ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ И НАУКИ ГОРОДА МОСКВЫ
ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ПРОФЕССИОНАЛЬНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ Г. МОСКВЫ
«КОЛЛЕДЖ ПРЕДПРИНИМАТЕЛЬСТВА №11»
ЦЕНТР ИНФОРМАЦИОННО–КОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Допустить к защите

Заместитель директора по содержанию и ИКТ

ГАПОУ КП № 11

_____ Адилова К.О.

« » мая 2021 г.

ДИПЛОМНАЯ РАБОТА

Разработка автоматизированной системы управления для построения бизнес-процесса на основании данных для компании ООО Эррайвал Рус

по специальности: **09.02.07 Информационные системы и программирование**

Выполнил:
студент группы ИСиП-41
Пан Филипп Юльевич

Научный руководитель:
преподаватель Центра ИКТ
Кузнецов Кирилл Александрович

_____ подпись

_____ подпись

Москва, 2021г

Аннотация

Дипломная работа Пана Филиппа Юльевича на тему: Разработка автоматизированной системы управления для построения бизнес-процесса на основании данных для компании ООО «Эррайвал Рус».

Научный руководитель преподаватель Центра ИКТ Колледжа Предпринимательства №11 Кузнецов Кирилл Александрович.

Работа включает в себя XX страниц.

Объектом дипломной работы является компания, производящая электромобили и автономное производство ООО «Эррайвал Рус».

Предметом работы является процесс разработки системы для сбора данных о процессах компании ООО «Эррайвал Рус».

Целью дипломной работы является разработка системы для автоматической подготовки данных для построения бизнес-процесса в системе процесс майнинга для компании ООО «Эррайвал Рус».

Задачи работы:

- Изучение организации ООО «Эррайвал Рус»;
- Анализ области деятельности компании, с которой будет производиться работа;
- Описать требования к системе;
- Выбрать ПО для реализации;
- Подготовка данных к анализу;
- Построение схемы бизнес-процесса
- Выбор метрик и ключевых показателей эффективности
- Определить узкие места и выявить инсайты по итогам анализа
- Описать технико-экономической составляющую
- Описать требования по технике безопасности и охране труда

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Тема дипломной работы: Разработка автоматизированной системы управления для построения бизнес-процесса на основании данных для компании ООО Эррайвал Рус.

В настоящее время рынок автомобильной промышленности заполнен техникой, использующей бензиновое и дизельное топливо. Их число с каждым годом растет, а производственные мощности усиливаются. Добыча и обработка топлива, выхлопные газы – все это наносит определенный вред экологии. На предприятиях работают десятки тысяч человек, это оплата труда каждого из них влияет и на итоговую стоимость продукции. Но в машиностроении до сих пор преобладают традиционные способы производства.

Компания ООО «Эррайвал Рус» собирается сделать прорыв в этой области. Они производят электромобили и стремятся к полной автоматизации производства. Идея в том, чтобы производить экологичный продукт и при этом минимизировать человеческий фактор на заводах. Отдать все в «железные руки» роботов. Это позволит быстро и максимально дешево производить экологичные автомобили на электродвигателях из очень прочных и легких композитных материалов и брать на работу на заводы только программистов, специалистов по роботам и дата-стюардов.

Сейчас компания активно занимается разработкой и созданием этой технологии автоматизации большинства процессов для перехода со стадии R&D (research and development – англ. исследование и разработка) на массовое производство. Одно из препятствий – отсутствие единого понимания и схемы эталонных бизнес-процессов. Необходимо построить карту всех процессов для того, чтобы по этим алгоритмам работало производство. И очень важным этапом является перевод всех данных о процессах всех отделов в понятный вид, который помог бы построить автономную работу этих отделов.

На данный момент проблемой является отсутствие системы для автоматической обработки и сбора сырых данных для подготовки их к анализу.

Поэтому целью дипломной работы будет разработать систему, которая поможет подготовить данные, чтобы потом на их основе построить схему бизнес-процессов, получить инсайты и найти проблемные места в построенной карте процессов.

Для достижения этой цели были поставлены следующие задачи:

- Изучить организации ООО «Эррайвал Рус»;
- Проанализировать область деятельности компании, с которой будет производиться работа;
- Описать требования к системе;
- Выбрать ПО для реализации;
- Разработать проект системы;
- Разработать программу для подготовки данных;
- Построить схемы бизнес-процесса;
- Выбрать метрики и ключевые показателей эффективности;
- Определить узкие места и выявить инсайты по итогам анализа;
- Описать технико-экономической составляющую;
- Описать требования по технике безопасности и охране труда;

Объектом дипломной работы является компания ООО «Эррайвал Рус».

Предметом работы является процесс разработки системы для сбора данных о процессах компании ООО «Эррайвал Рус».

ГЛАВА 1. ПРОЕКТИРОВАНИЕ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ УПРАВЛЕНИЯ ДЛЯ ПОСТРОЕНИЯ БИЗНЕС-ПРОЦЕССА

1.1 Анализ организации ООО Эррайвал Рус

Данная компания занимается разработкой и производством электромобилей на всех стадиях: от изучения и производства композитных материалов до продаж готового продукта конечному пользователю. Видением компании является производство экологичного электротранспорта, который будет дешевле и доступнее автомобилей, использующих горючие топлива. Для того, чтобы сделать свой продукт максимально дешевым для покупателей, компания стремится максимально удешевить производство, сохраняя при этом достойное качество. Достичь этого ООО Эррайвал Рус планирует путем создания уникального полностью автономного и роботизированного производства с минимальным привлечением сотрудников. Это является основной задачей для достижения конечной цели и позволит потом продавать не только электротранспорт как продукт, но и способ, технологию автономного производства.

Ключевая разработка компании — платформа для модульной сборки, в которую входят аккумуляторная батарея, электродвигатель и компоненты трансмиссии. Эррайвал может производить как легковые автомобили, так и грузовые фургоны. Производственные мощности и научно-исследовательские центры компании расположены в США, Германии, Израиле, России и Великобритании.

Компания состоит из множества различных департаментов.

- Отдел инженеров-проектировщиков
- Отдел создания прототипов изделий
- Производственный отдел

- Отдел IT
- Отдел продаж
- Финансовый отдел

Каждый отдел производит большой объем данных в разных формах и хранит в разных системах. На основе данных каждого из них будут строиться бизнес-процессы.

Сейчас Эррайвал Рус находится на этапе перехода со стадии R&D на массовое производство. Это означает автоматизацию и роботизацию всех производственных процессов. В компании все бизнес-процессы разрознены, все производимые данные мало связаны друг с другом, не структурированы и не имеют единой системы хранения, а системы в свою очередь проинтегрированы напрямую. А для того, чтобы сделать роботизированное предприятие, нужно точно от и до знать весь процесс производства на этапе R&D, иметь все данные о нем и составить модель процессов. Но ни один бизнес-аналитик не может изобразить бизнес-процесс таким, какой он будет в реальной жизни.

По итогам 2018 года компания понесла убыток €61,5 млн (€32 млн годом ранее). В 2020 должна быть представлена первая реальная модель автомобиля, а в 2021 году планируется полноценный запуск производства.

Разработанная система для построения бизнес-процесса позволит собрать все данные о процессах, происходящих во всех отделах предприятия, навести в них порядок и построить на их основе правильные карты процессов.

В дипломной работе будут рассматриваться данные о закупках материалов, но такой подход при определенных корректировках может использоваться и для всех остальных отделов и сфер.

1.2 Описание требований к системе

Система должна быть реализована в виде консольного приложения под компьютер с операционной системой Windows.

Возможности программы:

- Создание таблиц в базе данных по файлам со скриптами sql;
- Создание таблицы cases с добавлением существующих исходных данных
- Поиск необходимых данных в нескольких таблицах по уникальным ключам
- Поиск и объединение данных по определенным условиям
- Создание таблицы activities и заполнение ее актуальными значениями
- Отображение текущего статуса работы в консоли

Дальнейшая работа должна быть реализована с использованием технологии Process Mining.

