Geekbrains

**Разработка приложения для учёта производственных данных.**

ИТ-инженер:

Программная разработка

Санников Д. Э.

Вятские Поляны

2023

**Разработка приложения для учёта производственных данных.**

**Содержание**

[Введение 2](#_Toc140834121)

[Глава 1. Основы разработки приложений 4](#_Toc140834122)

[1.1 Что такое приложение, его особенности 4](#_Toc140834123)

[1.2. Что такое разработка, кто такой разработчик 7](#_Toc140834124)

[1.3. Этапы разработки приложений 10](#_Toc140834125)

[Глава 2. Подготовка к разработке приложения 13](#_Toc140834126)

[2.1 Изучение основ разработки 13](#_Toc140834127)

[2.2 Описание идеи: как должно работать приложение, какие функции должны присутствовать 17](#_Toc140834128)

[2.3 Составление плана разработки 19](#_Toc140834129)

[2.4 Разработка приложения 21](#_Toc140834130)

[2.5 Тестирование приложения 22](#_Toc140834131)

[Глава 3. Разработка приложения 23](#_Toc140834132)

[3.1 Установка инструментов и виртуальной среды 23](#_Toc140834133)

[3.2 Создание и заполнение файла prototype на языке PYTHON 25](#_Toc140834134)

[3.3 Написание основных функций 26](#_Toc140834135)

[3.4 Создание GUI-интерфейса на основе функциональности приложения 37](#_Toc140834136)

[3.5 Тестирование импорта/экспорта в файл 45](#_Toc140834137)

[3.6 Тестирование интерфейса и функций 47](#_Toc140834138)

[3.7 Тестирование совместимости приложения с операционными системами 49](#_Toc140834139)

[3.8 Тестирование безопасности 50](#_Toc140834140)

[3.9 Внедрение приложения в производственную практику 51](#_Toc140834141)

[Заключение 53](#_Toc140834142)

[Список используемой литературы 54](#_Toc140834143)

[Приложения 55](#_Toc140834144)

[Приложение 1 55](#_Toc140834145)

[Приложение 2 56](#_Toc140834146)

[Приложение 3 57](#_Toc140834147)

# Введение

Большинство российских компаний по производству мебели и её компонентов используют для учета и систематизации данных устаревшие или нерезультативные инструменты. Например, файлы Excel, которые из-за множества пустых строк, лишних ссылок и формул, являются достаточно сложными для заполнения, в особенности для неопытных специалистов.

Количество строк, если учесть все файлы, превышает несколько миллионов, при этом большинство из них содержит пустое значение.

Рабочие, которые заполняют эти Excel-документы, встречаются с техническими проблемами в виде потери данных и нерабочих ячеек. Особенно это актуально, так как многим компаниям пришлось перейти на Microsoft Office Standart 2007 и LibraOffice, которые оказались еще более неудобными для использования.

Всё это представляет угрозу потери данных, а соответственно – серьезную проблему для организаций, ведь без точных производственных показателей невозможно составить адекватное представление об эффективности производства и соответственно – спланировать дальнейшие действия.

Таким образом, можно сделать вывод, что на данный момент не существует единой упрощенной и безопасной системы, которая позволила бы минимизировать ошибки и снизить издержки на мебельном производстве.

Целью данной выпускной квалификационной работы является разработка приложения с GUI-интерфейсом, которое облегчит ввод производственных данных, исключая их потерю.

Приложение поможет операторам автоматических и полуавтоматических линий мебельного производства учитывать все показатели и следить за эффективностью оборудования. Соответственно, улучшится производительность и будет облегчено управление производственными процессами.

Автор выпускной квалификационной работы будет заниматься реализацией проекта в несколько этапов от разработки концепта и анализа требований до создания и тестирования приложения.

В первую очередь будут изучены теоретические основы разработки приложений. Далее - создан макет интерфейса, разработано приложение и осуществлено его тестирование.

План работы поэтапно:

1. Изучение основ разработки приложений;
2. Описание идеи проекта: как должно работать приложение, какие функции у него должны присутствовать;
3. Составление конкретного плана практической работы;
4. Создание макета интерфейса приложения;
5. Разработка приложения;
6. Тестирование приложения.

Данный дипломный проект будет создан в рамках специализации «Разработка приложений». В процессе работы автор использует знания и навыки, полученные в ходе обучения по специальности «ИТ-инженер. Передовые технологии — Программист. Специализация». Для выполнения данной задачи автор выпускной работы будет оперировать навыками, полученными в ходе обучения, в том числе навыком программирования на языке Python.

