Национальный исследовательский университет
« Высшая школа экономики »

Лицей

Индивидуальная выпускная работа

Редактор иерархических диаграмм

Выполнила Осипова Елизавета Павловна

Москва 2024

Описание проекта

Редактор иерархических диаграмм – программный комплекс, состоящий из серверной и клиентской части, предназначенный для создания и редактирования иерархических диаграмм.

Основная задача клиентской части продукта – помощь пользователю в визуализации иерархической структуры данных и выделении в ней цветом наиболее значимых или проблемных мест, например:

- файлов, которые имеют большой размер и при этом давно не использовались, чтобы подсветить их в качестве первоочередных целей для удаления и освобождения места на диске;
- размеров отделов, одновременно цветом давая понять руководителю, какие из них эффективно справляются с поставленными перед ними задачами;
- объема учебного материала в тех или иных темах, подсвечивая те из них, где учащиеся имеют низкий средний балл по проверочным работам.

Построенная с помощью программы диаграмма может быть экспортирована в виде изображения и вставлена в слайды презентации для демонстрации широкой аудитории.

Основная задача серверной части продукта – версионирование диаграммы. Пользователь по мере редактирования может сохранять её текущее состояние в облачное хранилище. Благодаря этому он может впоследствии вернуться к любой её версии.

Я решила разработать этот проект, чтобы помочь своей маме, которая является руководителем и использует такие диаграммы для анализа работы своих подразделений. Она регулярно проводит анализ достижения целей в своих

отделах и нуждается в инструменте, который бы позволял ей наглядно демонстрировать результаты в виде диаграмм. Целью было создать удобный инструмент, который позволит визуализировать данные за минимальное время и с учетом их иерархической структуры.

У меня не было опыта выполнения подобных проектов, и я училась создавать этот проект “с нуля”. На начальном этапе мне пришлось освоить новый для меня язык программирования Swift, изучить фреймворки SwiftUI для разработки графического интерфейса для macOS, а также Vapor и Docker для серверной части. Основная сложность заключалась в разработке алгоритма упаковки окружностей для визуализации, так как мне пришлось самостоятельно разрабатывать его без готовых решений, что потребовало изучения литературы по математике.

Проблемное поле

Существует большое количество предметных областей, в основе которых лежат иерархические структуры данных, например:

- файловая система на наших компьютерах состоит из директорий, содержащих другие директории и файлы;
- организационная структура компаний представляет собой иерархию отделов, в которых числятся сотрудники;
- учебный курс разбивается на части, которые содержат темы, которые содержат уроки, которые содержат проверочные задания.

Традиционные способы отображения иерархий, такие как простые списки, могут быть неудобными для восприятия, особенно если количество элементов велико. Проблема заключается в том, что для эффективной визуализации иерархий требуется не только корректно разместить элементы (например, окружности, представляющие узлы иерархического дерева), но и сделать это так, чтобы получившаяся диаграмма была наглядной и легко читаемой.

Полученные с помощью моей программы изображения могут быть вставлены в слайды презентации для демонстрации широкой аудитории выявленных характеристик входных данных.

Кроме того, для таких диаграмм важна высокая производительность, особенно если структура данных сильно разветвленная или включает большое количество элементов. Решение этой проблемы часто требует применения специальных алгоритмов упаковки объектов (например, окружностей), что является нетривиальной задачей в области вычислительной геометрии.

Основная сложность задачи заключается в реализации алгоритма упаковки окружностей. Анализ академических статей показал отсутствие описания решений, которые бы позволяли решить эту задачу за приемлемое для пользователя время. Приведенные в них методы нацелены на решение задач промышленности, где минимизация площади результирующей внешней окружности важнее времени, затраченного на расчёт местоположения составляющих её элементов, так как позволяет экономить ресурсы физического мира. В пользовательских приложениях у нас ситуация ровно противоположная – нам важно визуализировать данные за минимальное время в ущерб в том числе её пространственным характеристикам.

Моя мама, как руководитель, регулярно сталкивается с необходимостью анализировать эффективность работы своих подразделений. Она ставит перед отделами цели, и по завершении отчетных периодов ей нужно представить результаты работы в понятной и наглядной форме. Однако стандартные методы отчетности (например, таблицы или текстовые документы) часто оказываются неудобными и трудными для восприятия, особенно если нужно показать соотношение данных в иерархической структуре (например, количество сотрудников и процент выполнения целей в отделах).