Возможности системы Process mining:

- Построение визуальной карты процессов
- Отображение статистических данных по определенным метрикам
- Определение проблемных мест и процессов, которые стоит исправить

1.3 Выбор ПО для реализации

Программный код системы будет написан в среде разработки Visual Studio 2019 на языке программирования C# версии 7.3. Данные будут храниться в PostgreSQL, работа с данными производится с помощью среды JetBrains DataGrid. Система процесс майнинга будет от компании Celonis.

- **Celonis: система Process Mining**

Для выявления фактического бизнес-процесса, как и для его последующего совершенствования, часто используется графическое или текстовое описание на базе информации, полученной в ходе интервью с участниками процесса. Несмотря на всю простоту данного подхода,

ключевым его недостатком является сложность извлечения знаний из участников процесса, что часто вызвано их нежеланием рассказывать об особенностях процесса, и главное, о недостатках в своей работе. Process Mining позволяет делать это автоматически.

Подобный человеческий фактор способствует тому, что часть нужных данных просто теряется, а созданная таким образом модель скоро теряет свою актуальность, что вызывает необходимость повторного анализа.

Однако нужно отметить, что в век активного применения информационных технологий, все больше и больше записей о событиях реального мира накапливается в информационных системах, обеспечивая тем самым детализированную информацию об истории исполнения бизнес-процессов. Для сбора и анализа данной информации в целях дальнейшего совершенствования бизнес-процесса и появилась технология Process mining (Идеолог Process Mining — Вил ван дер Аалст — профессор Эйндховенского технического университета и Квинслендского технического университета), которая позволяет выявлять и анализировать фактические бизнес-процессы за счет извлечения знаний из журналов событий, доступных в современных информационных системах.

Методы Process mining предполагают, что можно последовательно записывать события, так чтобы каждое из них в дальнейшем поставить в соответствии с четко определенным шагом в определенном бизнес-процессе. Помимо событий из журналов необходимо по возможности извлекать дополнительную информацию о том, кто какое действие выполнил, и в какое время, а также связанные с этим действием данные, например название клиента или объем платежа.

Таким образом, технология Process mining включает в себя автоматизированное построение моделей фактически исполняемых бизнес-процессов на основании анализа журнала событий. При этом восстановленные

модели процессов в совокупности данными по времени исполнения процесса и элементами организационной структуры позволяют видеть все скрытые недостатки, обеспечивая владельцев бизнес-процесса и аналитиков огромным количеством материалов для дальнейшего совершенствования.

Celonis — компания разработчик одноименной системы класса process mining. Это решение помогает воссоздать цифровые копии реальных бизнес-процессов, опираясь на данные из информационных систем. ы называют Celonis лидером рынка по итогам 2018 года. Компания уже внедрила свое решение более чем в 30 странах. В её активе более 120 партнеров (консалтинг и внедрение), более 600 клиентов и более 2000 проектов. Самый крупный из них – 6000 пользователей системы, самый большой объем данных – 15 млрд записей.

Celonis делает акцент на цифровой трансформации и вариантах оптимизации операционных ресурсов. Исследование показывает, что компания фокусируется на предсказательной аналитике и анализе взаимосвязи различных процессов, их влиянии на результативность.

Celonis предлагает несколько версий своего решения: локальную, гибридную или облачную (например, Amazon Web Services [AWS] или Microsoft Azure).

- **Visual Studio 2019**

Интегрированная среда разработки Visual Studio — это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Интегрированная среда разработки (IDE) представляет собой многофункциональную программу, которую можно использовать для различных аспектов разработки программного обеспечения. Помимо стандартного редактора и отладчика, которые существуют в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции

для упрощения процесса разработки. Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

Интегрированная среда разработки, ИСР (англ. Integrated development environment — IDE), также единая среда разработки, ЕСР — комплекс программных средств, используемый программистами для разработки программного обеспечения (ПО).

Среда разработки включает в себя:

- текстовый редактор,
- компилятор и/или интерпретатор,
- средства автоматизации сборки,
- отладчик.

- **Язык программирования C#**

Причины использования именно этого языка:

C# является объектно-ориентированным языком, но поддерживает также и компонентно-ориентированное программирование. Разработка

современных приложений все больше тяготеет к созданию программных компонентов в форме автономных и самоописательных пакетов, реализующих отдельные функциональные возможности. Важная особенность таких компонентов — это модель программирования на основе свойств, методов и событий. Каждый компонент имеет атрибуты, предоставляющие декларативные сведения о компоненте, а также встроенные элементы документации. C# предоставляет языковые конструкции, непосредственно поддерживающие такую концепцию работы. Благодаря этому C# отлично подходит для создания и применения программных компонентов.

- Приложение будет разработано под **операционную систему Windows10.**

Windows 10 — операционная система для персональных компьютеров и рабочих станций, разработанная корпорацией Microsoft в рамках семейства Windows NT.

- **PostgreSQL**

PostgreSQL — объектно-реляционная система управления базами данных

PostgreSQL — это популярная свободная объектно-реляционная система управления базами данных. PostgreSQL базируется на языке SQL и поддерживает многочисленные возможности.

Преимущества PostgreSQL:

- поддержка БД неограниченного размера;
- мощные и надёжные механизмы транзакций и репликации;
- расширяемая система встроенных языков программирования и поддержка загрузки C-совместимых модулей;
- наследование;
- легкая расширяемость.

Текущие ограничения PostgreSQL:

- Нет ограничений на максимальный размер базы данных
- Нет ограничений на количество записей в таблице
- Нет ограничений на количество индексов в таблице
- Максимальный размер таблицы — 32 Тбайт
- Максимальный размер записи — 1,6 Тбайт
- Максимальный размер поля — 1 Гбайт
- Максимум полей в записи250—1600

Особенности PostgreSQL:

Функции в PostgreSQL являются блоками кода, исполняемыми на сервере, а не на клиенте БД. Хотя они могут писаться на чистом SQL, реализация дополнительной логики, например, условных переходов и циклов, выходит за рамки, собственно, SQL и требует использования некоторых языковых расширений. Функции могут писаться с использованием различных языков программирования. PostgreSQL допускает использование функций, возвращающих набор записей, который далее можно использовать так же, как и результат выполнения обычного запроса. Функции могут выполняться как с правами их создателя, так и с правами текущего пользователя. Иногда функции отождествляются с хранимыми процедурами, однако между этими понятиями есть различие.

- **DataGrid**

Программное обеспечение JetBrains DataGrip представляет собой IDE-инструмент для работы с базами данных MySQL, PostgreSQL, Oracle, SQL Server, Sybase, DB2, SQLite, HyperSQL, Apache Derby и H2. Решение JetBrains DataGrip включает мощный текстовый редактор с мультикурсорами, обеспечивает синтаксическое выделение кода, поддерживает интеграцию с системами контроля версий Git, Subversion и т. д.

DataGrip предоставляет инструменты для работы с объектами базы данных. Если пользователь создает или изменяет таблицу, добавляет или

изменяет колонку, ему пригодится графический интерфейс JetBrains DataGrip. Подобные изменения сопровождаются генерацией соответствующего скрипта: можно сразу выполнить сделанные изменения в базе или скопировать сгенерированный DDL-запрос в редактор и работать уже непосредственно с кодом.

1.4 Порядок работ

1. Спроектировать систему, составить схему потоков данных
2. Провести подготовительную разметку имеющихся данных в разных таблицах
3. Создать консольное приложение и написать код, по которому данные будут правильно сортироваться в нужных таблицах
4. Консолидировать данные в двух таблицах – activities и cases, разметить их по столбцам – case_id, timestamp, event. Объединить таблицы в одну таблицу processes и экспортировать как CSV-файл
5. Загрузить итоговую таблицу в систему процесс майнинга
6. Построить карты процессов
7. Определить метрики и KPI
8. Получить необходимые инсайты

1.5 Проектирование системы

Для процесса закупок материалов, в который входит оформление заявки, прослеживание статуса заявки, получение счета за доставленный товар и оплата счета, используется множество разных систем хранения данных.

