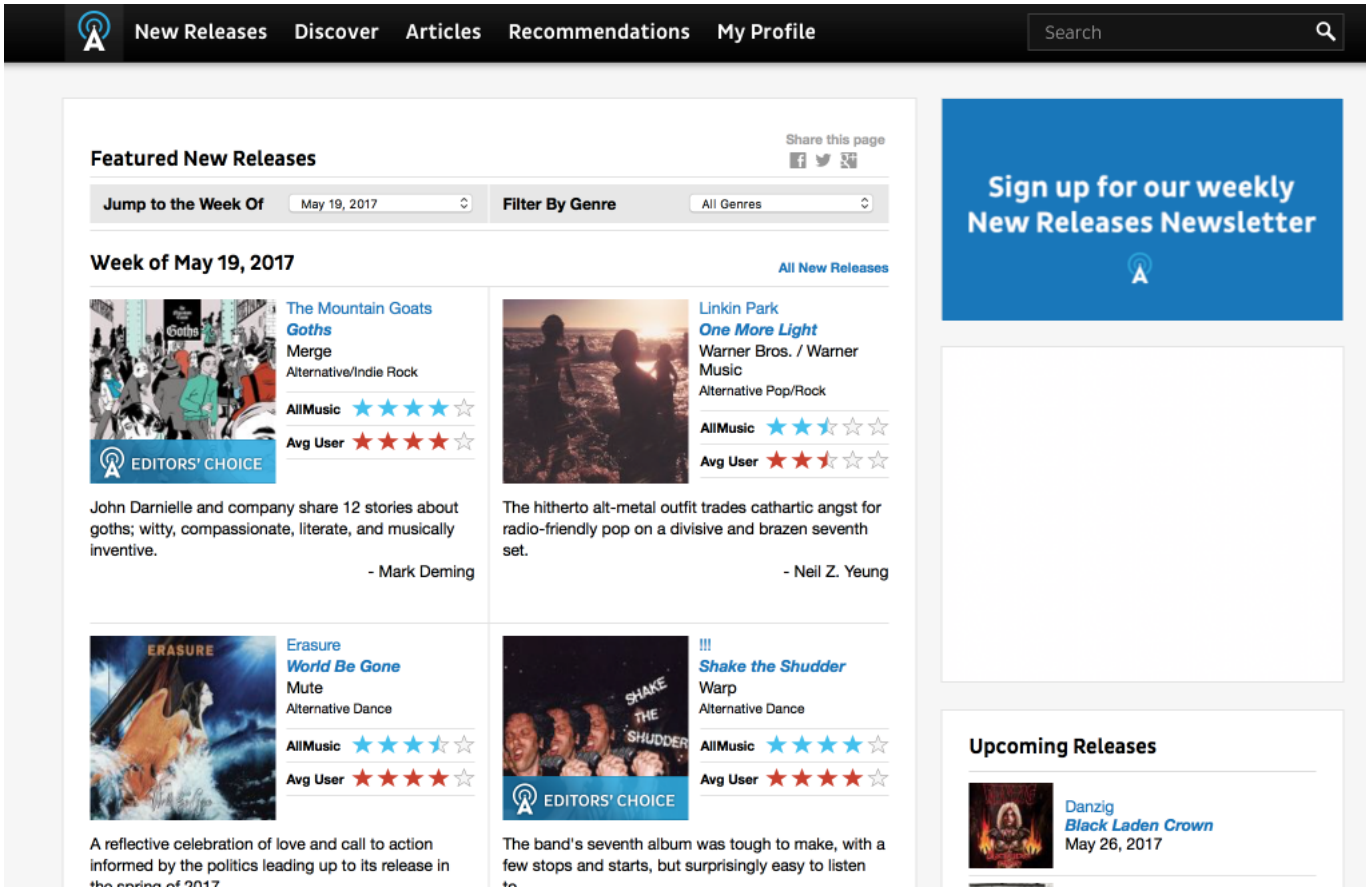


Рисунок 1.2 – Изображение сервиса AllMusic

Преимущества AllMusic:

– возможность регистрации через социальные сети;

– большая база данных публикаций;

– возможность оценивать альбомы, что позволяет пользователям судить о качестве;

– богатство жанров;

– возможность получать уведомления;

– наличие персональных рекомендаций;

– возможность прослушать отрывки композиций;

Также можно обозначить следующие недостатки:

– отсутствие концентрированности на публикациях артистов, и, как следствие, массивный интерфейс;

– отсутствие ссылок на сервисы для приобретения альбомов;

– отсутствие возможности импортировать артистов с устройств/сервисов;

– небольшая интерактивность;

1.2.3 ВыводУчитывая результаты ознакомления с уже присутствующими аналогами сервисов для получения информации о новинках музыкальной индустрии, можно сделать вывод, что большинство подобных сервисов либо слишком усложнены, либо делают акцент не на основную потребность конечного пользователя, а именно получение уведомлений, информации о новых публикациях любимых артистов.

1.3 Музыкальный альбом (публикация)

Музыкальный альбом - это некоторая совокупность музыкальных композиций, которые выпускаются вместе, в стандартном формате. Такие композиции должны быть доступны для произведения на популярных воспроизводящих устройствах. Существуют классификации альбомов по нескольким признакам: по объёмы, типу записи и так далее. Некоторые из них будут рассмотрены далее.

1.3.1 Классификация по объёму

– Сингл не относят к альбомам, так как состоят из одной-двух песен. Рассматривается в данной работе для большей ясности нижеприведённой информации.

– Стандартный альбом, или LP (от английского “Long Play”) обычно размещается на одной пластинке или же на одном компакт-диске. Время звучания насчитывает около 30-80 минут.

– Мини-альбом, или EP (от английского “Extended Play”) по своему размеру является промежуточным вариантом между стандартным и синглом.

– Двойной альбом выпускается на двух носителях информации, и время звучания такого альбома в среднем в два раза больше, чем стандартный альбом.

1.3.2 Классификация по типам записей и композиций

– Студийный альбом - альбом, записанный в звукозаписывающей студии.

– Концертный альбом - альбом, который записывается в то время, как идёт “живое” выступление перед публикой, то есть запись концерта.

– “Программный” альбом - альбом, в котором обычно содержаться только ранее не издававшиеся композиции, хотя бывают и исключения, когда в “программный” альбом включаются концертные записи.

1.3.3 Другие типы альбомов

– Демо представляет собой демонстрационную запись. В ней обычно содержится “сырой” материал, который используется для ознакомления других лиц с творчеством исполнителя. Зачастую это некоторые первые записи музыкантов, которые официально не издавались.

– Концептуальный альбом - это альбом, все композиции которого объединены некоторой идеей (концепцией). В таких альбомах можно усмотреть некоторый замысел в составе и порядке композиций.

– Промо - альбом, который выпускается небольшим тиражом специально для бесплатного распространения с целью рекламы исполнителя.

– Саундтрек - альбом, в составе которого используются композиции из одного или нескольких фильмов, компьютерных игр.

– Трибьют - альбом, в котором в дань уважения и в знак почитания творчества одни исполнители переигрывают песни другого определённого исполнителя. Такие композиции называются каверами.

1.4 API

API(application programming interface, программный интерфейс приложения) - это совокупность процедур, функций, и классов и т.д. , которую предоставляет сервис для использования в некоторых внешних приложениях (клиентах).

API используется для того, чтобы представить функциональность программы, при этом абстрагируясь от конкретной реализации.

WebAPI - это API, основу в котором составляют HTTP-запросы и HTTP-ответы, сконструированные по определённым принципам. API используется, как источник данных для клиентского приложения. Клиент использует HTTP-запросы с глаголами GET, POST, PATCH, DELETE и другие. При этом в теле запроса передаются необходимые параметры. Далее API принимает этот запрос, и, если необходимо, авторизирует клиент. Может использоваться ролевая авторизация, когда в системе присутству-

ют разные типы пользователей: администраторы, модераторы, конечные пользователи. После запрос обрабатывается, и возвращается ответ в виде XML или JSON. Ответ принимается клиентом, также обрабатывается и на основе результатов производятся необходимые действия.

2 ОБЗОР ИСПОЛЬЗУЕМЫХ ТЕХНОЛОГИЙ

2.1 Ruby on Rails

Ruby on Rails - веб-фреймоворк, написанный на языке Ruby. Ruby on Rails известен тем, что позволяет очень быстро начать разрабатывать приложения. В данном веб-фреймворке есть возможность под названием “scaffold”, которая позволяет одной командой создавать представления, контроллер и модель для нужного объекта.

Философия Ruby on Rails складывается из двух основных пунктов:

– “Convention over configuration” - что переводится, как преобладание соглашений над конфигурацией. Это означает, что в экосистеме Ruby on Rails разработчикам предлагается пользоваться общепринятыми соглашениями для того, чтобы ускорить разработку. Например, названия таблиц в БД должны называться множественным числом модели, для которой создаётся таблица. Также в настройках самого фреймворка присутствует множество дефолтных значений. Это не означает, что разработчик не может сконфигурировать приложение по своему желанию. Такая возможность, безусловно, присутствует, однако предлагается пользоваться соглашениями. Это не только ускоряет разработку, но и увеличивает уровень взаимопонимая в команде.

– “DRY” - что расшифровывается, как “don’t repeat yourself”. Это означает избегать дублирования кода. Желательно писать приложение таким образом, чтобы оно легко масштабировалось, методы и компоненты могли быть переиспользованы в различных местах.

В Ruby on Rails используется ORM ActiveRecord. ORM представляет собой прослойку между логической моделью и её представлением в БД. ActiveRecord предоставляет богатый функционал для работы с моделями. ActiveRecord - это также шаблон проектирования. Принцип работы ActiveRecord состоит в том, что в приложении создаётся класс, который отображается на таблицу в БД, так что:

– любой объект класса, унаследованного от ActiveRecord::Base отображается на строку в таблице БД;

– чтобы создать новую запись в таблице, нужно создать новый валидный объект;

– в качестве свойств объекта выступают поля в соответствующей строке БД;

– строка в таблице БД изменяется или удаляется, если изменяется

или удаляется объект ActiveRecord::Base;

Также ActiveRecord предоставляет инструмент манипулирования таблицами посредством миграций - методов, в которых на языке Ruby описываются действия с БД. Для создания столбцов таблиц ActiveRecord предлагает следующие типы:

- binary
- boolean
- date
- datetime
- decimal
- float
- integer
- bigint
- primary_key
- references
- string
- text
- time
- timestamp

Веб-фреймворки в большинстве своём обладают HTML-процессорами, которые позволяют дополнять HTML-разметку различными вставками, и Ruby on Rails не является исключением. Стандартный HTML-процессор в Ruby on Rails называется ERB. Он позволяет помещать в HTML-разметку Ruby-код, который создаётся в контроллере, тем самым давая возможность разработчикам избежать статических HTML-страниц. Однако помимо ERB существуют и другие HTML-процессоры, которые предоставляют возможность не только дополнять HTML-разметку, но и вводить новый вид разметки веб-страниц со вставками Ruby-кода.

В Ruby on Rails присутствует инструмент для соответствия URL-адреса определённому методу в контроллере. Такой инструмент называется router или маршрутизатор. В проекте, созданном на базе Ruby on Rails, существует файл routes.rb, в который помещаются так называемые маршруты. В маршрутизаторе есть возможность не только создавать маршруты не только по одному, но и целыми группами с помощью методы resources, который предлагает создание всех необходимый операций для ресурса: CREATE, READ, UPDATE, DELETE - посредством HTTP запросов.

Выбор фреймворка Ruby on Rails для разработки обусловлен удоб-

ством его использования, расширяемостью его с помощью различных библиотек, созданных Ruby on Rails-сообществом, которое на протяжении не менее десяти лет активно развивается и предлагает богатый выбор инструментов для разработки.