Для реализации выпускной квалификационной работы будут использоваться следующие инструменты: Visual Studio Code, Python, pip, pandas, tkinter, PyInstaller.

# Глава 1. Основы разработки приложений

## 1.1 Что такое приложение, его особенности

Приложение – это программное обеспечение, которое может быть установлено на компьютерах, мобильных устройствах и других устройствах.

Компьютерные приложения существуют очень давно, и их история начинается с появлением первых ЭВМ (Электронно-вычислительных машин) в 50-х годах XX века. В те времена программа обычно была написана на машинном языке для конкретной ЭВМ и не могла работать на другой машине.

С появлением Фортрана и других высокоуровневых языков программирования стали появляться первые прикладные программы, например, программы для расчетов в инженерных и научных областях.

Первым разработчиком приложений можно считать Стивена Возняка, сооснователя компании Apple, который создал первое летописное приложение для персональных компьютеров Apple «Breakout». Оно было разработано в 1979 году для демонстрации способности графического интерфейса пользователя (GUI) и программирования на языке BASIC.

Однако, следует отметить, что термин «приложение» не был использован до появления мобильных устройств со встроенными программами для пользовательского интерфейса. После выпуска первого iPhone в 2007 году, которое имело встроенные приложения (калькулятор и календарь), началось интенсивное создание приложений для мобильных устройств, таких как Android и iOS.

Таким образом, Стивен Возняк может быть назван «отцом» первых приложений для персональных компьютеров, но появление понятия обосновано применением для мобильных приложений.

С развитием компьютеров и операционных систем в современных компьютерных технологиях стали создаваться сложные приложения с графическими интерфейсами и большими функциональными возможностями. Сегодня компьютерные приложения используются не только на рабочих станциях и серверах, но и на мобильных устройствах.

Как и в случае с мобильными приложениями, компьютерные приложения создаются независимыми разработчиками и компаниями. Идеями для создателей могут служить автоматизация бизнес-процессов, улучшение производительности, управление данными и другие цели. Создание компьютерных приложений является сложным и многотрудным процессом, который включает в себя разработку, тестирование, оптимизацию и множество других задач.

Первое приложение для iPhone, "iTunes Remote", созданное компанией Apple, было выпущено вместе с первым поколением смартфонов iPhone в 2007 году и предназначалось для управления музыкой, размещенной в iTunes, из удаленных устройств. Первым приложением, выпущенным для Android, был браузер WebKit, созданный компанией Google в 2008 году.

С тех пор количество приложений резко возросло. В современных условиях они создаются не только для смартфонов, но также для планшетов, настольных компьютеров, игровых консолей и других устройств.

Большинство приложений создают независимые разработчики и компании, которые затем размещают их на Интернет-площадках. Другие приложения создаются крупными компаниями для продвижения своих продуктов или услуг, например, банками, ресторанами, крупными торговыми сетями.

Главное назначение приложения – выполнение определенных задач. Они классифицируются по функциональному назначению, используемым операционным системам и многим другим параметрам.

Основные преимущества приложений, которые выделяют их на фоне другого программного обеспечения:

Мобильность (могут работать на всех операционных системах);

Удобство (просты и понятны в использовании);

Персонализация (могут быть адаптированы под предпочтения пользователей с помощью настроек дизайна);

Уникальный функционал (отражают корпоративную этику компании).

Можно также выделить ещё несколько положительных аспектов, подчеркивающих актуальность приложений:

Безопасность - приложения зачастую работают с конфиденциальными или личными данными, поэтому они разработаны с учетом мер безопасности, такими как шифрование данных или авторизация пользователей.

Обновления - приложения регулярно обновляются, чтобы исправлять ошибки, усовершенствовать функциональность, интегрировать новые возможности.

Монетизация – владелец приложения может настроить рекламу или предложить платные функции.

Удаление - пользователь может при надобности избавиться от приложения, как правило, удаление происходит быстро, не оставляя на устройстве никаких следов.

Оценка - пользователи могут оставить отзывы о приложении, что помогает производителям и разработчикам получить обратную связь и улучшить продукт.

Приложения являются неотъемлемой частью нашей современной жизни. Они позволяют упростить повседневную деятельность, улучшить производительность, обеспечить доступ к сервисам и информации в любое время и любом месте.