Заявки на запчасти и материалы хранятся в системе Airtable и Dear, весь процесс изменения статуса заявки фиксируется в нескольких таблицах в системе Jira, все счета за заявки собираются в таблицах Xero. В ходе выполнения дипломной работы, чтобы построить схему бизнес-процесса закупок, например, необходимо будет во всех этих таблицах найти столбцы с данными, которые нужны для процесс-майнинга.

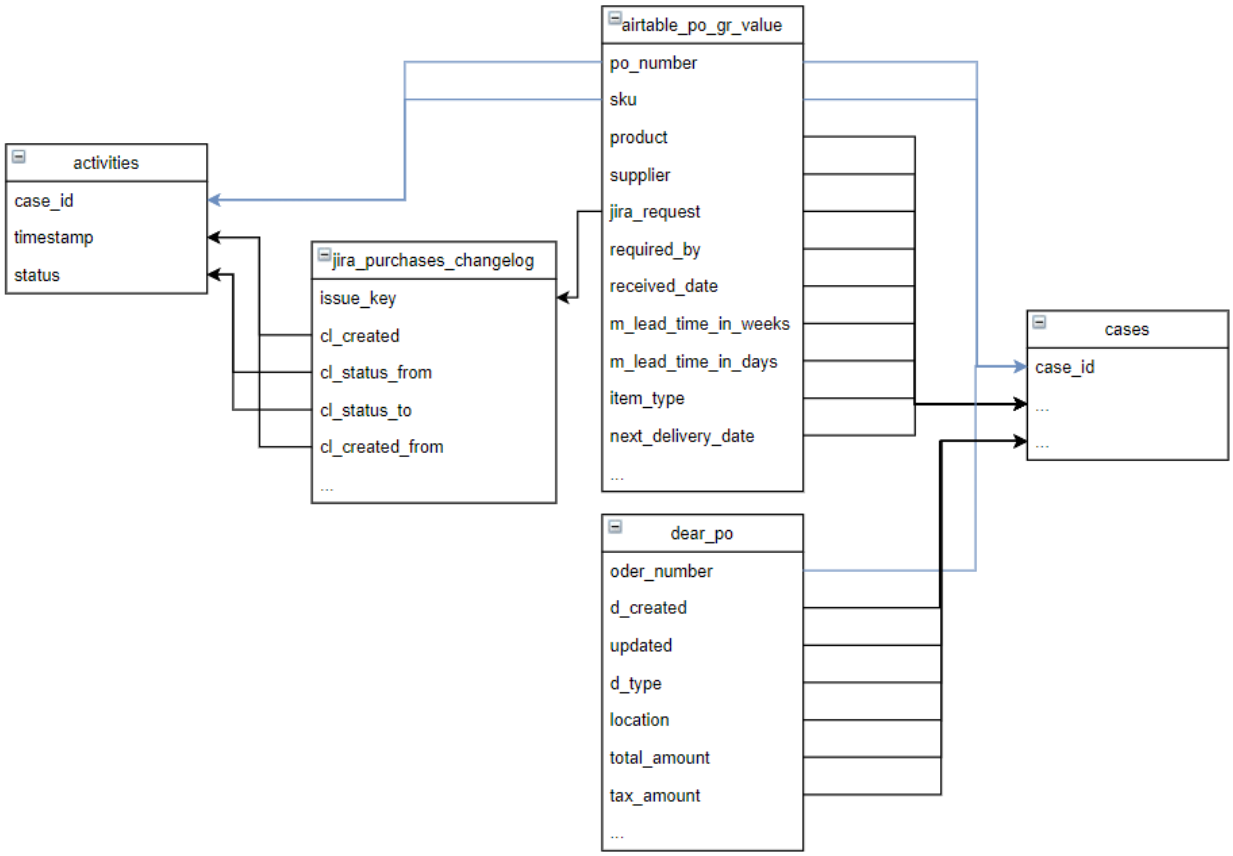

Рисунок 1 – ERD Диаграмма

Для работы будет использоваться три основные таблицы:

- airtable_po_gr_value – здесь хранятся такие данные о заказах, как номера товаров в заказе, названия продуктов, данные о поставщиках, время, когда товар должен был прийти и когда он на самом деле пришел, разницу в этих двух показателях и другие
- dear_po – данная таблица хранит в себе данные о поставщике, местонахождении, количестве товаров в заказе, данные о валюте и стоимости, времени создания и изменения статуса заказа и другие
- jira_purchases_changelog – представляет собой лог изменений статуса заказа и хранит данные о том, в какое время и в какой статус перешел заказ

Система автоматически создает две таблицы. Они продемонстрированы на рисунке 1.

- Cases – таблица, в которой хранятся все основные данные из вышеописанных таблиц по конкретному товару конкретного заказа
- Activities – таблица хранит в себе основные три столбца, которые будут использоваться при построении карты бизнес процессов.
- Case_ID – номер заказа и номер товара в нем
- Timestamp – временная отметка события, когда статус изменился
- Status – статус, в который перешел товар в данное время

ГЛАВА 2. РАЗРАБОТКА СИСТЕМЫ

После изучения данных той сферы работы компании, с которой будет производиться работа, можно приступать к подготовке данных.

2.1 Разработка плана реализации

Изначально имеется озеро данных, которые с помощью разрабатываемой системы будут подготовлены к процедуре процесс-майнинга посредством сбора необходимых данных из разных таблиц в одну и определением столбцов case_id, timestamp, event, actor. Далее полученный CSV файл загружается в систему процесс-майнинга. Эта система должна сформировать визуальную схему процессов. Анализ результатов происходит в системе.

2.2 Подготовка данных

Хранение данных будет осуществляться в системе управления базами данных PostgreSQL. Добавим исходные таблицы в базу данных при помощи SQL скриптов. На рисунке 2 представлен пример скрипта.

```
create table airtable_po_gr_value
(
    conc                  text,
    po_number             text,
    sku                   text,
    product               text,
    supplier              text,
    jira_request          text,
    a_status              text,
    po_status             text,
    po_date               text,
    required_by           text,
    received_date         text,
    m_lead_time_in_weeks  numeric,
    m_lead_time_in_days   numeric,
    item_type             text,
    next_delivery_date    text
);
INSERT INTO public.airtable_po_gr_value (conc, po_number, sku, product, supplier, jira_reque
INSERT INTO public.airtable_po_gr_value (conc, po_number, sku, product, supplier, jira_reque
INSERT INTO public.airtable_po_gr_value (conc, po_number, sku, product, supplier, jira_reque
INSERT INTO public.airtable_po_gr_value (conc, po_number, sku, product, supplier, jira_reque
INSERT INTO public.airtable_po_gr_value (conc, po_number, sku, product, supplier, jira_reque
INSERT INTO public.airtable_po_gr_value (conc, po_number, sku, product, supplier, jira_reque
INSERT INTO public.airtable_po_gr_value (conc, po_number, sku, product, supplier, jira_reque
INSERT INTO public.airtable_po_gr_value (conc, po_number, sku, product, supplier, jira_reque
INSERT INTO public.airtable_po_gr_value (conc, po_number, sku, product, supplier, jira_reque
```

Рисунок 2 – Скрипт добавления таблицы с данными

2.3 Создание консольного приложения

В среде разработки Visual Studio создано консольное приложение (.NET Framework)

Функционал приложения включает в себя возможность создания таблиц по файлам со скриптами SQL, создания таблиц cases и activities, заполнения этих таблицы данными из других таблиц. Программный код представлен в Приложении А.

Первый блок кода отвечает за подключение к базе данных (БД). Данные для подключения передаются через параметры. После этого была разработана функция добавления таблиц в базу при помощи файла с SQL скриптом. Программа прочитывает файл, выделяет в нем отдельные команды и передает их как запрос в БД на создание таблицы и добавления туда данных. Перед этим пользователь может выбрать, добавлять таблицы или нет.

После этого формируется два скрипта для создания таблиц cases и activities с их полями. Первая включает в себя определенные столбцы из других таблиц с их данными, разложенными по каждому товару в каждом заказе. Вторая таблица создается пустой, но далее следуют алгоритмы по ее наполнению.