2.2 AngularJS

AngularJS - javascript-фреймворк, созданный в компании Google, совершивший прорыв в веб-разработке. AngularJS позволяет создавать динамические SPA. AngularJS, как и Ruby on Rails, обладает MVC-архитектурой. На момент создания AngularJS эта архитектура была революционной по причине того, что javascript-код преимущественно представлял отдельные неорганизованные скрипты, которые очень сложно масштабировать. AngularJS в свою очередь предлагает разделить код на контроллеры, представления и модели(ресурсы), которые в свою очередь могут находиться в отдельных модулях.

3 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ

Разработанное программное обеспечение представляет из себя веб-приложение, написанное на фреймворке Ruby on Rails. Приложение предназначено для получения новинок музыкальной индустрии или, другими словами, музыкальных релизов выбранных пользователем музыкантов.

Существует множество подходов для построения архитектуры веб-приложений. Я выбрал в своей реализации трёхуровневую архитектуру.

3.1 Трёхуровневая архитекутра приложения

Архитектура приложения - это определённый подход в определении взаимоотношений компонентов приложения. Обычно архитектура приложений соответствует задачам разработки. В данной дипломной работе выбрана трёхуровневая архитектура. Эти уровни называются: уровень модели, уровень контроллера и уровень представления. Иначе говоря, MVC - Model, View, Controller.

Данная архитектура используется во множестве веб-фреймворков. Она удобна тем, что позволяет разделить зоны ответственности между компонентами, облегчить тестирование, придать гибкость разработке. Фреймворк Ruby on Rails и AngularJS также в своей основе используют данную архитектуру.

3.1.1 Уровень модели

Уровень модели - один из трёх уровней в MVC-архитектуре. Его обязанности можно разделить на две категории:

– бизнес-логика приложения;

– управление связью в базой данных;

Бизнес-логика включает в себя определённые правила, отношения и принципы, которые должны существовать в приложении. Эти правила могут быть выражены как в коде самого приложения, так и базе данных. Однако в данной дипломной работе бизнес-логика содержится именно в коде приложения, а именно в ruby-классах, наследующихся от класса ActiveRecord::Base, который является стандартом де-факто для моделей, использующихся во фреймворке Ruby on Rails.

В данной дипломной работе присутствует класс ReleaseIdentity, который используется для хранения и обработки объектов музыкальных публи-

каций. Пример реализации класса ReleaseIdentity представлен на листинге 3.1:

Листинг 3.1 – Базовая реализация класса ReleaseIdentity

```
class ReleaseIdentity < ActiveRecord::Base
end
```

Классы-наследники класса ActiveRecord::Base обладают богатым функционалом. Они содержат в себе возможности валидирования, организации связей между объектами, возможностями создания транзакций, сложных запросов в БД и многими другими. Подробнее об этом будет написано несколько позже, когда будет рассматриваться взаимодействие с БД.

Ярким примером бизнес-логики может служить проверка корректности конструируемых объектов. Например, название артиста не должно быть пустым и содержать только текстовые данные. Или же год начала карьеры артиста не может быть отрицательными или больше нынешнего года по значению. Всё это реализуется в Ruby on Rails с помощью механизма валидации.

В данной дипломной работе присутствует класс RymArtistsInfo, который используется для хранения результатов работы с сервисом Rate Your Music[1)]. В этом классе присутствуют следующие поля:

– id - уникальный идентификатор записи в таблице;

– formed year - год формирования музыкальной группы(исполнителя);

– disbanded yead - год окончания деятельности музыкальной группы(исполнителя);

– genres - жанры, которые присущи музыкальной группе(исполнителю);

– created at - дата создания записи в БД приложения;

– updated at - дата обновления записи в БД приложения;

В данном случае годы формирования и окончания деятельности музыкальной группы исполнителя должны придерживаться правил, описанных выше. Также строка, содержащая жанры, не может иметь длину, менее 2 символов. Для обеспечения этой части бизнес-логики я воспользовался инструментарием фреймворка Ruby on Rails. Используя метод validates, можно добиться проверки определённых правил перед сохранением записи, чтобы хранить в базе только корректные записи. Также существует метод validate, в который передаётся название метода, в котором можно описать более сложную логику. Таким образом можно проверить то, что год окончания деятельности музыкальной группы больше или равен го-

[1)] https://rateyourmusic.com/

ду начала карьеры. Пример использования продемонтрирован на листинге 3.2:

Листинг 3.2 – Реализация валидации в классе RymArtistsInfo

```
class RymArtistsInfo < ActiveRecord::Base
  validates :formed_year
  validates :disbanded_year, numericality: {only_integer: true, greated_than: 0,
      less_than: DateTime.now.year}
  validates :genres, length: {minimum: 2}
  validate disbanded_more_or_eql_formed

  private

  def disbanded_more_or_equal_formed
    disbanded_year >= formed_year
  end
end
```

Фреймворк Ruby on Rails предлагает ещё один полезный инструмент, который называется "функция обратного вызова". Такие функции существуют и на уровне базы данных, но в этом и состоит преимущество веб-фреймворка - позволять более компактно и удобно писать необходимый функционал. Функции обратного вызова работают таким образом, что в определённые моменты жизненного цикла объектов, такие, как перед сохранением, удалением или валидированием объекта, вызываются методы, в который можный поместить необходимый для этого момента код. Это ещё одно преимущество слоя модели, потому как позволяет избегать длинных цепочек вызовов методов, а воспользоваться удобным интерфейсом для обработки моментов жизненного цикла. В Ruby on Rails такие моменты обрабатываются путём добавления методов:

- before_save;
- before_create;
- before_validate;
- around_save;
- after_save;
- after_update;
- и др.;
- КОД КОЛБЭКА –

Так как в дипломном проекте рассматривается одна из разновидностей графовых моделей, то, очевидно, для представления таких моделей в разрабатываемой библиотеке должна быть часть, отвечающая за представление и работу с графами. При реализации здесь было несколько альтернативных путей: использовать одну из доступных библиотек для платформы

Microsoft .NET для работы с графами или реализовать собственную. Среди готовых библиотек можно было бы использовать QuickGraph[1)], Directed Graph for .NET[2)] или GrapheNET[3)]. Но было принято решение остановиться на варианте, подразумевающем разработку собственных типов для работы с графами. Это решение было обосновано тем, что вышеуказанные библиотеки являются сложными и содержат в себе очень много функциональности не нужной для решения поставленных задач, в дополнение не было приобретено лишних мегабайтовых внешних зависимостей для библиотеки.

Одним из важных решений, которое было принято в начале проектирования модуля работы с графами, было использование по возможности неизменяемых структур данных. Это решение выгодно отличает разработанную реализацию от существующих библиотек, от использования которых было принято решение отказаться. Существующие библиотеки ориентированы на работу в императивном стиле и с изменяемым состоянием. Также использование неизменяемых структур данных для реализации типов для представления графов в дальнейшем положительно сказалось на простоте реализации поиска структуры вероятностной сети в алгоритмах вывода структуры по данным. Часть разрабатываемой библиотеки, содержащая типы для работы с графами, реализована на языке программирования F#. Краткое описание основных особенностей данного языка приведено в подразделе ??. Решение использовать данный язык было продиктовано желанием сократить количество возможных ошибок и размер кодовой базы, необходимой для реализации поставленной задачи, а так же желанием применить в «боевых» условиях язык программирования с хорошей поддержкой функционального программирования.

Внутренним представлением графа является неизменяемый словарь, содержащий вложенный неизменяемый мульти-словарь[4)]. Основные определения структуры данных графа приведены в листинге 3.3:

Листинг 3.3 – Определение структуры данных для представления графа

```
/// Graph arc
type Arc<'T> =
    | Outgoing of 'T
    | Incoming of 'T

/// Immutable Graph class
```

[1)] http://quickgraph.codeplex.com/
[2)] http://directedgraph4net.codeplex.com/
[3)] http://graphenet.codeplex.com/
[4)] Словарь позволяющий хранить множество значений с одинаковым ключом.

```
[<ReferenceEquality; NoComparison>]
type Graph<'Vertex, 'Arc when 'Vertex : comparison and 'Vertex : equality and 'Vertex :>
    IComparable> =
    private Graph of Map<'Vertex, MultiMap<'Vertex, Arc<'Arc>>>
```

Данная структура данных для представления графа подходит для работы как с неориентированными так и с ориентированными мультиграфами. Воспринимать граф как ориентированный или нет задача конкретного алгоритма, работающего с графом. Разработанная библиотека для представления графа предоставляет необходимые операции для манипулирования структурой графа. Библиотека также предоставляет небольшое количество алгоритмов для работы с графами, необходимых в рамках решения задач, возникающих при поиске структуры вероятностной сети. В частности реализованы алгоритмы топологической сортировки, поиска в глубину и ширину, поиска сильно-связанных компонетов и проверки графа на наличие направленных циклов.

Одной из особенностей разработанной библиотеки является ориентированность на использование как из языка F# в «функциональном стиле», так и из языка C# — в «императивном». Для реализации данной возможности были учтены рекомендации приведенные в [1]. Функциональность разработанной библиотеки покрыта большим набором модульных тестов, написанных с использованием библиотек xUnit[1)] и Unquote[2)].

3.2 Представление вероятностной сети

Другой важной частью разработанной библиотеки являются типы для представления и работы с самими вероятностными сетями. Первостепенными требованиями, поставленными перед началом проектирования типов, были следующие пункты:

– Типы предназначены для представления модификации классических байесовых сетей, упомянутой в разделе ?? на странице ??.

– Представление сети должно быть «многослойным». Под «многослойностью» понимается возможность расширения представления сети дополнительными «слоями» атрибутов, с целью увеличения количества сценариев, в которых данные типы пригодны к использованию. Например, в случае когда нужно знать лишь структуру сети можно использовать лишь информацию о структуре — граф. Для проведения статистического вывода суждений добавляется дополнительный «слой», содержащий талицы услов-

[1)] http://xunit.codeplex.com/
[2)] http://code.google.com/p/unquote/

ных и безусловных вероятностей. В случаях, когда нужно отображать сеть пользователю, добавляется еще один «слой», содержащий дополнительную информацию о переменных и их состояниях.

– Сеть должна предоставлять возможность отмены вносимых в нее изменений, т. е. по сути поддерживать версионность.

– Сеть должна предоставлять возможность валидации её структуры.

Приняв во внимание приведенные выше требования были приняты следующие решения:

– Необходимо разработать отдельные типы для представления вершин вероятностной модели и связей между переменными в этой модели. В разработанной библиотеке за это отвечают типы `Node<'T>` и `Link<'T>`, содержащие информацию о переменных, таблицы распределения и дополнительные атрибуты. Использование параметрического полиморфизма в реализации данных типов играет ключевую роль в обеспечении «многослойности» и расширяемости представления вероятностной сети.