Для этого ей требовался инструмент, который бы помог эффективно визуализировать эти данные в виде иерархической диаграммы. Это подтверждает, что проблема необходимости удобной и быстрой визуализации иерархических данных реально существует, и заказчик (мама) нуждается в решении, которое позволит ей наглядно представлять данные и выявлять проблемные области (например, отделы с низкой результативностью или высокой текучестью кадров).

Заказчик явно показал, что нуждается в инструменте, который позволил бы быстро и понятно отобразить эти данные, выявить сильные и слабые стороны, а также представить информацию в виде визуальных отчетов для демонстрации коллегам или руководству.

Целевая аудитория

Целевая аудитория моего продукта включает специалистов из различных сфер, для которых важна наглядная визуализация данных иерархического типа, например:

Менеджеры и руководители

Потребности: Менеджеры нуждаются в инструменте, который помогает оценить эффективность работы сотрудников, распределение задач и текучесть кадров. Для руководителей важно иметь возможность выделять проблемные зоны в отчетах и иллюстрировать достижения отделов.

Цели использования: Быстрое создание диаграмм для отчетов, где можно наглядно показать статус выполнения целей, анализировать динамику и выявлять отделы, которые требуют особого внимания.

Специалисты в сфере аналитики

Потребности: Аналитики, работающие с большими объемами данных, часто сталкиваются с необходимостью визуализировать иерархические структуры. Им важно эффективно демонстрировать сложные связи в данных, а также выделять аномалии в визуальной форме.

Цели использования: Создание диаграмм для анализа иерархий и взаимосвязей между объектами, визуализация данных, имеющих сложную структуру, для более точной и быстрой аналитики.

Сфера образования (преподаватели, методисты)

Потребности: Специалисты в образовании заинтересованы в наглядной демонстрации структуры учебных программ. Это может быть полезно для анализа, планирования и отображения модулей курсов, тем, подтем и заданий.

Цели использования: Систематизация и визуализация образовательных программ, представление структуры курсов и сложности их прохождения для более полного понимания учебного процесса.

Исследователи и научные сотрудники

Потребности: В научной работе также часто используется структурирование данных, будь то иерархия научных статей по теме, структура экспериментов или систематизация результатов исследований.

Цели использования: Визуализация иерархии исследований, упорядочение больших объемов информации для быстрого анализа и создания научных презентаций, которые можно представить коллегам или на конференциях.

Заказчик

Заказчиком приложения является моя мама, которая, будучи руководителем, отслеживает эффективность работы вверенных ей организационных подразделений. По итогам отчетных периодов она проводит аудит достижения

целей, которая она ставит перед своими отделами, и демонстрирует полученные результаты широкой аудитории в формате презентаций.

Техническое задание

Приложение должно:

- отображать иерархию данных, например, по каждому отделу в организационной структуре компании указывать численность сотрудников и процент выполнения целей;
- давать возможность наглядного выделения проблемных зон (например, отделы с низким уровнем выполнения целей или с высокой текучестью);
- размещать и упаковывать элементы диаграммы компактно и при этом за минимальное время;
- давать возможность добавлять или удалять элементы, корректировать данные;
- иметь простой и интуитивно понятный интерфейс;
- экспортировать диаграммы в формате PNG для включения в презентации;
- обеспечивать доступ к диаграммам с любого устройства через веб-сервис;
- давать возможность сохранения версий диаграммы, чтобы отслеживать изменения по периодам.

Функциональные требования

Основные функциональные требования:

1. Приветственное окно

Требование: Приложение должно предлагать пользователю выбрать одно из трёх действий: создать новую диаграмму, загрузить существующую с диска или выбрать из недавно открытых файлов.

Проблема, которую решает: Упрощение работы с ранее созданными проектами или быстрый старт для новых пользователей, что позволяет экономить время при создании и редактировании диаграмм.

2. Создание и редактирование иерархии элементов

Требование: Возможность добавлять, переименовывать и удалять элементы в структуре диаграммы с помощью удобного контекстного меню.

Проблема, которую решает: Необходимость наглядного отображения иерархической структуры организации. Дерево элементов даёт чёткое представление о связях между элементами, что важно для анализа структурных данных

3. Редактирование свойств элементов

Требование: Возможность редактировать параметры элементов диаграммы, такие как название, цвет и размер, с учетом их типа.

Проблема, которую решает: Необходимость визуально выделять различия между отделами и уровнями иерархии для улучшенного восприятия информации, например, используя цвет для отображения статуса выполнения задач.