В таблице activities необходимо хранить данные о текущем состоянии каждого товара в каждый момент времени. В таблице airtable у многих заказов есть специальный ключ для доступа к данным о статусах этих заказов в таблицах системы jira. Сначала программа получает все номера заказов, а потом по каждому ищет ключ и находит соответствующие данные о заказе в другой таблице.

В системе jira фиксировались моменты, когда заказ переходил из одного статуса в другой, там же сохранялись и отметки времени, когда произошел переход и когда был переход в предыдущий статус. Приложение проводит анализ этих данных, находит по датам, какой статус за каким шел и записывает

в таблицу activities по-отдельности в одну строку номер товара, статус и время перехода в него. Таким образом мы получаем таблицу с данными об актуальном на определенный момент времени статусе заказа каждого товара.

После этого готовые две таблицы при помощи скрипта объединяются в одну таблицу processes. Последняя уже будет экспортирована как CSV-файл и загружена в систему Celonis для анализа.

2.4 Использование таблиц в системе процесс-майнинга

CSV файл таблицы processes загружается в Celonis. В системе необходимо определить какие поля таблиц являются определяющими (case_id), в каких находятся данные о времени (timestamp), и где хранятся описания основных событий (event).

2.5 Анализ данных в Celonis

На основании данных загруженной в Celonis таблицы processes будут построены графики и схемы процессов.

1. Вкладка Overview

Рисунок 3 – Вкладка Overview

На рисунке 3 слева отображается визуальная схема процессов, которую можно масштабировать по двум параметрам – количеству разных статусов и количеству путей переходов. Она позволяет увидеть, сколько заказов

двигалось по каждому из путей, и как быстро это происходило. Справа отображается таблица с данными о количестве действий в каждом заказе.

2. Вкладка Throughput time

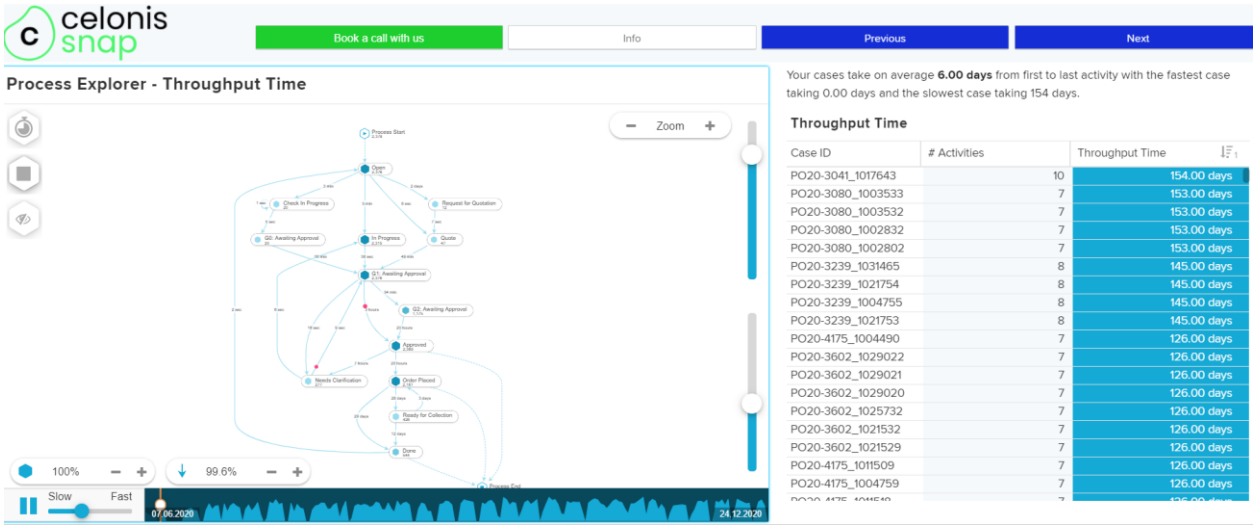

Рисунок 4 – Вкладка Throughput time

Слева на рисунке 4 можно увидеть ту же визуальную карту процессов, но уже с данными о времени на каждой линии, а справа – табличное представление данных о том, сколько дней заняли процессы каждого заказа.

3. Вкладка Throughput time Details

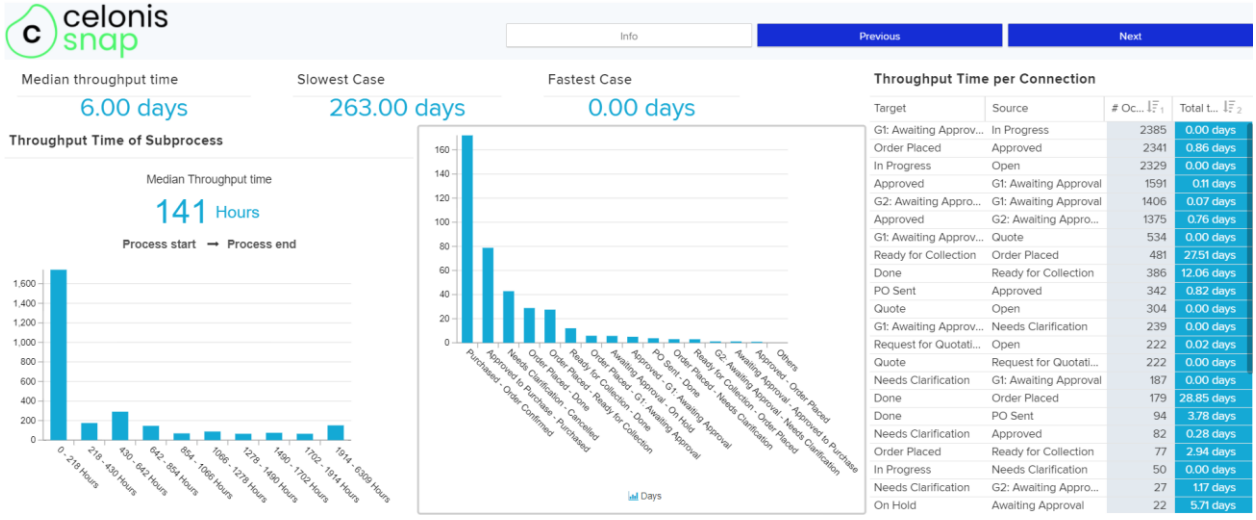

Рисунок 5 – Вкладка Throughput time Details

Окно детализированной информации по времени на рисунке 5 включает в себя (слева направо) диаграмму с данными о том, сколько заказов было

завершено за определенный промежуток времени, диаграмму, в котором показано, сколько дней занимают переходы из одного статуса в другой, и таблица с числовыми значениями тех же данных.

4. Вкладка Rework rate

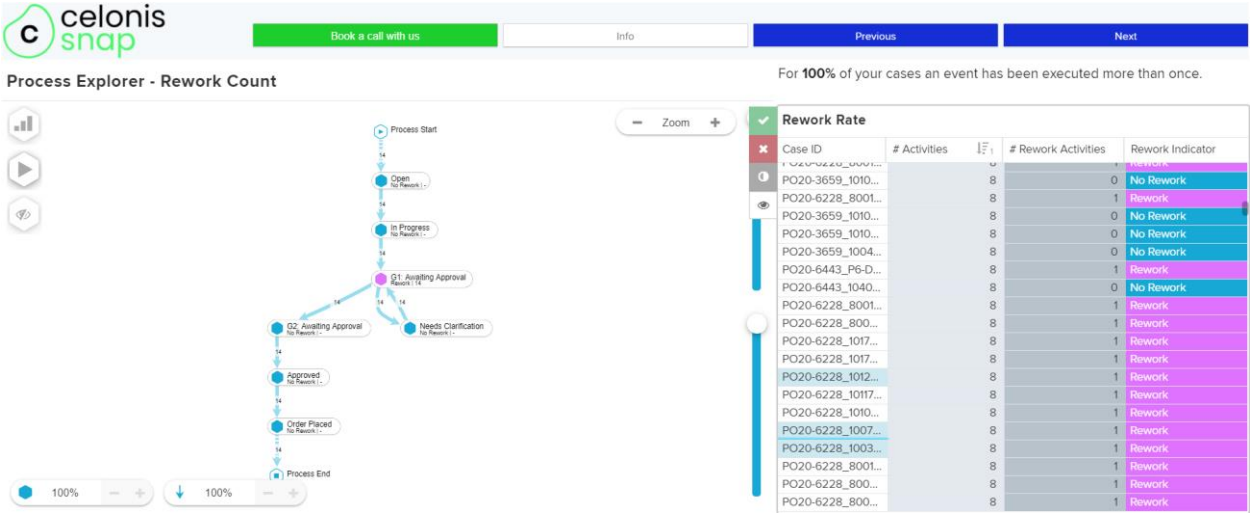

Рисунок 6 – Вкладка Rework rate

На рисунке 6 показаны данные о количестве переработок (возвращений в прежний статус), и на схеме слева можно увидеть, где именно был сделан шаг назад.