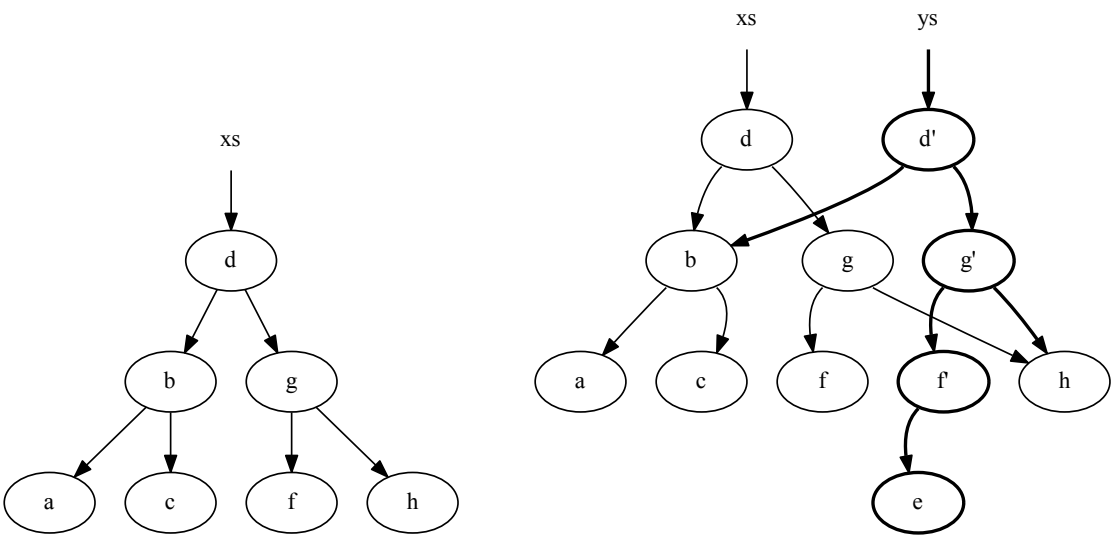
– Необходимы типы для представления распределения. В предложенной реализации был разработан тип для представления безусловного распределения случайной величины, эта таблица хранится в сети как один из аттрибутов типа `Node<'T>`, и тип для представления условного распределения пары случайных величин, экземпляр данного типа хранится как аттрибут связи между переменными — `Link<'T>`. Было сочтено целесообразным в качестве внутренней реализации таблиц распределения использовать готовую библиотеку для работы с матрицами и другими математическими объектами и понятиями — Math.NET Numerics[1)]. Соответственно в предложенной реализации использовались типы `Vector<float>` и `Matrix<float>` и сопутствующие операции над ними. Использование данной библиотеки позволило сократить объём сопутствующего кода, необходимого для реализации библиотеки для работы с вероятностными сетями, также уменьшив множество потенциальных ошибок реализации. В данном случае преимущества от использования библиотеки превысили затраты на добавление и поддержку дополнительных зависимостей.

– Требование возможности отмены изменений вносимых в вероятностную сеть привело к реализации сети, как и в случае типов для представления графов, к реализации сети как неизменяемой структуры данных. Все операции, при условии использования специальных функций, возвращают новый экземпляр сети, оставляя старый не изменённым. Подобная реализация типов автоматически дает возможность производить версиони-

[1)] `http://numerics.mathdotnet.com/`

рование экземпляров типа, т. к. всегда есть доступ к изменённой копии и исходному экземпляру. С первого взгляда данный подход кажется очень расточительным по памяти, но на самом деле оказывается, что все с точностью до наоборот, т. к. обычно, и в данном конкретном случае, при модификации неизменяемой структуры данных большая часть структуры разделяется между копией и исходной структурой, а физически копирование памяти происходит лишь в тех местах, которые действительно необходимо было поменять. Для убедительности, сказанное проиллюстрировано на рисунке 3.1.

– Валидация сети происходит на этапе её построения и модификации. Дополнительно существуют функции для проверки структуры сети на ацикличность. Ацикличность ориентированного графа проверяется с помощью алгоритма нахождения компонент сильной связности Косарайю[1)].



(а) (б)

Рисунок 3.1 – Пример разделения структуры в неизменяемых структурах данных: а — исходное дерево; б — измененное дерево, добавлена вершина e;

В результате получилось довольно простое и расширяемое представление сети, основное определение которого приведено в листинге 3.4. Параметризация типа параметрами `'NodeAttributes` и `'LinkAttributes` и расширяемое устройство типов `Node<'T>` и `Link<'T>` позволяет достигнуть заявленной расширяемости представления сети и её «многослойности». Библиотека содержит

[1)] `http://en.wikipedia.org/wiki/Kosaraju's_algorithm`

предопредёленные типы, для представления уровней. Параметризация типа `BNet<unit, unit>` представляет структуру сети и таблицы распределения. «Слой» с дополнительными аттрибутами, планируемыми для использования в алгоритмах статистического вывода суждений представлен типами `NodeAttributes<'Annotations>` и `LinkAttributes`, но на момент защиты дипломного проекта, из-за неготовой реализации алгоритмов статистического вывода суждений, данные типы содержат не все необходимые атрибуты для работы таких алгоритмов, а лишь прогнозируемых заготовки. Дополнительный «слой», потенциально необходимый при построении пользовательских приложений, представлен типом `VarAnnotations`. Данный тип содержит дополнительную информацию о переменной, такую как её название и аннотации возможных значений переменной. Таким образом, наиболее полное представление сети, содержащее все «слои», в коде параметризуется следующим образом `BNet<NodeAttributes<VarAnnotations>, LinkAttributes>`.

Листинг 3.4 – Определение структуры данных для вероятностной сети

```
/// Represents immutable BN. Nodes and links between nodes.
type BNet<'NodeAttributes, 'LinkAttributes> =
    private { nodes: Map<int, Node<'NodeAttributes>>;
              links: Map<int * int, Link<'LinkAttributes>>;
              graph: Graph<int, unit>; }
```

Таким образом, использование параметрического полиморфизма и неизменяемых типов данных, позволило добиться поставленных при проектировании целей, а также довольно легкой возможности расширять сеть в дальнейшем.

3.3 Сохранение сети

Помимо функциональности, связанной с представлением, манипуляцией и выведением структуры сети, разработанная библиотека предоставляет возможность импорта и экспорта вероятностной сети из и в различные форматы. Т. к. библиотека предназначена для работы с модификацией вероятностных сетей, отличающейся в нескольких ключевых моментах от классических байесовых сетей, то нельзя было использовать общепринятые форматы для хранения сетей во внешней памяти и необходимо было разработать свой формат. Разработанный формат хранения сетей основывается на XML и предназначен для полного сохранения состояния представления сети, используемого в программе. Данный формат в большей степени похож на ручную сериализацию, чем на удобный формат для обмена вероятностными сетями. Пример сети, представленной в данном формате,

приведен в листинге 3.5.

У разработанного формата есть существенный недостаток, его «понимает» только разработанная библиотека. В связи с тем, что в данном дипломном проекте основной целью является реализация лишь малой части возможных операций над вероятностными сетями — построение структуры по данным, целесообразно было добавить в библиотеку, хотя и весьма ограниченную, возможность загрузки и сохранения вероятностных сетей из и в существующие распространённые форматы. Список поддерживаемых форматов приведён в таблице 3.1. Поддержка нескольких форматов понадобилась потому, что многие существующие программы, которые использовались в различной степени для оценки результатов проделанной работы, поддерживают весьма ограниченный набор форматов. Таким образом, имея возможность экспортировать сеть, построенную одним из алгоритмов реализованных в разработанной библиотеке по данным, в общеиспользуемый формат можно использовать обученную сеть для статистического вывода суждений и других операций в существующих программах, т. к. в данный момент в разработанной библиотеке данная функциональность не реализована. Отдельно стоит отметить возможность сохранения структуры сети в формат представления графов программы GraphViz[1)]. Данное ПО использовалось с целью визуализации выведенных структур и экспорта полученной визуализации в один из векторных графических форматов. Утилита dot из состава GraphViz умеет автоматически визуализировать сложные графы наилучшим для отображения образом.

[1)] `http://www.graphviz.org/`

Листинг 3.5 – Пример представления простой вероятностной сети в собственном XML-формате

```
<network>
  <variables>
    <variable id="1" dim="2" />
    <variable id="2" dim="3" />
  </variables>
  <node_attributes>
    <node variable_id="1"> <answered>false</answered> </node>
    <node variable_id="2"> <answered>true</answered> </node>
  </node_attributes>
  <link_attributes>
    <link parent_id="1" child_id="2" />
  </link_attributes>
  <variable_annotations>
    <variable_annotation variable_id="1">
      <name>My variable</name>
      <annotations>
        <label>Yes</label> <label>No</label>
      </annotations>
    </variable_annotation>
    <variable_annotation variable_id="2">
      <name>Color</name>
      <annotations>
        <label>Red</label> <label>Green</label> <label>Blue</label>
      </annotations>
    </variable_annotation>
  </variable_annotations>
  <probability_tables>
    <probability_table variable_id="1">
      <vector>0.2 0.8</vector>
    </probability_table>
    <probability_table variable_id="2">
      <vector>0.4 0.3 0.3</vector>
    </probability_table>
  </probability_tables>
  <forward_probability_tables>
    <forward_probability_table variable_id="2" condition_variable_id="1">
      <matrix nrows="3" ncols="2">0.1 0.2 0.7 0.6 0.1 0.3</matrix>
    </forward_probability_table>
  </forward_probability_tables>
</network>
```

3.4 Представление экспериментальных данных

Немаловажной задачей в обучении и построении структуры сети по данным является представление набора экспериментальных данных в оперативной и постоянной памяти. В машинном обучении и других областях, связанных с обработкой массивов данных, для хранения данных на диске в большинстве случаев применяется простой текстовый формат csv —

Таблица 3.1 – Поддерживаемые форматы хранения вероятностных сетей

Формат	Поддержка импорта	Поддержка экспорта
Собственный xml-формат	полная	полная
XMLBIF	частичная	частичная
GeNIe	частичная	отсутствует
GraphViz dot	отсутствует	полная

значения, разделённые специальным символом и записанные в текстовый файл построчно. В разработанной библиотеке также используется данный формат для импортирования экспериментальных данных с диска в память программы для дальнейшей обработки. Для чтения csv файлов используется легковесная внешняя библиотека LumenWorks.Framework.IO[1)].

Для представления набора экспериментальны данных в библиотеке присутствует специальный тип — DataFrame, который представляет из себя информацию о переменных и, собственно, набор экспериментальных данных в компактном для хранения виде. Из особенностей реализации стоит упомянуть способ достижения компактности хранения. При чтении csv файла каждому состоянию переменной назначается некоторое 8-битное число, которое является представлением данного состояния в памяти компьютера. Использование 8-битного числа с одной стороны ограничивает число возможных состояний одной переменной до 256, с другой стороны — данное представление достаточно компактно, чтобы быть пригодным для работы на персональном компьютере разработчика с ограниченным размером ОЗУ и уметь обрабатывать наборы данных из миллионов случаев для десятков переменных. Одной из дополнительных возможностей DataFrame является возможность производить случайные выборки из имеющегося набора данных. Данная возможность была использована при реализации алгоритмов вывода структуры вероятностной сети по данным.

3.5 Байесовы сети Asia и ALARM

Перед тем как перейти к обсуждению разработанных алгоритмов вывода структуры сети по данным целесообразно обсудить известные сети, которые использовались в качестве моделей для вывода по экспериментальным данным. Речь пойдет о ставших уже классикой в таких задачах — сетях Asia и ALARM.

[1)] `http://www.codeproject.com/Articles/9258/A-Fast-CSV-Reader`

3.5.1 Asia Байесова сеть Asia является небольшой синтетической сетью, обычно используемой при изучении вероятностных сетей. Данная вероятностная сеть рассмотрена в работе [2]. Искусственная байеосова сеть Asia предназначена для диагностики у пациентов заболеваний связанных с лёгкими. В перечень диагностируемых болезней входят туберкулёз, рак и бронхит. Данная сеть имеет восемь бинарных случайных величин. Структура данной сети приведена на рисунке ?? (б) на странице ??. В таблице 3.2 приведено описание переменных.

Таблица 3.2 – Описание переменных сети Asia

Переменная	Количество состояний	Примечание
VisitAsia	2	посещал ли пациент Азию
Tuberculosis	2	болен туберкулёзом
Smoking	2	курит
Cancer	2	имеет рак легких
TbOrCa	2	имеет рак или туберкулёз
XRay	2	плохая рентгенография
Bronchitis	2	болен бронхитом
Dyspnea	2	испытывает удушье

3.5.2 ALARM Данная байесова сеть также очень часто рассматривается для оценки качества различных алгоритмов, работающих с вероятностными сетями. Данная сеть была рассмотрена в работе [3]. Сеть предназначена для медицинской диагностики, и используется для обработки физиологических наблюдений пациента. Сеть состоит из переменных трех типов: диагнозов, физиологических показателей и скрытых переменных, которые измерить на практике нельзя. Данная сеть содержит 37 переменных и 46 связей между ними. Максимальное количество переменных-родителей равно четырём. Рассматриваемая вероятностная сеть относится к сетям средник размеров. Данная сеть хорошо изучена и представляет интерес, как модель для оценки качества реализованных в библиотеке алгоритмов. Структура сети приведена на рисунке 3.2. Из-за довольно большого количества переменных здесь не приводится их описание и назначение. В этой информации нет необходимости для оценки качества реализованных алгоритмов, важно знать общую структуру сети.