4. Экспорт диаграммы в изображение

Требование: Возможность экспорта диаграммы в графический файл

Проблема, которую решает: Потребность представлять диаграмму в презентабельном виде для отчетов и презентаций.

5. Сохранение диаграмм в собственном формате

Требование: Возможность сохранять диаграмму в формате .de и повторно открывать её из списка недавних файлов.

Проблема, которую решает: Возможность продолжить редактирование ранее созданных диаграмм и обмениваться ими с другими пользователями.

6. Регистрация, авторизация в облачном хранилище

Требование: Возможность создавать учётную запись, входить в неё.

Проблема, которую решает: Ограничение доступа к данным только по признаку авторства.

7. Хранение версий диаграммы в облачном хранилище

Требование: Возможность сохранять и загружать версии диаграмм.

Проблема, которую решает: Возможность откатиться к более ранним версиям диаграммы.

Соответствие требованиям проблемному полю:

Эти требования создают инструмент, который делает данные, имеющие сложную структуру, понятными и наглядными.

Визуализация позволяет руководителю быстро идентифицировать проблемы в структуре, например, отделы с низким выполнением задач.

Интуитивно понятное меню для добавления и редактирования элементов делают процесс настройки иерархии простым, что подходит пользователям с минимальными навыками работы с данными.

Возможность работать с файлами на диске и в хранилище помогает сохранять актуальные версии и легко восстанавливать старые данные для анализа динамики.

Аналогичные/похожие продукты

Существует популярный фреймворк D3.js, написанный на JavaScript, который позволяет встраивать подобного рода диаграммы на страницы веб-сайтов. Посмотреть на внешний вид такой диаграммы можно на официальной странице. В виду того, что это лишь фреймворк, пользоваться им могут лишь IT-специалисты, а для специалистов-предметников порог вхождения в него по сравнению с разработанной программой представляется крайне высоким.

Стек технологий

- язык программирования Swift
- фреймворки AppKit и SwiftUI для реализации приложения для macOS
- фреймворк Vapor для реализации API и логики бэкенда Web-сервиса
- SQL и фреймворк Fluent для работы с базой данных Web-сервиса
- Docker для развёртывания Web-сервиса

Рефлексия

1. Возникающие проблемы:

Главной сложностью реализации проекта являлась необходимость изучения большого объёма информации в сжатые сроки.

Поскольку я достаточно хорошо изучила язык программирования Swift, для написания серверной части проекта я решила использовать его же. Самым популярным фреймворком в этой области оказался Vapor. В его официальной документации мне оказалось достаточно много подходящего для моих нужд

кода, который можно было применить с минимальными изменениями, благодаря чему удалось уложиться в заданные сроки.

В числе прочего там содержался код реализации сценариев логина и регистрации. Реализация прочих методов API работы с диаграммой была в высокой степени сходна с имевшимся в документации кодом для работы с сущностями ToDo-листа.

Сложность реализации клиентской части проекта. Как и в серверной части сложность заключалась в необходимости освоения большого количества материала по новым для меня фреймворкам AppKit и SwiftUI, связи компонентов, написанных на них между собой.

2. Как возможно развивать проект:

Проект может быть расширен и улучшен в будущем следующими способами:

- добавление инструмента для автоматического анализа на основе данных, встроенных в диаграмму;
- добавление возможности работы над диаграммой сразу несколькими пользователями.

В дальнейшем те части проекта, которые касаются алгоритма упаковки и визуализации диаграммы можно выделить в отдельные пакеты для распространения в open source. Полученную программу можно использовать как демонстрационное приложение, которое показывает возможности библиотек.

3. Чему я научилась:

- Предвидеть различные проблемы с совместимостью компонентов программы. Этот навык поможет мне в будущем предотвращать возможные проблемы совместимости компонентов на ранних этапах

разработки и избежать множества ошибок и недоработок. Это позволит мне разрабатывать проекты, где использование различных фреймворков и библиотек не приведет к неожиданным сбоям

- Разрабатывать интуитивно понятный интерфейс, удобный для пользователей. Работа с цветовой палитрой и настройкой визуализации помог мне лучше понять, как малейшие изменения интерфейса могут влиять на восприятие информации

4. Что необходимо было бы сделать по-другому:

Одним из решений для уменьшения сложности кода могло бы стать полное использование компонентов фреймворка SwiftUI без обращения к AppKit при разработке приложения для macOS