5. Вкладка Suppliers

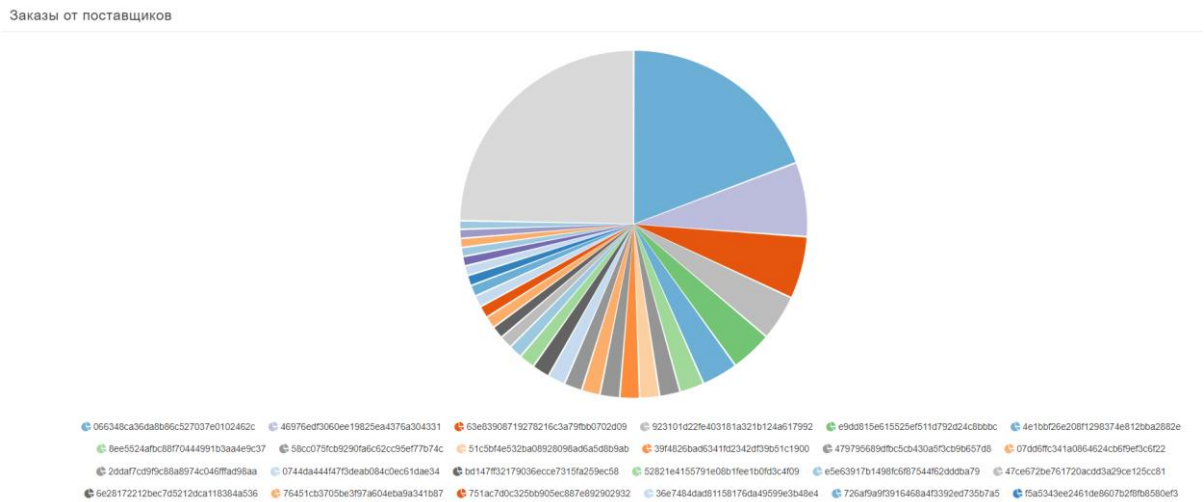

Рисунок 7 – Вкладка Suppliers

Круговая диаграмма на рисунке 7 показывает, какую долю занимают заказы от разных поставщиков.

6. Вкладка Conformance

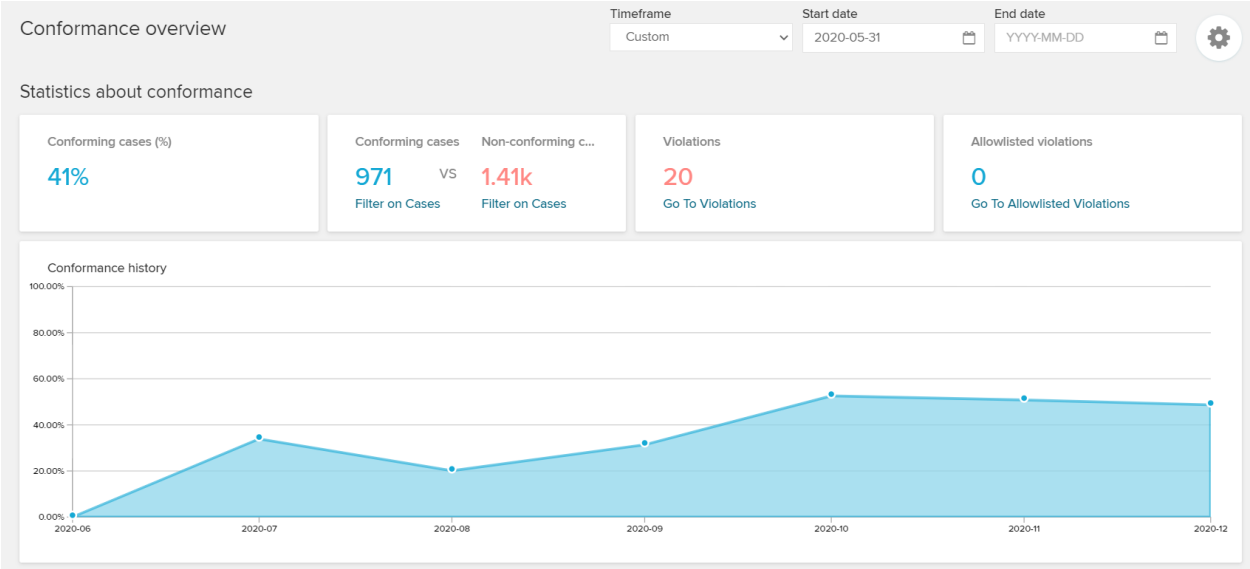

Рисунок 8 – Вкладка Conformance

Графики и данные рисунка 8 показывают, сколько заказов соответствуют эталонному процессу, который был определен прежде и может быть отредактирован.

7. Вкладка Lead time

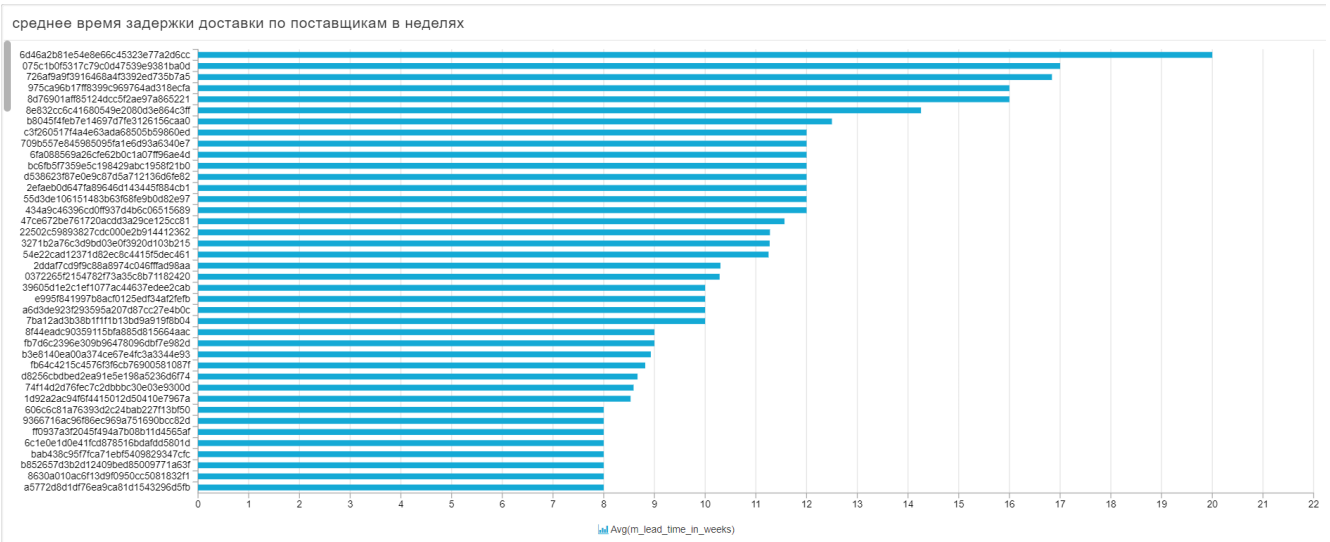

Рисунок 9 – Вкладка Lead time

Столбчатая диаграмма на рисунке 9 показывает, насколько в среднем каждый поставщик задерживал время доставки (в неделях).