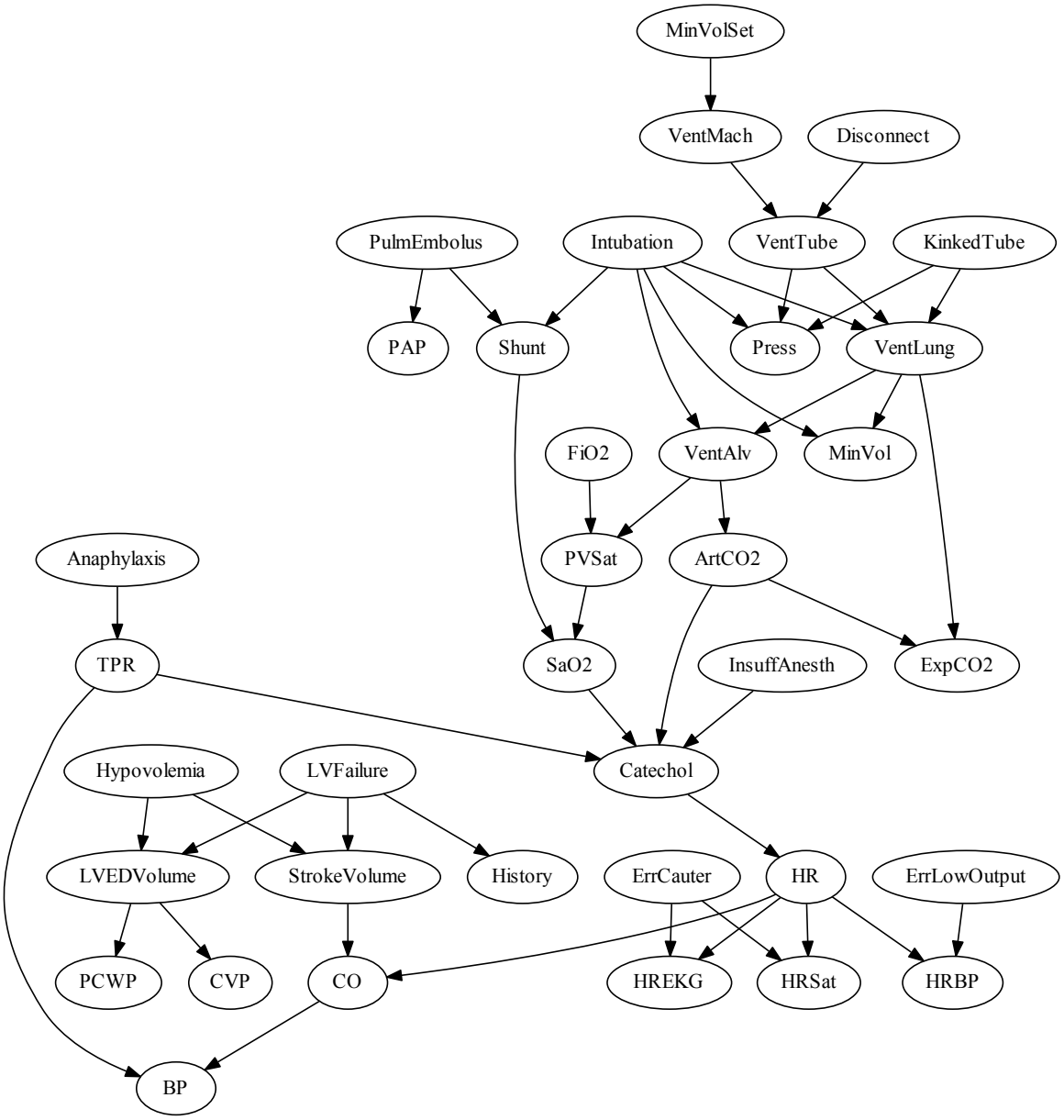


Рисунок 3.2 – Структура байесовой сети ALARM

3.6 Алгоритм на основе оценки апостериорной вероятности структуры

В данном подразделе рассматривается известный алгоритм, использующий оценку апостериорной вероятности в качестве критерия поиска. Подробное описание данной оценки и базового алгоритма поиска приведены в работе [4].

Формулу (??) для оценки совместной вероятности на практике напрямую использовать не получится, без введения дополнительных предполо-

жений. Необходимо сделать предположение, что все возможные структуры равновероятны, т. е. $P(B_S)$ равно некоторой малой константе c. Таким образом нахождение оптимальной структуры сводится к максимизации формулы (3.1), т. е. задача сводится к нахождению множества вершин-предков π_i для каждой вершины X_i, оптимизирующих целевую функцию [4].

$$\max\left[P(B_S, x^R[n])\right] = \\ = c\prod_{i=1}^{R}\max_{\pi_i}\left[\prod_{j=1}^{q_i}\frac{(\alpha_i-1)!}{(n[\phi_i[j],i,B_S]+\alpha_i-1)!}\prod_{k=1}^{\alpha_i}n[v_{ik},\phi_i[j],i,B_S]!\right]. \quad (3.1)$$

Таким образом наивный алгоритм поиска состоит в полном переборе всех возможных родителей для каждой вершины и оптимизации при этом функции (3.2):

$$g(i,\pi_i) = \prod_{j=1}^{q_i}\frac{(\alpha_i-1)!}{(n[\phi_i[j],i,B_S]+\alpha_i-1)!}\prod_{k=1}^{\alpha_i}n[v_{ik},\phi_i[j],i,B_S]!\,. \quad (3.2)$$

На практике наивный вариант не годится из-за большого количества возможных вариантов структур сетей. В разработанной реализации использовались те же ограничения и стратегия поиска, что и в работе [4]. Перед началом выполнения алгоритма требуется знание о порядке вершин, таком, что вершины родители всегда находятся раньше вершин потомков. Схематически алгоритм поиска выглядит следующим образом:

Листинг 3.6 – Псевдокод реализации алгоритма K2

```
function k2 =
  (* Input: dataset x^R[n], ordering of variables, u - maximum number of parents per
      variable.
     Output: for each node, a printout of the parents of the node. *)
  for i in 1 .. R do
    π_i := ∅
    P_old := g(i, π_i); // formula(3.2)
    OkToContinue := true;
    while OkToContinue and |π_i| < u do
      let z = node in Pred(X_i) − π_i that maximizes g(i, π_i ∪ z)
      P_new := f(i, π_i ∪ z)
      if P_new > P_old then
        P_old := P_new;
        π_i := π_i ∪ z
      else OkToContinue := false
    end while
    printfn("Node: ", X_i, "Parents of X_i:", π_i)
  end for
end
```

В разработанной в рамках дипломного проекта реализации за основу был взят алгоритм K2, приведенный в работе [4], псевдокод которого показан в листинге 3.6. В разработанном алгоритме слегка изменен способ подсчета целевой функции. Приняв во внимание ограничение на представление в компьютере вещественных и больших целых чисел, а также то, что операции умножения, деления и возведения в степень более сложные, было принято решение воспользоваться прологарифмированной версией формулы (3.2). Ниже приводится точная формула (3.3), по которой вычисляется оценка в реализованном алгоритме. Данная формула более удобная для вычисления на компьютере:

$$\begin{aligned}
\log(g(i,\pi_i)) &= \sum_{j=1}^{q_i} \log\left(\frac{(\alpha_i-1)!}{(n_{ij}+\alpha_i-1)!}\prod_{k=1}^{\alpha_i} n_{ijk}!\right) = \\
&= \sum_{j=1}^{q_i}\left(\log\frac{(\alpha_i-1)!}{(n_{ij}+\alpha_i-1)!} + \log\prod_{k=1}^{\alpha_i} n_{ijk}!\right) = \\
&= \sum_{j=1}^{q_i}\left(\log(\alpha_i-1)! - \log(n_{ij}+\alpha_i-1)! + \sum_{k=1}^{\alpha_i}\log n_{ijk}!\right) = \\
&= \sum_{j=1}^{q_i}\left(\log\Gamma(\alpha_i) - \log\Gamma(n_{ij}+\alpha_i) + \sum_{k=1}^{\alpha_i}\log\Gamma(n_{ijk}+1)\right) = \\
&= q_i\log\Gamma(\alpha_i) + \sum_{j=1}^{q_i}\left(\sum_{k=1}^{\alpha_i}\log\Gamma(n_{ijk}+1) - \log\Gamma(n_{ij}+\alpha_i)\right),
\end{aligned} \tag{3.3}$$

где Γ — гамма-функция — расширение понятия факториала на поле комплексных чисел;
n_{ijk} — условное, более краткое, обозначение для $n[v_{ik}, \phi_i[j], i, B_S]$;
n_{ij} — условное, более краткое, обозначение для $n[\phi_i[j], i, B_S]$.

Помимо использования формулы (3.3) в реализации были произведены дополнительные оптимизации, продиктованные результатами профилирования реализации алгоритма.

Результаты обучения сетей Asia и ALARM на наборах данных разного объёма реализованным алгоритмом приведены в таблицах 3.3 и 3.4 соот-

ветственно[1)]. Как видно из результатов, с увеличением количества данных улучшается качество извлеченной из данных сети. Стоит обратить внимание, что данный алгоритм потребовал априорных знаний о распределении, по которому были сгенерированы данные. Алгоритму на вход необходим определенный порядок вершин и знание максимального количества переменных-родителей для каждой переменной.

Таблица 3.3 – Качество структуры извлеченной из данных для сети Asia алгоритмом K2 из разработанной библиотеки

Размер данных	Соединения			Время построения
	пропущено	добавлено	инвертировано	
1000	1	1	0	00:00:00.01
2000	1	1	0	00:00:00.02
4000	1	0	0	00:00:00.04
8000	1	1	0	00:00:00.09
16 000	0	0	0	00:00:00.17
32 000	0	0	0	00:00:00.36
64 000	0	0	0	00:00:00.80
1 048 576	0	0	0	00:00:11.41

Таблица 3.4 – Качество структуры извлеченной из данных для сети ALARM алгоритмом K2 из разработанной библиотеки

Размер данных	Соединения			Время построения
	пропущено	добавлено	инвертировано	
1000	1	5	0	00:00:00.75
2000	1	1	0	00:00:00.72
4000	1	0	0	00:00:01.29
8000	1	2	0	00:00:02.45
16 000	0	2	0	00:00:04.83
32 000	0	1	0	00:00:09.48
64 000	0	1	0	00:00:18.61
128 000	0	1	0	00:00:36.58
1 048 576	0	1	0	00:04:57.18
8 388 608	0	1	0	00:44:13.00

Произведём сравнение реализованного алгоритма, с аналогичным алгоритмом реализованным в программе GeNIe, описанной в пункте ?? на странице ??. В таблице 3.5 приводятся полученные результаты.