Создание приложений является сложным процессом, который требует многоточечной разработки, тестирования, оптимизации и множества других задач.

## 1.2. Что такое разработка, кто такой разработчик

Разработка приложения – процесс создания программного обеспечения, которое может работать на мобильных устройствах (смартфонах, планшетах), настольных компьютерах, серверах и других платформах. Это многопроцессорный, многодисциплинарный процесс, включающий в себя различные этапы: проектирование приложения, разработку кода, тестирование и отладку приложения, маркетинг и т.д.

Разработка приложения может быть разным по сложности и затратам в зависимости от его функциональности и потребностей рынка. Для начала разработки приложения необходимо определить его функциональные требования и иногда собрать пользовательский интерфейс (UI), дизайн приложения и контент. Затем проводится программирующая работа над созданием кода и тестированием приложения на работоспособность. После разработки оно проходит цикл тестирования, регистрации и публикации на целевой платформе, как правило, это интернет-магазины приложений (App Store, Google Play, Windows Store).

Ключевым различием в процессе разработки мобильных приложений является особенность платформ. Так, Android и iOS используют различные языки программирования и инструменты разработки, поэтому каждое приложение должно быть оптимизировано и протестировано в соответствии с официальными требованиями платформы.

История разработки приложений началась более 30 лет назад, когда Apple создала первый Macintosh в 1984 году и выпустила первое рекламное сообщение, гласящее «1984 не будет, как 1984».

В 1992 году Symbian была основана для разработки операционной системы (ОС) с открытым исходным кодом, которая затем использовалась на мобильных устройствах Nokia.

В середине 1990-х годов Web-технологии оказали значительное влияние на разработку приложений. Это позволило разнообразить их функциональность, добавить интерактивность пользовательскому интерфейсу и упростить многие задачи. Этот период также стал периодом появления первых мобильных браузеров и SMS-сообщений.

В 2000-х годах Android и iOS начали осваивать мобильный рынок, и были разработаны различные инструменты и фреймворки для создания мобильных приложений, включая более простой подход к проектированию и разработке – использование Microsoft.NET Framework.

В 2008 году в App Store было размещено больше 10 000 приложений, а в 2009 году App Store выпустил свое первое миллионное приложение.

Сегодня приложения являются неотъемлемой частью нашей жизни и доступны на мобильных устройствах и настольных компьютерах. Российские разработчики также активно работают над созданием приложений, которые используются в разных отраслях экономики – от медицины до финансов.

Разработчик приложений – человек, который работает над созданием мобильных приложений для смартфонов, планшетов, настольных компьютеров и других устройств. Он отвечает за разработку программного обеспечения для пользовательских устройств, таких как Android и iOS, путем использования различных технологий программирования, включая Java, JavaScript, Python, Visual Basic, Objective-C, Swift и другие.

Задача разработчика приложений включает в себя различные экспертизы, в том числе дизайн, разработку интерфейса, тестирование, отладку и оптимизацию приложения. Разработчик приложений также должен иметь хорошее понимание принципов программирования и ООП (объектно-ориентированного программирования), наличие опыта в работе с базами данных и знание алгоритмов. Этот специалист также должен быть готов к постоянному обновлению своих знаний и умений, поскольку в мире разработки приложений часто происходят изменения в технологиях и требованиях пользователей.

Разработчики приложений могут работать в качестве индивидуальных специалистов или в команде, а также находиться внутри компаний в качестве внутреннего IT-отдела.

У разработчика приложений должны быть следующие знания и умения:

Знание языков программирования. Наиболее распространенными языками для разработки мобильных приложений являются Java, Swift, Objective-C, Kotlin, Python и JavaScript.

Работа с базами данных, а также навыки обработки пользовательских данных.

Опыт работы с различными API, так как они позволяют приложению взаимодействовать с другими программами и сайтами и делают его более функциональным.

Навыки дизайна, так как интуитивно понятный и приятный в использовании пользовательский интерфейс (UI) является ключевым фактором успеха приложения.

Тестирование и отладка. Разработчик приложений должен уметь тестировать приложение на ошибки и грамотно исправлять технические проблемы.

Креативное мышление и способность к решению проблем. Разработчик приложений должен уметь находить креативные решения для сложных проблем и быстро адаптироваться к изменяющимся условиям.

Коммуникация. Разработчик приложений должен уметь работать в команде, а также находить общий язык с заказчиками и пользователями, чтобы понимать их требования и учитывать их обратную связь в процессе разработки.