8. Вкладка Orders per month

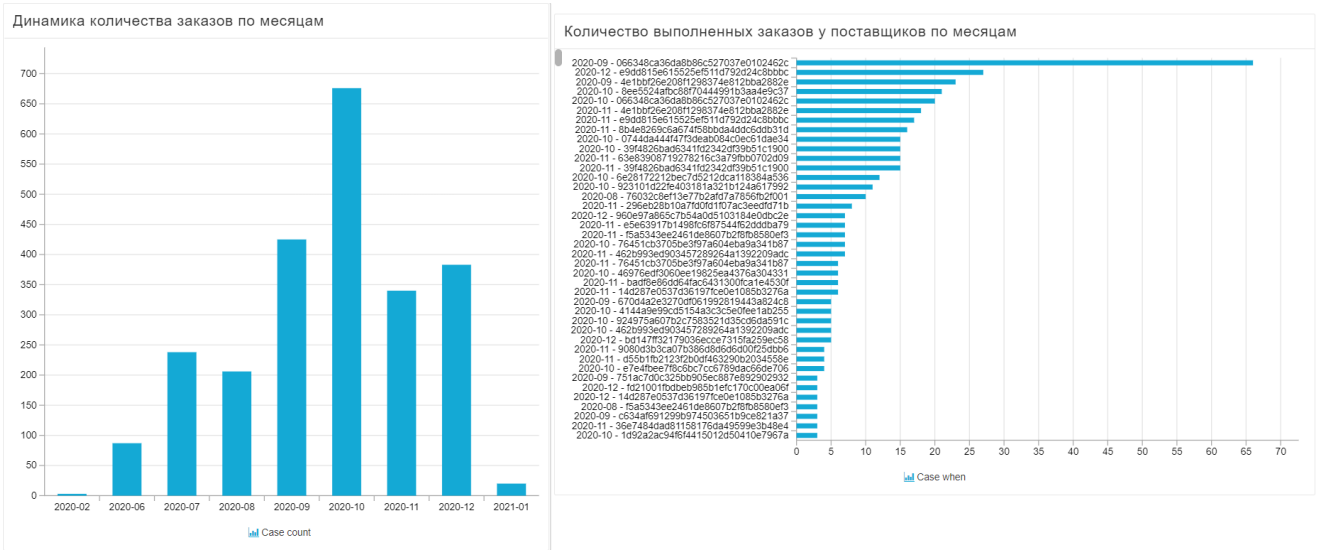

Рисунок 10 – Вкладка Orders per month

Столбчатая диаграмма слева на рисунке 10 отображает сколько заказов было сделано за каждый месяц, а схема справа показывает количество заказов у определенных поставщиков в каждом месяце.

2.7 Выводы из анализа

По полученным схемам и диаграммам можно сделать некоторые выводы:

- Больше всего времени занимают переходы в статус Order Confirmed из Purchased и в Purchased из Approved to Purchase
- Поставщик 6d46a2b81e54e8e66c45323e77a2d6cc (названия зашифрованы) в среднем задерживал поставки больше всех – на 19 недель
- Поставщик 066348ca36da8b86c527037e0102462c занимает самую большую долю заказов (порядка 66 только за один сентябрь 2020), его задержки по доставке составляют в среднем 6 недель, а с октября по декабрь 2020 около половины всех процессов, связанных с заказами у этого

поставщика, соответствуют эталонному процессу. Среднее время выполнения заказа – 95 часов.

Эти данные передаются бизнес-аналитикам, и они на основе выводов принимают решения по улучшению процессов.

ГЛАВА 3. ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ

В виду того, что еще нигде нет полностью автономного производства, к которому стремится ООО Эррайвал Рус, расчет экономической составляющей может быть только гипотетический.

В среднем на автомобилестроительном заводе в России работает около 9 тыс. человек. Сотрудники заводов в Ленинградской области (именно там в России находится производство автомобилей компании ООО Эррайвал Рус) получают от 50 до 95 тыс. руб. Автономное производство может позволить сократить штаб сотрудников примерно до пятисот человек. Это позволит экономить до 612 тыс. руб. ежемесячно.

Стоимость лицензии на ПО Celonis, которое используется для построения карт процессов, составляет 30 тыс. евро в год (по текущему курсу составляет 2 696 715 руб.), разработка своего ПО для тех же целей обойдется примерно в такую же сумму.

Средняя зарплата аналитика данных в России на сентябрь 2020 составляет 54 000 руб. в месяц, в Москве – 133 000 руб. в месяц, в Санкт-Петербурге – 102 000 руб. в месяц. Средняя зарплата аналитика бизнес-процессов в 2021 году в Москве – 88 206 руб. в месяц.

Итого, разработка системы подготовки данных в Москве и ее анализ в системе процесс-майнинга может обойтись примерно в 445 932,25 руб. в месяц.

ГЛАВА 4. ТЕХНИКА БЕЗОПАСНОСТИ

Проектирование, разработка и использование представленной информационной системы подразумевает работу за компьютером. Для безопасной работы нужно следовать правилам техники безопасности при работе с электрооборудованием (компьютерами, принтерами, интерактивными устройствами).

1. Общие требования охраны труда

1.1. К самостоятельной работе с ПК допускаются пользователи после прохождения ими инструктажа на рабочем месте, обучения безопасным методам работ и проверки знаний по охране труда, прошедшие медицинское освидетельствование на предмет установления противопоказаний к работе с компьютером.

1.2. При работе с ПК рекомендуется организация перерывов на 15 минут через каждые 1 час 15 минут работы.

1.3. При работе на ПК могут воздействовать опасные и вредные производственные факторы:

- физические: повышенный уровень электромагнитного излучения;
- повышенный уровень статического электричества;
- повышенная яркость светового изображения;
- повышенный уровень пульсации светового потока;
- повышенное значение напряжения в электрической цепи, замыкание которой может произойти через тело человека;
- повышенный или пониженный уровень освещенности;
- повышенный уровень прямой и отраженной блескости;
- психофизиологические: напряжение зрения и внимания;

интеллектуальные и эмоциональные нагрузки;

- длительные статические нагрузки; монотонность труда.

1.4. Запрещается находиться возле ПК в верхней одежде, принимать пищу и курить, употреблять во время работы алкогольные напитки, а также быть в состоянии алкогольного, наркотического или другого опьянения.

1.5. Пользователь должен знать месторасположение первичных средств пожаротушения и уметь ими пользоваться.

1.6. О каждом несчастном случае пострадавший или очевидец несчастного случая немедленно должен известить коллег.

1.7. Пользователь должен знать местонахождения медицинской аптечки, правильно пользоваться медикаментами; знать инструкцию по оказанию первой медицинской помощи пострадавшим и уметь оказать медицинскую помощь. При необходимости вызвать скорую медицинскую помощь или доставить в медицинское учреждение.

1.8. При работе с ПК пользователи должны соблюдать правила личной гигиены.

1.9. За невыполнение данной инструкции виновные привлекаются к ответственности согласно правилам внутреннего распорядка или взысканиям, определенным Кодексом законов о труде Российской Федерации.

1.10. По всем вопросам, связанным с работой компьютера, следует обращаться к руководителю.

2. Требования охраны труда перед началом работы

2.1. Перед включением используемого на рабочем месте оборудования пользователь обязан:

2.1.1. Осмотреть и привести в порядок рабочее место, убрать все посторонние предметы, которые могут отвлекать внимание и затруднять работу.

2.1.2. Проверить правильность установки стола, стула, подставки под ноги, угол наклона экрана монитора, положения клавиатуры в целях исключения неудобных поз и длительных напряжений тела. Особо обратить

внимание на то, что дисплей должен находиться на расстоянии не менее 50 см от глаз (оптимально 60-70 см).

2.1.3. Проверить правильность расположения оборудования.

2.1.4. Кабели электропитания, удлинители, сетевые фильтры должны находиться с тыльной стороны рабочего места.

2.1.5. Убедиться в отсутствии засветок, отражений и бликов на экране монитора.