[1)] Формат времени в колонке «Время построения» — часы:минуты:секунды.милисекунды

Таблица 3.5 – Качество структуры извлеченной из данных для сети Asia программой GeNIe с применением алгоритма Greedy Thick Thinning с оценкой K2

Размер данных	Соединения			Время построения
	пропущено	добавлено	инвертировано	
1000	2	2	2	не измерялось
2000	2	3	2	не измерялось
4000	1	1	3	не измерялось
8000	2	4	2	не измерялось
16 000	2	5	2	не измерялось
32 000	1	4	2	не измерялось
64 000	0	1	3	не измерялось
1 048 576	1	4	2	не измерялось

Для сравнения реализованных в библиотеке алгоритмов с теми, которые есть в GeNIe, были произведены дополнительные испытания последних. Для наиболее «точного» алгоритма реализованного в GeNIe в таблице 3.6 приводится оценка качества полученной структуры на наборах данных различного размера Для трех других алгоритмов в таблице 3.7 — на самом большом наборе данных. Как видно из полученных экспериментально данных, реализация алгоритма на основе оценки апостериорной вероятности из разработанной библиотеки ведет себя лучше как на малых объёмах данных, так и на больших, не смотря на то, что некоторые алгоритмы реализованные в GeNIe используют тот же метод оценки.

Таблица 3.6 – Качество структуры извлеченной из данных для сети Asia программой GeNIe с использованием алгоритма PC

Размер данных	Соединения			Время построения
	пропущено	добавлено	инвертировано	
1000	2	1	2	не измерялось
2000	3	0	0	не измерялось
4000	1	0	0	не измерялось
8000	2	0	0	не измерялось
16 000	3	1	3	не измерялось
32 000	2	0	0	не измерялось
64 000	1	0	0	не измерялось
1 048 576	1	0	0	не измерялось

Таблица 3.7 – Качество структуры извлеченной из данных для сети Asia программой GeNIe на наборе данных из 1 048 576 случаев

Алгоритм	Соединения		
	пропущено	добавлено	инвертировано
Bayesian Search	0	2	5
Essential Graph Search	5	0	2
Greedy Thick Thinning с оценкой BDeu	1	4	2

3.7 Алгоритм на основе оценки минимальной длины описания

Помимо алгоритма использующего оценку апостериорной вероятности в разработанной библиотеке был реализован алгоритм использующий оценку на основе принципа МДО. Описание принципа МДО приведено в подразделе ?? данной пояснительной записки. Т. к. способ подсчета оценки уже был здесь описан, то необходимо привести описание процедуры поиска. Процедура поиска в разработанной реализации алгоритма поиска вдохновлена процедурой поиска, использованной в работе [5].

Данный алгоритм нахождения структуры не требует предварительных знаний об истинном распределении, в отличие от алгоритма описанного в подразделе 3.6, что является существенным преимуществом на практике. Вместо использования априорных знаний, реализация алгоритма используем предварительные вычисления, извлекающие полезные данные о взаимозависимостях между переменным. Затем эта информация используется в стратегии поиска структуры.

В качестве оценки степени зависимости двух произвольных переменных в работе [6] было предложено использовать значение взаимной информации[1)]. Эта информация задаёт приоритет поиска зависимостей между переменными. По своей сути значение обоюдной информации является аналогом корреляции, но по своему содержанию — это оценка количества информации содержащейся в одной переменной о другой [5]. Значение взаимной информации принимает неотрицательные значения и равно нулю в случае независимости случайных величин. Для вычисления взаимной информации была предложена формула (3.4):

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p(x)\, p(y)} \right), \tag{3.4}$$

[1)] В англоязычной литературе используется термин mutual information

где $p(x, y)$ — совместное распределение случайных величин X и Y;
$p(y)$ — безусловное распределение случайной величины X;
$p(x)$ — безусловное распределение случайной величины Y.

На практике при вычислении $I(X;Y)$ следует соблюдать осторожность, т. к. относительные частоты, используемые для оценки $p(x)$, $p(y)$ и $p(x, y)$, могут необоснованно принимать значение 0 из-за отсутствия некоторых состояний переменных в наблюдаемых данных. В таких случаях вместо нуля используется достаточно малое число.

Таким образом алгоритм поиска состоит из следующих шагов:

а) Вычислить значения взаимной информации между всеми парами переменных и отсортировать список уникальных пар по убыванию взаимной информации. Пусть данный список обозначается символом $list$.

б) Алгоритм поиска начинается с извлечения двух пар переменных (X_{i1}, X_{i2}) и (X_{j1}, X_{j2}) из списка $list$ с максимальным значением взаимной информации. Затем среди всех возможных ацикличных моделей, построенных из переменных $X_{i1}, X_{i2}, X_{j1}, X_{j2}$ выбирается модель с наименьшей оценкой, вычисленной по формуле (??). Эта модель принимается за стартовую модель g_0

в) Пока список $list$ не пуст из него извлекается пара переменных (X_{k1}, X_{k2}) и строится множество новых моделей $\{g_0; g_0 \cup (X_{k1}, X_{k2}); g_0 \cup (X_{k2}, X_{k1})\}$. И этого множества удаляются циклические модели и выбирается модель с минимальной оценкой по формуле (??) и присваивается переменной g_0.

г) Когда список $list$ пуст, то поиск прекращается, модель g_0 считается оптимальной и рассматривается как структура вероятностной сети выведенная из данных. На практике можно завершить алгоритм раньше не дожидаясь пустоты $list$.

При реализации данного алгоритма важным моментом для повышения производительности является мемоизация значений функций $n[s, k, g]$ и $n[q, s, k, g]$, т. к. оценка длины описания считается для всей модели сразу, но между двумя последовательными итерациями разница между моделями составляем максимум одно соединение, т. е. для большинства вершин множество предков не меняется и значения $n[s, k, g]$ и $n[q, s, k, g]$ остаются неизменными и их можно не вычислять каждый раз. Так, для сети ALARM на экспериментальных данных из 1 048 576 случаев, время построения сети сократилось с более чем четырех часов, до двух с половиной минут. Ниже приведены результаты качества обучения данного алгоритма для сети

Asia, в табилце 3.8, и для сети ALARM, в таблице 3.9.

Таблица 3.8 – Качество структуры извлеченной из данных для сети Asia алгоритмом из разработанной библиотеки, использующим оценку МДО

Размер данных	Соединения			Время построения
	пропущено	добавлено	инвертировано	
1000	2	1	0	00:00:00.26
2000	2	2	2	00:00:00.28
4000	1	2	2	00:00:00.47
8000	1	1	0	00:00:00.93
16 000	1	2	0	00:00:01.70
32 000	0	1	1	00:00:03.37
64 000	0	1	0	00:00:07.05
1 048 576	0	1	0	00:01:46.53

Таблица 3.9 – Качество структуры извлеченной из данных для сети ALARM алгоритмом из разработанной библиотеки, использующим оценку МДО

Размер данных	Соединения			Время построения
	пропущено	добавлено	инвертировано	
1000	6	3	11	00:00:01.78
2000	5	5	11	00:00:01.50
4000	3	4	11	00:00:01.84
8000	3	7	9	00:00:02.26
16 000	2	10	17	00:00:03.27
32 000	2	13	19	00:00:05.17
64 000	1	17	15	00:00:09.30
128 000	1	21	21	00:00:18.50
1 048 576	0	22	16	00:02:22.82

Как видно данный алгоритм работает хуже алгоритма из подраздела 3.6. По результатам измерений, приведенных в табилце 3.9 можно видеть, что с увеличением набора данных структура сети усложняется. Количество пропущенных связей уменьшается, но также растет количество лишних и инвертированных связей. На практике возможна доработка структуры сети с участием эксперта. Это займет меньше времени, чем разработка сети «с нуля», т. к. большинство связей и общая структура уже найдены. В защиту данного алгоритма можно сказать, что он работает

без каких-либо априорных знаний о структуре сети и не ограничен максимальным количеством вершин-предков, как алгоритм из предыдущего подраздела. Также можно заметить, что не смотря на то, что алгоритм проигрывает реализации алгоритма K2, качество обучаемой структуры в многих случаях лучше того, что может предоставить программа GeNIe. Сравнение с различными алгоритмами GeNIe приведено в таблице 3.10, необходимо отметить, что алгоритмы из GeNIe субъективно работают в разы медленнее разработанной библиотеки на данных из 1 048 576 случаев для сети ALARM. Например, выполнение алгоритма Bayesian Search заняло более четырёх часов.

Таблица 3.10 – Качество структуры извлеченной из данных для сети ALARM программой GeNIe на наборе данных из 1 048 576 случаев

Алгоритм	Соединения		
	пропущено	добавлено	инвертировано
Bayesian Search	8	66	21
PC	создал циклическую структуру		
Essential Graph Search	32	2	4
Greedy Thick Thinning с оценкой BDeu	0	26	24
Greedy Thick Thinning с оценкой K2	0	30	23

В процессе оценки результатов данного алгоритма было выявлено экспериментальным путём, что оценка на основе МДО довольная чувствительна к данным, т. е. имея два набора данных одинакового размера, сгенерированных одним распределением, можно получить немного разные сети из-за случайных различий в данных. В связи с этим была проделана следующая модификация в существующем алгоритме, которая позволила получать улучшенные сети на малых объёмах данных. Когда на вход алгоритму подается набор данных из n случаев, то алгоритм случайным образом отбрасывает из него небольшой процент данных и строит структуру сети на уменьшенном объёме данных. Далее указанная операция повторяется некоторое количество раз. В итоге получается некоторое множество сетей из которых выбирается лучшая, используя оценку МДО и исходный набор экспериментальных данных. Результаты работы модифицированного алгоритма приведены в таблицах 3.11 и 3.12. Как видно из результатов, в некоторых случаях результаты незначительно улучшились, в некоторых — ухудшились. С учетом времени работы алгоритмов вопрос о целесообраз-

ности данной модификации остается открытым.

Таблица 3.11 – Качество структуры извлеченной из данных для сети Asia модифицированным алгоритмом из разработанной библиотеки, использующим оценку МДО

Размер данных	Соединения			Время построения
	пропущено	добавлено	инвертировано	
1000	1	0	1	00:00:00.92
2000	1	0	0	00:00:00.36
4000	0	0	0	00:00:00.42
8000	0	0	1	00:00:00.67
16 000	1	2	0	00:00:01.44
32 000	0	1	0	00:00:02.25
64 000	0	1	0	00:00:03.90
1 048 576	0	1	0	00:01:06.87

Таблица 3.12 – Качество структуры извлеченной из данных для сети ALARM модифицированным алгоритмом из разработанной библиотеки, использующим оценку МДО

Размер данных	Соединения			Время построения
	пропущено	добавлено	инвертировано	
1000	7	6	10	00:00:11.21
2000	3	8	10	00:00:08.22
4000	3	8	15	00:00:09.89
8000	3	7	11	00:00:13.00
16 000	1	7	15	00:00:18.10
32 000	2	12	9	00:00:31.90
64 000	0	8	17	00:00:51.33
128 000	0	11	16	00:01:33.80
1 048 576	0	15	11	00:13:51.61

Также, вероятно, стоит упомянуть что в разработанной библиотеке реализован еще один алгоритм, использующий оценку МДО, но с использованием другой стратегии поиска и другого способа вычисления длинны описания. Данный алгоритм и оценка были позаимствованы из работы [7]. Реализация данного алгоритма проявила себя хуже, чем предыдущие два рассмотренных алгоритма и поэтому детальная информация по данному алгоритму здесь не приводится.

В качестве промежуточного итога для данного раздела стоит отметить, что два известных алгоритма и одна модификация, реализованные в библиотеке, показывают результаты лучше как по качеству, так и по времени, чем все алгоритмы, представленные в бесплатной программе GeNIe. Из-за лицензионных ограничений сравнить реализованные алгоритмы с другим коммерческим ПО не удалось.

4 ОХРАНА ТРУДА

4.1 Обеспечение пожарной безопасности на предприятии малого бизнеса «Техартгруп»

Целью дипломного проекта является реализация и анализ алгоритмов построения вероятностных сетей. Вероятностная сеть является компактным и эффективным способом представления знаний. Вероятностные сети используются в программном обеспечении для принятия решения в условиях недостаточной определенности. Данный способ статистического моделирования показал свою пригодность в реальных условиях в сложных предметных областях: медицине, космической промышленности, финансовой сфере и других областях. Первоначальные стадии разработки дипломного проекта выполнялись на предприятии ООО «Техартгруп» во время прохождения преддипломной практики. В настоящем разделе рассматриваются вопросы, связанные с обеспечением пожарной безопасности на предприятии.