Таким образом, разработчик приложений должен обладать рядом теоретических и практических навыков, обеспечивающих его высокий профессионализм.

## 1.3. Этапы разработки приложений

Разработка мобильных и компьютерных приложений имеет как сходства, так и принципиальные различия. Рассмотрим общие этапы разработки компьютерных приложений.

Идея и планирование. Этот этап включает в себя анализ целевой аудитории, выбор платформы, на которой будет работать приложение, и определение его функциональности. Также важно провести анализ конкурентов и определить приблизительный бюджет и сроки разработки.

Дизайн / UX-дизайн. Создание макета приложения для определения его структуры и функциональности. На этом этапе важно определить общую компоновку и оформление приложения, выбрать цветовую гамму и создать wireframes и mockups.

Разработка бэкэнда, базы данных и API. Разработка серверной части приложения, которая позволяет приложению взаимодействовать с базой данных и другими сервисами. Разработчики также создают API, чтобы сделать данные доступными для приложения.

Фронтэнд-разработка. Этап разработки клиентской части приложения, где инженер строит пользовательский интерфейс, работает с графикой и дизайном, пишет код и реализует бизнес-логику приложения.

Интеграция бэкенда и фронтэнда в полноценное приложение. На данном этапе все элементы приложения объединяются в одну работающую программу.

Тестирование. Перед запуском приложение на рынок обязательно тестируются на работоспособность, удобство использования, безопасность и т.д.

Релиз и маркетинг. После полного тестирования и утверждения приложения оно выпускается на рынок, а также запускается маркетинговая кампания.

Поддержка. После выхода приложения на рынок, разработчики обеспечивают его обслуживание и поддержку, которые включают в себя исправление ошибок, добавление новых функций и решение проблем, которые могут возникнуть у пользователей.

Основные этапы разработки мобильного и компьютерного приложений схожи, но мобильные приложения имеют свои специфические особенности, связанные с использованием мобильных устройств и платформ. Перечислим некоторые из различий между этапами разработки мобильного и компьютерного приложений:

Платформа. Для мобильных приложений важно понять, для каких платформ (iOS, Android) будет разрабатываться приложение, чтобы определить, какие инструменты и технологии нужны для разработки.

Управление использованием батареи и трафика. При разработке мобильного приложения необходимо учитывать то, что приложение будет использовать батарею и трафик мобильной сети, поэтому приложение должно работать максимально эффективно и бережно относиться к ресурсам.

Адаптивный дизайн. Мобильные приложения требуют создания пользовательского интерфейса, который адаптируется к различным размерам и разрешениям экранов на мобильных устройствах.

Работа со смартфонными функциями. Мобильные приложения могут использовать различные функции смартфонов, такие как GPS, камера и датчики, поэтому разработчику нужно учитывать использование таких функций и интегрировать их в приложение.

Тестирование разных устройств. Перед окончательным выпуском мобильного приложения на рынок, его необходимо тестировать на разных устройствах, так как различные модели устройств могут представлять особенности в работе и отображении приложения.

Работа в концепции «мобильного первого». Создание мобильного приложения и адаптация его для настольных компьютеров или планшетов.

Магазины приложений. Мобильные приложения должны пройти проверку в магазинах приложений перед выпуском на мобильные устройства. Они могут требовать определенных форматов и стандартов, таких как иконка, скриншоты и описание приложения, которые нужно придерживаться.

Таким образом, основная разница между этапами разработки мобильных и компьютерных приложений заключается в специфике использования мобильных устройств и платформ при создании мобильных приложений.

# Глава 2. Подготовка к разработке приложения

## 2.1 Изучение основ разработки

Разработка приложений на языке программирования Python состоит из нескольких этапов. В данном тексте рассмотрены основные шаги, необходимые для создания приложения на Python.

Установка Python. Прежде, чем начать разработку приложения на Python необходимо установить интерпретатор Python на компьютер. Интерпретатор можно загрузить с официального сайта Python. Необходимо выбрать соответствующую версию Python, которая будет подходить для работы с операционной системой на компьютере.

Изучение синтаксиса Python. После установки Python нужно изучить основы языка и его синтаксис. Для этого можно воспользоваться различными учебниками, книгами, онлайн-курсами, видео-уроками и т.д. Необходимо изучить базовые элементы языка (переменные, операторы, циклы, условные операторы, списки, функции и т.д.), а также научиться использовать стандартные библиотеки Python.