2.1.6. Убедиться в том, что на устройствах ПК (системный блок, монитор, клавиатура) не располагаются сосуды с жидкостями, сыпучими материалами (чай, кофе, сок, вода и пр.).

2.1.7. Включить электропитание в последовательности, установленной инструкцией по эксплуатации на оборудование; убедиться в правильном выполнении процедуры загрузки оборудования, правильных настройках.

3. Требования охраны труда во время работы

3.1. В течение всего времени работы со средствами компьютерной и оргтехники пользователь обязан:

- содержать в порядке и чистоте рабочее место;

- следить за тем, чтобы вентиляционные отверстия устройств ничем не были закрыты;

- выполнять требования инструкции по эксплуатации оборудования;

- соблюдать, установленные расписанием, трудовым распорядком регламентированные перерывы в работе, выполнять рекомендованные физические упражнения.

3.2. Пользователю не рекомендуется во время работы:

- отключать и подключать интерфейсные кабели периферийных устройств;

- класть на устройства средства компьютерной и оргтехники: бумаги, папки и прочие посторонние предметы;

- прикасаться к задней панели системного блока (процессора) при включенном питании;

- отключать электропитание во время выполнения программы, процесса;

- допускать попадание влаги, грязи, сыпучих веществ на устройства средств компьютерной и оргтехники;

- производить самостоятельно вскрытие и ремонт оборудования;

- производить самостоятельно вскрытие и заправку картриджей принтеров или копиров;

- работать со снятыми кожухами устройств компьютерной и оргтехники;

- располагаться при работе на расстоянии менее 50 см от экрана монитора.

3.3. При работе с текстами на бумаге, листы надо располагать как можно ближе к экрану, чтобы избежать частых движений головой и глазами при переводе взгляда.

3.4. Рабочие столы следует размещать таким образом, чтобы видеодисплейные терминалы были ориентированы боковой стороной к световым проемам, чтобы естественный свет падал преимущественно слева.

3.5. Освещение не должно создавать бликов на поверхности экрана.

3.6. Продолжительность работы на ПК без регламентированных

перерывов не должна превышать 1-го часа 15 минут. Во время регламентированного перерыва с целью снижения нервно-эмоционального напряжения, утомления зрительного аппарата, необходимо выполнять комплексы физических упражнений.

4. Требования охраны труда в аварийных ситуациях

4.1. Обо всех неисправностях в работе оборудования и аварийных ситуациях сообщать ответственному лицу.

4.2. При обнаружении обрыва проводов питания или нарушения целостности их изоляции, неисправности заземления и других повреждений

электрооборудования, появления запаха гари, посторонних звуков в работе оборудования и тестовых сигналов, немедленно прекратить работу и отключить питание.

4.3. При поражении пользователя электрическим током принять меры по его освобождению от действия тока путем отключения электропитания и до прибытия врача оказать потерпевшему первую медицинскую помощь.

4.4. В случае возгорания оборудования отключить питание, позвонить в пожарную охрану, после чего приступить к тушению пожара имеющимися средствами.

ЗАКЛЮЧЕНИЕ

В дипломной работе рассмотрена автоматизированная система построения бизнес-процессов на основании данных компании ООО «Эррайвал Рус». В ходе дипломного проектирования были решены следующие задачи:

- изучена организация ООО «Эррайвал Рус» и проанализирована область работы. Компания стремится к полностью автономному производству, для этого необходимо составить карту всех процессов и собрать о них все данные.

- описаны требования к системе и выбрано ПО для реализации. Система соответствует всем описанным требованиям и при небольшой корректировке может быть адаптирована для анализа данных других отделов.

- разработана система для подготовки данных к анализу. Консольное приложение выполняет свою функцию разметки и консолидации данных в нужных таблицах.

- данные о процессах подготовлены к анализу. Сформирован CSV файл, который будет использоваться в системе процесс-майнинга.

- на основании данных построены карты бизнес-процессов, диаграммы и графики. В системе Celonis сформировано восемь бордов с графиками и диаграммами для разных целей.

- получены инсайты и сделаны выводы на основании анализа данных. Сделаны определенные выводы касательно того, сколько занимают времени определенные процессы и с какими поставщиками лучше продолжать работать, а с какими нет.

При выполнении работы также рассмотрены вопросы, которые касаются технико-экономического обоснования выполнения проекта, а также вопросы безопасности жизнедеятельности и правил работы с оборудованием.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Медведев В.И. Особенности объектно-ориентированного программирования на C++/CLI, C# и Java. – Казань: РИЦ «Школа», 2008. – 360 с.: ил. – (Серия «Современная прикладная математика и информатика»).

2. Медведев В.И. Программирование на C++, C++.NET/C# и .NET компоненты. – Казань: Мастер Лайн, 2006. – 296 с.: ил.

3. Медведев В.И. Программирование на C++,C++.NETиC# (Серия “Современная прикладная математика и информатика”). – Казань: Мастер Лайн, 2005. – 270с.: ил.

4. Байдачный С.С. .NET Framework. Секреты создания Windows-приложений. – М.: СОЛОН-Пресс, 2004. – 496 с.: ил.

5. Гербердт Шилдт. C#: учебный курс. – СПб.: Питер; К.: Издательская группа BHV, 2003. – 512 с.: ил. СПб.: Питер, 2002. – 464 с.

6. Петцольд Ч. Программирование для Microsoft Windows на C#. В 2-х томах/Пер. с англ. – М.: Издательско-торговый дом “Русская Редакция”, 2002. – 576 + 624 с.: ил.

7. П. Наутон, Г. Шилдт. Java 2. Наиболее полное руководство в подлиннике. – СПб.: БХВ-Петербург, 2000. – 1072 с.: ил.

8. Рамбо Дж., Якобсон А., Буч Г. UML: специальный справочник. – СПб.: Питер, 2002. – 656 с.