Предприятие «Техартгруп» занимается предоставлением услуг по разработке информационных систем для иностранных предприятий. В минском офисе компании на данный момент работает более 200 человек. Большое количество конкурирующих компаний, разрабатывающих программное обеспечение в Минске, способствует повышению общего уровня условий труда. Это, в частности, сказывается на комфортабельности рабочих мест. Работникам предоставляются светлые, проветриваемые, тихие кабинеты, гибкий график рабочего времени, специальные комнаты отдыха и т. д. Современные компании негласно ориентируются на соответствие лучшим мировым практикам в области охраны труда и, в частности, пожарной безопасности.

На предприятии «Техартгруп» за пожарную безопасность отвечает директор компании. Для каждого нового сотрудника производится инструктаж по пожарной безопасности и технике безопасности, а так же знакомство с планом эвакуации при возникновении черезвычайных ситуаций [8, с. 324]. За проведение инструктажа отвечает специальный человек из отдела материально-технического снабжения предприятия. В компании действует набор правил, обязательных для исполнения сотрудниками. В целях повышения пожарной безопасности курение в здании офиса запрещено. Все сотрудники обязаны в конце рабочего дня выключить свои персональные компьютеры и обесточить их. В конце рабочего дня специальный

сотрудник проверяет соблюдение данного правила в каждом рабочем кабинете, чтобы там были выключены все электрические приборы: компьютеры, электрические чайники, кондиционеры, освещение и т. д. Все рабочие компьютеры подключены к источникам бесперебойного питания, которые подключены к сетевыми фильтрам, защищающим от скачков напряжения в электросети.

Офис компании расположен в центре города. Здание офиса представляет собой монолитную железобетонную конструкцию высотой шесть этажей, офис компании находится на двух верхних этажах. Конструкция здания предусматривает три способа эвакуации с этажа: выход в паркинг, лестничная клетка с выходом на улицу, лестничная клетка с выходом на первый этаж паркинга. В случае недоступности основных эвакуационных выходов из каждого кабинета можно через окно попасть на лоджию [8, с. 314 – 316]. Схемы эвакуации выдаются в виде электронного документа каждому новому сотруднику, а также находятся на специальном стенде в рабочих кабинетах. Все кабинеты офиса расположены вдоль длинного коридора, который оборудован специальными аварийными светильниками и знаками, указывающими направление эвакуации. На случай отключения электроэнергии компания имеет два дизельных-генератора, обеспечивающих нужды предприятия на случай отключения электроэнергии.

Офис компании оборудован необходимыми средствами сигнализации о пожаре [9, с. 215]. Каждый кабинет оборудован пожарным дымовым оптико-электрическим точечным извещателем ИП212-02М1 (рисунок 4.1). На предприятии производиться регулярный контроль и проверка работоспособности пожарных извещателей специальным человеком из отдела материально-технического снабжения предприятия. В коридорах дополнительно установлены ручные пожарные извещатели ИП 5-2Р (рисунок 4.1). Для извещения о пожаре также может быть использована корпоративная электронная почта, а также другие современные способы обмена информацией.

На случай возникновения пожара в каждом рабочем кабинете находиться ручной порошковый огнетушитель ОП-10 (з) МИГ М (рисунок 4.2), пригодный для тушения пожаров различного типа, в том числе для тушения электрических приборов [8, с. 221 – 323]. Каждый этаж здания офиса оборудован двумя пожарными кранами для тушения пожара. Пожарные краны расположены в противоположных частях коридора, недалеко от эвакуационных выходов (рисунок 4.2). На случай воспламенения электрической проводки или другого электрического оборудования в каждом кабинете установлены электрические щитки, необходимые для отключения

(а)

(б)

Рисунок 4.1 – а — автономный пожарный извещатель; б — ручной пожарный извещатель.

подачи электроэнергии в пределах кабинета. Во всех помещениях офиса предприятия установлена оросительная система пожаротушения для ликвидации возгорания до приезда пожарной службы [8, с. 318 – 320]. При расследовании возможных причин возникновения пожара может быть задействована система видео-наблюдения, установленная во всех помещениях предприятия.

(а)

(б)

Рисунок 4.2 – а — порошковый огнетушитель ОП-10 (з) МИГ М; б — пожарный кран.

Основной род деятельности на предприятии — разработка информационных систем — не предусматривает непосредственный контакт с горючими или легко-воспламеняющимися веществами, что сильно снижает рис-

ки возникновения пожара на предприятии. Наиболее вероятными причинами возникновения пожара, с учетом специфики предприятия, могут являться нарушение правил внутреннего распорядка — курение на рабочем месте, и неисправность электрического оборудования, которого в офисе компании достаточно [8, с. 312]. С целью снижения риска возникновения пожара по причине неисправности электрического оборудования в компании запрещено пользоваться неисправным оборудованием, а все исправное оборудование подключается в сеть через специальные сетевые фильтры и источники бесперебойного питания. В целом правила распорядка на предприятии и высокая культура работы с электрическим оборудованием снижают риски возникновения пожара до минимума.

Большой проблемой в достижении максимальной пожарной безопасности предприятия является доступность подъезда пожарной техники к зданию офиса. В будние дни прилегающие улицы, стоянки, пешеходные переходы заняты неправильно припаркованным личным транспортом. Большую часть светлого времени суток движение по прилегающим улицам очень затруднено. Данную проблему предприятие не в силах решить самостоятельно, проблема заключается в низкой культуре владельцев транспорта и игнорировании многочисленных нарушений правил дорожного движения сотрудниками ГАИ. Таким образом, изложенные выше предложения, не смотря на проблемы с подъездом пожарной техники, обеспечат пожарную безопасность на предприятии «Техартгруп».

5 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ЭФФЕКТИВНОСТИ СЕРВИСА, ПРЕДОСТАВЛЯЮЩЕГО ВОЗМОЖНОСТЬ ПОЛУЧЕНИЯ НОВИНОК МУЗЫКАЛЬНОЙ ИНДУСТРИИ

5.1 Введение и исходные данные

Целью данной дипломной работы является создание программного продукта, позволяющего пользователю получать новинки музыкальной индустрии в виде уведомлений о недавно вышедших публикациях выбранных им музыкальных исполнителей. Данный программный продукт позволит получать информацию из нескольких источников, тем самым помогая пользователю удобнее ориентироваться среди различных сервисов, получать только необходимую и интересную информацию. Это очень эффективно, поскольку сокращается время на поиски новых публикаций самостоятельно. В экономическом смысле - это сократит расходы на подписки на несколько сервисов, которые занимаются размещением на своей платформе новых музыкальных публикаций.

Для разработчика экономическая эффективность будет определяться как прибыль от реализации, а для пользователя - как экономия затрат. Для определения экономической эффективности рассчитаем смету затрат и цену программного продукта.

5.2 Расчёт сметы затрат и цены программного продукта

Для представления программного продукта на рынке он должен быть законченным, иметь презентабельный вид. Для реализации программного продукта необходимо пройти через два этапа: этап, связанный с разработкой ПО(выяснение требований, программирование, тестирование, отладка), и этап, связанный с непосредственной реализацией продукта на рынке (реализация, поддержка).

Программный комплекс относится ко 2-й группе сложности. Категория новизны – “В”. Расчеты выполнены на основе методического пособия [1].

Исходные данные для разрабатываемого проекта указаны в таблице 5.1.

Таблица 5.1 – Исходные данные

Наименование	Условное обозначение	Значение
Категория сложности		2
Коэффициент сложности, ед.	$К_с$	1,12
Степень использования при разработке стандартных модулей, ед.	$К_т$	0,7
Коэффициент новизны, ед.	$К_н$	0,7
Годовой эффективный фонд времени, дн.	$Ф_{эф}$	231
Продолжительность рабочего дня, ч.	$Т_ч$	8
Месячная тарифная ставка первого разряда, Br	$Т_{м_1}$	600 000
Коэффициент премирования, ед.	К	1,5
Норматив дополнительной заработной платы, ед.	$Н_д$	20
Норматив отчислений в ФСЗН и обязательное страхование, %	$Н_{сз}$	34,5
Норматив командировочных расходов, %	$Н_к$	15
Норматив прочих затрат, %	$Н_{пз}$	20
Норматив накладных расходов, %	$Н_{рн}$	100
Прогнозируемый уровень рентабельности, %	$У_{рп}$	35
Норматив НДС, %	$Н_{дс}$	20
Норматив налога на прибыль, %	$Н_п$	18
Норматив расхода материалов, %	$Н_{мз}$	3
Норматив расхода машинного времени, ч.	$Н_{мв}$	4,5
Цена одного часа машинного времени, Br	$Н_{мв}$	5000
Норматив расходов на сопровождение и адаптацию ПО, %	$Н_{рса}$	30

На основании сметы затрат и анализа рынка ПО определяется плановая отпускаемая цена. Для составления сметы затрат на создание ПО необходима предварительная оценка трудоемкости ПО и его объёма. Расчет объёма программного продукта (количества строк исходного кода) предполагает определение типа программного обеспечения, всестороннее техническое обоснование функций ПО и определение объёма каждой функций. Согласно классификации типов программного обеспечения [10, с. 59, приложение 1], разрабатываемое ПО с наименьшей ошибкой можно классифицировать как ПО методо-ориентированных расчетов.

Общий объём программного продукта определяется исходя из количества и объёма функций, реализованных в программе:

$$V_o = \sum_{i=1}^{n} V_i \,, \tag{5.1}$$

где V_i — объём отдельной функции ПО, LoC;
n — общее число функций.

На стадии технико-экономического обоснования проекта рассчитать точный объём функций невозможно. Вместо вычисления точного объёма функций применяются приблизительные оценки на основе данных по аналогичным проектам или по нормативам [10, с. 61, приложение 2], которые приняты в организации.

Таблица 5.2 – Перечень и объём функций программного модуля

№ функции	Наименование (содержание)	Объём функции, LoC	
		по каталогу (V_i)	уточненный ($V_i^{\text{у}}$)
101	Организация ввода информации	100	60
102	Контроль, предварительная обработка и ввод информации	520	520
111	Управление вводом/выводом	2700	700
304	Обслуживание файлов	520	580
305	Обработка файлов	750	750
309	Формирование файла	1100	1100
506	Обработка ошибочных и сбойных ситуаций	430	430
507	Обеспечение интерфейса между компонентами	730	730
605	Вспомогательные и сервисные программы	460	280
701	Математическая статистика и прогнозирование	8370	3500
Итог		15 680	8650

Каталог аналогов программного обеспечения предназначен для предварительной оценки объёма ПО методом структурной аналогии. В разных

организациях в зависимости от технических и организационных условий, в которых разрабатывается ПО, предварительные оценки могут корректироваться на основе экспертных оценок. Уточненный объём ПО рассчитывается по формуле:

$$V_{\text{у}} = \sum_{i=1}^{n} V_i^{\text{у}}\,, \tag{5.2}$$

где $V_i^{\text{у}}$ — уточненный объём отдельной функции ПО, LoC;
n — общее число функций.

Перечень и объём функций программного модуля перечислен в таблице 5.2. По приведенным данным уточненный объём некоторых функций изменился, и общий объём ПО составил $V_o = 15\,680\,\text{LoC}$, общий уточненный общем ПО — $V_{\text{у}} = 8650\,\text{LoC}$.

По уточненному объёму ПО и нормативам затрат труда в расчете на единицу объёма определяются нормативная и общая трудоемкость разработки ПО. Уточненный объём ПО — 8650 LoC. ПО относится ко второй категории сложности: предполагается его использование для сложных статистических расчетов и решения задач классификации, также необходимо обеспечить переносимость ПО [10, с. 66, приложение 4, таблица П.4.1]. По полученным данным определяется нормативная трудоемкость разработки ПО. Согласно укрупненным нормам времени на разработку ПО в зависимости от уточненного объёма ПО и группы сложности ПО [10, с. 64, приложение 3] нормативная трудоемкость разрабатываемого проекта составляет $\text{Т}_{\text{н}} = 224\,\text{чел./дн}$.

Нормативная трудоемкость служит основой для оценки общей трудоемкости $\text{Т}_{\text{о}}$. Используем формулу (5.3) для оценки общей трудоемкости для небольших проектов:

$$\text{Т}_{\text{о}} = \text{Т}_{\text{н}} \cdot \text{К}_{\text{с}} \cdot \text{К}_{\text{т}} \cdot \text{К}_{\text{н}}\,, \tag{5.3}$$

где $\text{К}_{\text{с}}$ — коэффициент, учитывающий сложность ПО;
$\text{К}_{\text{т}}$ — поправочный коэффициент, учитывающий степень использования при разработке стандартных модулей;
$\text{К}_{\text{н}}$ — коэффициент, учитывающий степень новизны ПО.

Дополнительные затраты труда на разработку ПО учитываются че-

рез коэффициент сложности, который вычисляется по формуле

$$\mathrm{K_c} = 1 + \sum_{i=1}^{n} \mathrm{K}_i \,, \tag{5.4}$$

где K_i — коэффициент, соответствующий степени повышения сложности ПО за счет конкретной характеристики;
n — количество учитываемых характеристик.

Наличие двух характеристик сложности позволяет [10, с. 66, приложение 4, таблица П.4.2] вычислить коэффициент сложности

$$\mathrm{K_c} = 1 + 0{,}12 = 1{,}12 \,. \tag{5.5}$$

Разрабатываемое ПО использует стандартные компоненты. Степень использования стандартных компонентов определяется коэффициентом использования стандартных модулей — $\mathrm{K_т}$. Согласно справочным данным [10, с. 68, приложение 4, таблица П.4.5] указанный коэффициент для разрабатываемого приложения $\mathrm{K_т} = 0{,}7$. Трудоемкость создания ПО также зависит от его новизны и наличия аналогов. Разрабатываемое ПО не является новым, существуют аналогичные более зрелые разработки у различных компаний и университетов по всему миру. Влияние степени новизны на трудоемкость создания ПО определяется коэффициентом новизны — $\mathrm{K_н}$. Согласно справочным данным [10, с. 67, приложение 4, таблица П.4.4] для разрабатываемого ПО $\mathrm{K_н} = 0{,}7$. Подставив приведенные выше коэффициенты для разрабатываемого ПО в формулу (5.3) получим общую трудоемкость разработки

$$\mathrm{T_o} = 224 \times 1{,}12 \times 0{,}7 \times 0{,}7 \approx 123 \,\text{чел./дн.} \tag{5.6}$$

На основе общей трудоемкости и требуемых сроков реализации проекта вычисляется плановое количество исполнителей. Численность исполнителей проекта рассчитывается по формуле:

$$\mathrm{Ч_р} = \frac{\mathrm{T_o}}{\mathrm{T_р} \cdot \Ф_{\text{эф}}} \,, \tag{5.7}$$

где $T_о$ — общая трудоемкость разработки проекта, чел./дн.;
$Ф_{эф}$ — эффективный фонд времени работы одного работника в течение года, дн.;
$T_р$ — срок разработки проекта, лет.

Эффективный фонд времени работы одного разработчика вычисляется по формуле

$$Ф_{эф} = Д_г - Д_п - Д_в - Д_о\,, \tag{5.8}$$

где $Д_г$ — количество дней в году, дн.;
$Д_п$ — количество праздничных дней в году, не совпадающих с выходными днями, дн.;
$Д_в$ — количество выходных дней в году, дн.;
$Д_п$ — количество дней отпуска, дн.

Согласно данным, приведенным в производственном календаре для пятидневной рабочей недели в 2013 году для Беларуси [11], фонд рабочего времени составит

$$Ф_{эф} = 365 - 9 - 104 - 21 = 231\,\text{дн.} \tag{5.9}$$

Учитывая срок разработки проекта $T_р = 3\,\text{мес.} = 0{,}25\,\text{года}$, общую трудоемкость и фонд эффективного времени одного работника, вычисленные ранее, можем рассчитать численность исполнителей проекта

$$Ч_р = \frac{123}{0{,}25 \times 231} \approx 2\,\text{рабочих.} \tag{5.10}$$

Вычисленные оценки показывают, что для выполнения запланированного проекта в указанные сроки необходимо два рабочих. Информация о работниках перечислена в таблице 5.3.

Таблица 5.3 – Работники, занятые в проекте

Исполнители	Разряд	Тарифный коэффициент	Чел./дн. занятости
Программист I-категории	13	3,04	61
Ведущий программист	15	3,48	62

Месячная тарифная ставка одного работника вычисляется по фор-

муле

$$\text{Т}_\text{ч} = \frac{\text{Т}_{\text{м}_1} \cdot \text{Т}_\text{к}}{\Phi_\text{р}}\,, \tag{5.11}$$

где $\text{Т}_{\text{м}_1}$ — месячная тарифная ставка 1-го разряда, Br;
$\text{Т}_\text{к}$ — тарифный коэффициент, соответствующий установленному тарифному разряду;
$\Phi_\text{р}$ — среднемесячная норма рабочего времени, час.

Подставив данные из таблицы 5.3 в формулу (5.11), приняв значение тарифной ставки 1-го разряда $\text{Т}_{\text{м}_1} = 600\,000\,\text{Br}$ и среднемесячную норму рабочего времени $\Phi_\text{р} = 160$ часов получаем

$$\text{Т}_\text{ч}^{\text{прогр. I-разр.}} = \frac{600\,000 \times 3{,}04}{160} = 11\,400\,\text{Br/час}; \tag{5.12}$$

$$\text{Т}_\text{ч}^{\text{вед. прогр.}} = \frac{600\,000 \times 3{,}48}{160} = 13\,050\,\text{Br/час}. \tag{5.13}$$

Основная заработная плата исполнителей на конкретное ПО рассчитывается по формуле

$$\text{З}_\text{о} = \sum_{i=1}^{n} \text{Т}_\text{ч}^i \cdot \text{Т}_\text{ч} \cdot \Phi_\text{п} \cdot \text{К}\,, \tag{5.14}$$

где $\text{Т}_\text{ч}^i$ — часовая тарифная ставка i-го исполнителя, Br/час;
$\text{Т}_\text{ч}$ — количество часов работы в день, час;
$\Phi_\text{п}$ — плановый фонд рабочего времени i-го исполнителя, дн.;
К — коэффициент премирования.

Подставив ранее вычисленные значения и данные из таблицы 5.3 в формулу (5.14) и приняв коэффициент премирования $\text{К} = 1{,}5$ получим

$$\text{З}_\text{о} = (11400 \times 61 + 13050 \times 62) \times 8 \times 1{,}5 = 18\,054\,000\,\text{Br}\,. \tag{5.15}$$

Дополнительная заработная плата включает выплаты предусмотренные законодательством от труде и определяется по нормативу в процентах от основной заработной платы

$$\text{З}_\text{д} = \frac{\text{З}_\text{о} \cdot \text{Н}_\text{д}}{100\%}\,, \tag{5.16}$$

где $Н_д$ — норматив дополнительной заработной платы, %.

Приняв норматив дополнительной заработной платы $Н_д = 20\%$ и подставив известные данные в формулу (5.16) получим

$$З_д = \frac{18\,054\,000 \times 20\%}{100\%} \approx 3\,610\,800\,\text{Br}. \tag{5.17}$$

Согласно действующему законодательству отчисления в фонд социальной защиты населения составляют 34% , в фонд обязательного страхования — 0,5%, от фонда основной и дополнительной заработной платы исполнителей. Общие отчисления на социальную защиту рассчитываются по формуле

$$З_{сз} = \frac{(З_о + З_д) \cdot Н_{сз}}{100\%}. \tag{5.18}$$

Подставив вычисленные ранее значения в формулу (5.18) получаем

$$З_{сз} = \frac{(18\,054\,000 + 3\,610\,800) \times 34{,}5\%}{100\%} \approx 7\,474\,356\,\text{Br}. \tag{5.19}$$

По статье «материалы» проходят расходы на носители информации, бумагу, краску для принтеров и другие материалы, используемые при разработке ПО. Норма расходов $Н_{мз}$ определяется либо в расчете на 100 строк исходного кода, либо в процентах к основной зарплате исполнителей 3% — 5%. Затраты на материалы вычисляются по формуле

$$М = \frac{З_о \cdot Н_{мз}}{100\%} = \frac{18\,054\,000 \times 3\%}{100\%} \approx 541\,620\,\text{Br}. \tag{5.20}$$

Расходы по статье «машинное время» включают оплату машинного времени, необходимого для разработки и отладки ПО, которое определяется по нормативам в машино-часах на 100 строк исходного кода в зависимости от характера решаемых задач и типа ПК, и вычисляются по формуле

$$Р_м = Ц_м \cdot \frac{V_о}{100} \cdot Н_{мв}, \tag{5.21}$$

где $Ц_м$ — цена одного часа машинного времени, Br;
$Н_{мв}$ — норматив расхода машинного времени на отладку 100 строк исходного кода, часов.

Согласно нормативу [10, с. 69, приложение 6] норматив расхода ма-

шинного времени на отладку 100 строк исходного кода составляет $Н_{мв} = 15$, применяя понижающий коэффициент 0,3 получаем $Н'_{мв} = 4{,}5$. Цена одного часа машинного времени составляет $Ц_{м} = 5000$ Br. Подставляя известные данные в формулу (5.21) получаем

$$Р_{м} = 5000 \times \frac{8650}{100} \times 4{,}5 = 1\,946\,250\,\text{Br}\,. \tag{5.22}$$

Расходы по статье «научные командировки» вычисляются как процент от основной заработной платы, либо определяются по нормативу. Вычисления производятся по формуле

$$Р_{к} = \frac{З_{о} \cdot Н_{к}}{100\%}\,, \tag{5.23}$$

где $Н_{к}$ — норматив командировочных расходов по отношению к основной заработной плате, %.

Подставляя ранее вычисленные значения в формулу (5.23) и приняв значение $Н_{к} = 15\%$ получаем

$$Р_{к} = \frac{18\,054\,000 \times 15\%}{100\%} = 2\,708\,100\,\text{Br}\,. \tag{5.24}$$

Статья расходов «прочие затраты» включает в себя расходы на приобретение и подготовку специальной научно-технической информации и специальной литературы. Затраты определяются по нормативу принятому в организации в процентах от основной заработной платы и вычисляются по формуле

$$П_{з} = \frac{З_{о} \cdot Н_{пз}}{100\%}\,, \tag{5.25}$$

где $Н_{пз}$ — норматив прочих затрат в целом по организации, %.

Приняв значение норматива прочих затрат $Н_{пз} = 20\%$ и подставив вычисленные ранее значения в формулу (5.25) получаем

$$П_{з} = \frac{18\,054\,000 \times 20\%}{100\%} = 3\,610\,800\,\text{Br}\,. \tag{5.26}$$

Статья «накладные расходы» учитывает расходы, необходимые для содержания аппарата управления, вспомогательных хозяйств и опытных производств, а также расходы на общехозяйственные нужны. Данная ста-

тья затрат рассчитывается по нормативу от основной заработной платы и вычисляется по формуле.

$$\text{Р}_\text{н} = \frac{\text{З}_\text{о} \cdot \text{Н}_\text{рн}}{100\%}\,, \tag{5.27}$$

где $\text{Н}_\text{рн}$ — норматив накладных расходов в организации, %.