9. http://koptelov.info/publikatsii/process-mining-biznes-protsessy/

10. https://aws.amazon.com/ru/lambda/features/

11. https://processmi.com/programms/celonis-sistema-process-mining/

12. https://docs.microsoft.com/ru-ru/microsoft-edge/visual-studio-code/

13. https://web-creator.ru/articles/postgresql

ПРИЛОЖЕНИЕ А. ПРОГРАММНЫЙ КОД

```
private static string Host = "localhost";
        private static string User = "postgres";
        private static string DBname = "pan_Diploma";
        private static string Password = "123";
        private static string Port = "5432";

        static void Main(string[] args)
        {
            // Build connection string using parameters from
portal
            string connString =
                String.Format(
"Server={0};Username={1};Database={2};Port={3};Password={4};SSLM
ode=Prefer",
                    Host,
                    User,
                    DBname,
                    Port,
                    Password)
            using (var conn = new NpgsqlConnection(connString)
            {
                Console.Out.WriteLine("Opening connection");
                conn.Open();
                int onoff = 0;
                Console.WriteLine("\nDo you want to add query
files?   y/n");
                switch (Console.ReadLine())
                {
                    case "y":
                        onoff = 1;
                        break;
                    case "n":
                        onoff = 0;
                        break;
                }
                while (onoff == 1)
                {
                    Console.WriteLine("------------ Drop scripts
for tables here");

                    // получаем файл и все его строки записываем
в массив
                    string queryPath = Console.ReadLine();
                    FileInfo fileInf = new FileInfo(queryPath);
                    String[] queryArray =
File.ReadAllLines(queryPath);
```

```
                    // определяем первую часть запроса, в
котором создается таблица. Ищем номер строки, на которой этот
запрос заканчивается по символу ;
                    string checkchar = ";";
                    int Arlength = queryArray.Length;
                    int RowN = 0; // номер сроки, в которой
находится нужный символ
                    for (int i = 0; i < Arlength; i++)
                    {
                        if (queryArray[i].Contains(checkchar))
                        {
                            RowN = i;
                            break;
                        }
                    }
                    // из всех строк до этого символа формируем
одну строку запроса
                    String queryString = "";
                    for (int i = 0; i <= RowN; i++)
                    {
                        queryString += queryArray[i];
                    }
                    // отправляем в БД бполученную строку с
запросом на создание таблицы
                    using (var command = new
NpgsqlCommand(queryString, conn))
                    {
                        command.ExecuteNonQuery();
                        Console.Out.WriteLine("Finished creating
table");
                        queryString = "";
                    }

                    // потом отправляем по одной все остальные
строки с добавлением данных в таблицу
                    int AddedRowsN = 0;
                    Console.Out.WriteLine("Adding Data. Please
wait...");
                    for (int i = RowN + 1; i > RowN; i++)
                    {
                        if (i < Arlength)
                        {
                            queryString += queryArray[i];
                            using (var command2 = new
NpgsqlCommand(queryString, conn))
                            {
                                command2.ExecuteNonQuery();
                                AddedRowsN++;
```

```
                                    Console.Write("\r" +
AddedRowsN);
                                }
                            }
                        }
                        Console.WriteLine(AddedRowsN + " rows were
added");
                        // даем возможность пользователю добавить
еще файлов
                        Console.WriteLine("\nDo you want to add more
files?   y/n");
                        switch (Console.ReadLine())
                        {
                            case "y":
                                onoff = 1;
                                break;
                            case "n":
                                onoff = 0;
                                break;
                        }
                    }
                    Console.WriteLine("\n*** Press RETURN to start
creating table ***");
                    Console.ReadLine();

                    // Создание таблицы activities
                    string createActQuery = "CREATE TABLE IF NOT
EXISTS activities (case_id text, timestamp timestamp with time
zone, status text);";
                    using (var command = new
NpgsqlCommand(createActQuery, conn))
                    {
                        command.ExecuteNonQuery();
                        Console.Out.WriteLine("Finishing creating
table activities");
                    }

                    // создание таблицы cases
                    string caseQuery = "CREATE TABLE IF NOT EXISTS
cases AS SELECT z.po_number ||'_'|| z.sku AS case_id, z.product,
x.d_type, z.supplier, z.jira_request, z.required_by,
z.received_date, z.m_lead_time_in_weeks, z.m_lead_time_in_days,
z.item_type, z.next_delivery_date, x.d_created, x.updated,
x.location, x.total_amount, x.tax_amount FROM
airtable_po_gr_value z  JOIN dear_po x ON z.po_number =
x.order_number";
                    using (var command = new
NpgsqlCommand(caseQuery, conn))
                    {
```

```
                    command.ExecuteNonQuery();
                    Console.Out.WriteLine("Finishing creating
table cases");
                }

                // Обозначение нужных переменных
                string getPOnumber = "SELECT po_number, sku FROM
airtable_po_gr_value";
                List<String> POnumber = new List<string>();
                List<String> Itemnumber = new List<string>();
                string IssueKey = "";
                String[,] JiraResArray = new string[15, 4];


                // Получения списка всех PO в лист
                NpgsqlCommand GetPOQ = new
NpgsqlCommand(getPOnumber, conn);
                NpgsqlDataReader Dreader =
GetPOQ.ExecuteReader();
                var o = 0;
                for(int i=0; i < 90000; i++)
                {
                    try
                    {
                        Dreader.Read();
                        POnumber.Add(Dreader.GetString(0));
                        Itemnumber.Add(Dreader.GetString(1));
                    }
                    catch { break; }
                }
                Dreader.Close();

                // В этом цикле мы проверяем каждый PO_number из
списка
                for (int i = 0; i < POnumber.Count; i++)
//POnumber.Count
                {
                    // Получение ключа запроса в таблице jira
для текущего PO
                    string getIssueKey = "SELECT jira_request
FROM airtable_po_gr_value WHERE po_number = '" + POnumber[i] +
"' limit 1";
                    NpgsqlCommand GetIssueQ = new
NpgsqlCommand(getIssueKey, conn);
                    NpgsqlDataReader Dreader2 =
GetIssueQ.ExecuteReader();
                    Dreader2.Read();
                    try
                    {
```

```
                    IssueKey = Dreader2.GetString(0);
                }
                catch { Dreader2.Close(); continue; }
                Console.WriteLine(IssueKey);
                Dreader2.Close();

                // Получение данных по текущему PO
                string activityQuery = "SELECT
cl_created_from, cl_created, cl_status_from, cl_status_to FROM
jira_purchases_changelog WHERE issue_key = '" + IssueKey + "'";
                NpgsqlCommand ActivityQ = new
NpgsqlCommand(activityQuery, conn);
                NpgsqlDataReader Dreader3 =
ActivityQ.ExecuteReader();

                // В этом цикле происходит перезапись всех
полученных данных в массив для дальнейшей работы
                for (int ii = 0; ii < 11; ii++)
                {
                    Dreader3.Read();
                    try
                    {
                        for (int j = 0; j <= 3; j++)
                        {
                            try
                            {
                                JiraResArray[ii, j] =
Dreader3.GetString(j);
                            }
                            catch { JiraResArray[ii, j] =
Dreader3.GetDateTime(j).ToString(); }
                        }
                    }
                    catch { continue; }
                }
                // Отображение данных из массива
                if (JiraResArray[0, 0] != null)
                {
                    for (int aa = 0; aa < 11; aa++)
                    {
                        for (int jj = 0; jj <= 3; jj++)
                        {

Console.Write(String.Format("{0,5} | ", JiraResArray[aa, jj]));
                        }
                        Console.WriteLine();
                    }
                    Dreader3.Close();
                }
```

```
                    else
                    {
                        Dreader3.Close();
                        //Console.ReadLine();
                        continue;
                    }

                    // Получение значений для записи в таблицу
                    String status = "", timestamp = "";
                    try
                    {
                        for (int z = 0; z <= 15; z++)
                        {
                            for (int x = 0; x < 2; x++)
                            {
                                switch (x)
                                {
                                    case 0:
                                        try
                                        {
                                            if ((JiraResArray[z,
0] == JiraResArray[z - 1, 1]) && (JiraResArray[z, 2] ==
JiraResArray[z - 1, 3]))
                                            {
                                                timestamp =
JiraResArray[z, 1].ToString();
                                                status =
JiraResArray[z, 3].ToString();
                                                x = 2;
                                            }
                                        }
                                        catch
                                        {
                                            timestamp =
JiraResArray[z, 0].ToString();
                                            status =
JiraResArray[z, 2].ToString();
                                        }
                                        break;
                                    case 1:
                                        try
                                        {
                                            timestamp =
JiraResArray[z, 1].ToString();
                                            status =
JiraResArray[z, 3].ToString();
                                        }
                                        catch { z = 16; }
                                        break;
```

```
                                    }
                                    Console.WriteLine("STATUS   " +
status);
                                    Console.WriteLine("TIME   " +
timestamp);

                                    string checkQuery = "SELECT *
FROM public.activities WHERE case_id = '" + POnumber[i] + "_" +
Itemnumber[i] + "' AND timestamp = '" + timestamp + "' AND
status = '" + status + "'";
                                    NpgsqlCommand CheckQ = new
NpgsqlCommand(checkQuery, conn);
                                    NpgsqlDataReader CHreader =
CheckQ.ExecuteReader();
                                    //CHreader.Read();
                                    if (!CHreader.Read())
                                    {
                                        CHreader.Close();
                                        string rowsActivityQuery =
"INSERT INTO public.activities (case_id, timestamp, status)
VALUES ('" + POnumber[i] + "_" + Itemnumber[i] + "', '" +
timestamp + "', '" + status + "');";
                                        using (var command = new
NpgsqlCommand(rowsActivityQuery, conn))
                                        {

command.ExecuteNonQuery();

Console.Out.WriteLine("Row added");
                                        }
                                    }
                                    else { z = 16; CHreader.Close();
break; }

                                    //Console.ReadKey();
                                }
                            }
                        } catch { Console.WriteLine("error"); }

                } Console.WriteLine("All rows were successfully
added to activities!");
            }
                Console.WriteLine("\n*** Press RETURN to exit
***");
            Console.ReadLine();
        }
```

Дипломная работа выполнена мной самостоятельно. Все использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют соответствующие ссылки на них.

«___» _____2021 г _____ (_____) (подпись студента)