Приняв норму накладных расходов $\text{Н}_\text{рн} = 100\%$ и подставив известные данные в формулу (5.27) получаем

$$\text{Р}_\text{н} = \frac{18\,054\,000 \times 100\%}{100\%} = 18\,054\,000\,\text{Br}\,. \tag{5.28}$$

Общая сумма расходов по смете на ПО рассчитывается по формуле

$$\text{С}_\text{р} = \text{З}_\text{о} + \text{З}_\text{д} + \text{З}_\text{сз} + \text{М} + \text{Р}_\text{м} + \text{Р}_\text{нк} + \text{П}_\text{з} + \text{Р}_\text{н}\,. \tag{5.29}$$

Подставляя ранее вычисленные значения в формулу (5.29) получаем

$$\text{С}_\text{р} = 55\,999\,926\,\text{Br}\,. \tag{5.30}$$

Расходы на сопровождение и адаптацию, которые несет производитель ПО, вычисляются по нормативу от суммы расходов по смете и рассчитываются по формуле

$$\text{Р}_\text{са} = \frac{\text{С}_\text{р} \cdot \text{Н}_\text{рса}}{100\%}\,, \tag{5.31}$$

где $\text{Н}_\text{рса}$ — норматив расходов на сопровождение и адаптацию ПО, %.

Приняв значение норматива расходов на сопровождение и адаптацию $\text{Н}_\text{рса} = 30\%$ и подставив ранее вычисленные значения в формулу (5.31) получаем

$$\text{Р}_\text{са} = \frac{55\,999\,926 \times 30\%}{100\%} \approx 16\,799\,978\,\text{Br}\,. \tag{5.32}$$

Полная себестоимость создания ПО включает сумму затрат на разработку, сопровождение и адаптацию и вычисляется по формуле

$$\text{С}_\text{п} = \text{С}_\text{р} + \text{Р}_\text{са}\,. \tag{5.33}$$

Подставляя известные значения в формулу (5.33) получаем

$$\text{С}_\text{п} = 55\,999\,926 + 16\,799\,978 = 72\,799\,904\,\text{Br}\,. \tag{5.34}$$

5.3 Расчёт экономической эффективности у разработчика

Важная задача при выборе проекта для финансирования это расчет экономической эффективности проектов и выбор наиболее выгодного проекта. Разрабатываемое ПО является заказным, т. е. разрабатывается для одного заказчика на заказ. На основании анализа рыночных условий и договоренности с заказчиком об отпускной цене прогнозируемая рентабельность проекта составит $\text{У}_\text{рп} = 35\%$. Прибыль рассчитывается по формуле

$$\text{П}_\text{с} = \frac{\text{С}_\text{п} \cdot \text{У}_\text{рп}}{100\%}\,, \tag{5.35}$$

где $\text{П}_\text{с}$ — прибыль от реализации ПО заказчику, Br;
$\text{У}_\text{рп}$ — уровень рентабельности ПО, %.

Подставив известные данные в формулу (5.35) получаем прогнозируемую прибыль от реализации ПО

$$\text{П}_\text{с} = \frac{72\,799\,904 \times 35\%}{100\%} \approx 25\,479\,966\,\text{Br}\,. \tag{5.36}$$

Прогнозируемая цена ПО без учета налогов включаемых в цену вычисляется по формуле

$$\text{Ц}_\text{п} = \text{С}_\text{п} + \text{П}_\text{с}\,. \tag{5.37}$$

Подставив данные в формулу (5.37) получаем цену ПО без налогов

$$\text{Ц}_\text{п} = 72\,799\,904 + 25\,479\,966 = 98\,279\,870\,\text{Br}\,. \tag{5.38}$$

Налог на добавленную стоимость рассчитывается по формуле

$$\text{НДС} = \frac{\text{Ц}_\text{п} \cdot \text{Н}_\text{дс}}{100\%}\,, \tag{5.39}$$

где $\text{Н}_\text{дс}$ — норматив НДС, %.

Норматив НДС составляет $\text{Н}_\text{дс} = 20\%$, подставляя известные значе-

ния в формулу (5.39) получаем

$$\text{НДС} = \frac{98\,279\,870 \times 20\%}{100\%} \approx 19\,655\,974\,\text{Br}\,. \tag{5.40}$$

Расчет прогнозируемой отпускной цены осуществляется по формуле

$$\text{Ц}_\text{о} = \text{Ц}_\text{п} + \text{НДС}\,. \tag{5.41}$$

Подставляя известные данные в формулу (5.41) получаем прогнозируемую отпускную цену

$$\text{Ц}_\text{о} = 98\,279\,870 + 19\,655\,974 \approx 117\,935\,844\,\text{Br}\,. \tag{5.42}$$

Чистую прибыль от реализации проекта можно рассчитать по формуле

$$\text{П}_\text{ч} = \text{П}_\text{с} \cdot \left(1 - \frac{\text{Н}_\text{п}}{100\%}\right)\,, \tag{5.43}$$

где $\text{Н}_\text{п}$ — величина налога на прибыль, %.

Приняв значение налога на прибыль $\text{Н}_\text{н} = 18\%$ и подставив известные данные в формулу (5.43) получаем чистую прибыль

$$\text{П}_\text{ч} = 25\,479\,966 \times \left(1 - \frac{18\%}{100\%}\right) = 20\,893\,572\,\text{Br}\,. \tag{5.44}$$

Программное обеспечение разрабатывалось для одного заказчика в связи с этим экономическим эффектом разработчика будет являться чистая прибыль от реализации $\text{П}_\text{ч}$. Рассчитанные данные приведены в таблице 5.4. Таким образом было произведено технико-экономическое обоснование разрабатываемого проекта, составлена смета затрат и рассчитана прогнозируемая прибыль, и показана экономическая целесообразность разработки.

Таблица 5.4 – Рассчитанные данные

Наименование	Условное обозначение	Значение
Нормативная трудоемкость, чел./дн.	$Т_н$	224
Общая трудоемкость разработки, чел./дн.	$Т_о$	123
Численность исполнителей, чел.	$Ч_р$	2
Часовая тарифная ставка программиста I-разряда, Br/ч.	$Т_ч^{прогр.\ I\text{-}разр.}$	11 400
Часовая тарифная ставка ведущего программиста, Br/ч.	$Т_ч^{вед.\ прогр.}$	13 050
Основная заработная плата, Br	$З_о$	18 054 000
Дополнительная заработная плата, Br	$З_д$	3 610 800
Отчисления в фонд социальной защиты, Br	$З_{сз}$	7 474 356
Затраты на материалы, Br	М	541 620
Расходы на машинное время, Br	$Р_м$	1 946 250
Расходы на командировки, Br	$Р_к$	2 708 100
Прочие затраты, Br	$П_з$	3 610 800
Накладные расходы, Br	$Р_н$	18 054 000
Общая сумма расходов по смете, Br	$С_р$	55 999 926
Расходы на сопровождение и адаптацию, Br	$Р_{са}$	16 799 978
Полная себестоимость, Br	$С_п$	72 799 904
Прогнозируемая прибыль, Br	$П_с$	25 479 966
НДС, Br	НДС	19 655 974
Прогнозируемая отпускная цена ПО, Br	$Ц_о$	117 935 844
Чистая прибыль, Br	$П_ч$	20 893 572

ЗАКЛЮЧЕНИЕ

В данном дипломном проекте был рассмотрен вопрос автоматического построения структуры вероятностной сети на основе экспериментальных данных. В рамках дипломного проекта была разработана библиотека кода для представления и автоматического построения структуры сети. В разработанной библиотеке использовались два различных подхода к оценке качества сети, на основе принципа МДО и оценке апостериорной вероятности структуры для имеющихся экспериментальных данных. Также для разных оценок использовались разные стратегии поиска оптимальной структуры сети в пространстве возможных решений.

В целом были получены удовлетворительные результаты на хорошо изученных и известных сетях Asia и ALARM. Результаты работы реализованных в библиотеке функций поиска в большинстве случаях превосходят по качеству функциональность уже существующего программного обеспечения. Также был предложен способ улучшения качества обучаемой сети на малом объеме данных, основанный на предварительной рандомизации экспериментальных данных. Данный способ удовлетворительно зарекомендовал себя в проведённых тестах. Помимо предложенной модификации были произведены небольшие улучшения в хорошо известных алгоритмах, направленные на повышение скорости их работы. Для повышения производительности применялась мемоизация и использовались прологарифмированные версии некоторых оценок.

В результате цель дипломного проекта была достигнута. Было создано программное обеспечение. Но за рамками рассматриваемой темы осталось еще много других алгоритмов вывода структуры и интересных вопросов, связанных, например, со статистическим выводом суждений в вероятностных сетях, нахождением параметров распределения и других вопросов, возникающих при работе с вероятностными сетями. Эти задачи также являются нетривиальными и требуют детального изучения и проработки — задача статистического вывода, например, является $\mathcal{NP}$-трудной [12] — и не рассматриваются в данном дипломном проекте из-за временных ограничений на его создание. В дальнейшем планируется развивать и довести существующее ПО до полноценной библиотеки, способной решать более широкий класс задач, возникающих в области применения вероятностных сетей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Research, Microsoft. Draft F# Component Design Guidelines. — 2010. — April.

[2] Lauritzen, S. L. Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems / S. L. Lauritzen, D. J. Spiegelhalter // Journal of the Royal Statistical Society. Series B (Methodological). — 1988. — Vol. 50, no. 2. http://dx.doi.org/10.2307/2345762.

[3] The ALARM Monitoring System: A Case Study with Two Probabilistic Inference Techniques for Belief Networks / I. A. Beinlich, H. J. Suermondt, R. M. Chavez, G. F. Cooper // Second European Conference on Artificial Intelligence in Medicine / Ed. by J. Hunter, J. Cookson, J. Wyatt. — Vol. 38. — Berlin, Germany: Springer-Verlag, 1989. — Pp. 247–256.

[4] Cooper, Gregory F. A Bayesian method for constructing Bayesian belief networks from databases / Gregory F. Cooper, Edward Herskovits // Proceedings of the Seventh conference on Uncertainty in Artificial Intelligence. — UAI'91. — San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1991. — Pp. 86–94. http://dl.acm.org/citation.cfm?id=2100662.2100674.

[5] Терентьев, А. Н. Эвристический метод построения байесовых сетей / А. Н. Терентьев, П. И. Бидюк // Математические машины и системы. — 2006. — № 3.

[6] Chow, C. I. Approximating discrete probability distributions with dependence trees / C. I. Chow, Sexior Member, C. N. Liu // IEEE Transactions on Information Theory. — 1968. — Vol. 14. — Pp. 462–467.

[7] Lam, Wai. Learning Bayesian belief networks: An approach based on the MDL principle / Wai Lam, Fahiem Bacchus // Computational Intelligence. — 1994. — Vol. 10. — Pp. 269–293.

[8] Михнюк, Т. Ф. Охрана труда : учебник [утв. МО РБ] / Т. Ф. Михнюк. — Минск : ИВЦ Минфина, 2009. — 345 с.

[9] Синилов, В. Г. Системы охранной, пожарной и охранно-пожарной сигнализации : учебник для нач. проф. образования / В. Г. Синилов. — 5-е изд. — М. : Издательский центр «Академия», 2010. — 512 с.

[10] Палицын, В. А. Технико-экономическое обоснование дипломных проектов: Метод. пособие для студ. всех спец. БГУИР. В 4-х ч. Ч. 4: Проекты программного обеспечения / В. А. Палицын. — Минск : БГУИР, 2006. — 76 с.

[11] Календарь праздников на 2013 год для Беларуси [Электронный ресурс]. — Электронные данные. — Режим доступа: http://calendar.by/

2013/#bkm. — Дата доступа: 05.03.2013.

[12] Koller, D. Probabilistic Graphical Models: Principles and Techniques / D. Koller, N. Friedman. — MIT Press, 2009. — 1270 P.