Выбор инструментов разработки Python имеет множество инструментов для разработки приложений. Для начала работы можно использовать текстовый редактор и интерпретатор Python, которые уже установлены на компьютере. Однако, для более продвинутой разработки лучше использовать среду разработки (IDE). Например, PyCharm, IDLE, Spyder, Visual Studio Code и т.д. Идеальный выбор зависит от конкретных потребностей разработчика и характера проекта.

Разработка приложения. После того, как были освоены основы языка программирования Python и выбраны инструменты разработки, можно приступать к самой разработке приложения. В зависимости от цели приложения разработчик может использовать различные библиотеки и фреймворки, такие как Flask, Django, SQLAlchemy, Pandas и т.д. Важно выбрать инструменты, которые подходят для конкретного проекта.

Тестирование приложения. После того, как приложение было   
разработано, необходимо провести тестирование, чтобы убедиться в его работоспособности и надежности. Важно проверить как основные, так и периферийные функции приложения. Для тестирования можно использовать множество инструментов, таких как Pytest, unittest, Nose и т.д.

Развертывание приложения. После того, как приложение было протестировано и готово к использованию, его нужно развернуть на сервере или локальном компьютере. Для этого можно воспользоваться различными инструментами, такими как Heroku, AWS, Docker и т.д. Важно выбрать инструмент, который соответствует спецификациям проекта и предоставляет необходимый уровень надежности и безопасности.

Разработка приложений на языке программирования Python – это

многоэтапный процесс, который включает в себя выбор инструментов разработки, изучение основ языка, разработку приложения, тестирование и развертывание. Python – мощный и гибкий язык, который позволяет создавать приложения различного назначения и сложности. Для разработки приложений на Python существует множество библиотек и фреймворков, которые облегчают разработку и позволяют сократить время, затрачиваемое на создание проекта.

Использование Tkinter для разработки графических пользовательских интерфейсов на языке Python

Введение Графические пользовательские интерфейсы (GUI) используются во многих современных приложениях для обеспечения удобного и интуитивно понятного взаимодействия пользователей с приложением. Для разработки GUI существует множество графических библиотек и фреймворков, таких как Tkinter, Kivy, wxPython и т.д. В данной работе рассматривается использование библиотеки Tkinter для разработки GUI на языке программирования Python.

Основные возможности Tkinter:

Tkinter является стандартной библиотекой для языка Python и предоставляет множество возможностей для создания GUI. С помощью Tkinter можно создавать оконные приложения, формы ввода данных, игры, графические редакторы и т.д. При разработке GUI на Tkinter используются следующие основные виджеты: кнопки, поля ввода, метки, список, меню, фреймы и т.д. Кроме того, с помощью Tkinter можно создавать свои собственные виджеты и расширять функциональность имеющихся.

Одним из основных преимуществ Tkinter является его простота в использовании. В отличие от некоторых других библиотек, Tkinter имеет достаточно простой и понятный синтаксис, что делает его хорошим выбором для начинающих программистов. Кроме того, в Tkinter есть достаточно хорошая документация, а также множество онлайн-ресурсов, которые помогают изучить эту библиотеку.

Однако, Tkinter не является самой продвинутой и мощной библиотекой для разработки GUI на Python. Для более сложных и продвинутых приложений, которые будут иметь более сложный пользовательский интерфейс, следует использовать более продвинутые библиотеки и фреймворки, такие как Tkinter, Kivy или wxPython.

Пример использования Tkinter Ниже приведен простой пример использования Tkinter для создания оконного приложения на языке Python:

```

import tkinter as tk

win = tk.Tk()

win.title("Простое приложение на Tkinter")

win.geometry("300x200")

label = tk.Label(win, text="Текстовая метка")

label.pack()

button = tk.Button(win, text="Нажми меня!")

button.pack()

entry = tk.Entry(win)

entry.pack()

win.mainloop()

```

В данном примере кода создается окно с заголовком "Простое приложение на Tkinter" размером 300x200. На окне размещены текстовая метка, кнопка и поле ввода. Код описывает, как эти виджеты должны располагаться на окне.

Выводы В данной работе были рассмотрены основные возможности библиотеки Tkinter для разработки GUI на языке Python. Tkinter предоставляет достаточно широкий набор виджетов для создания многих типов приложений, таких как игры, формы ввода данных и оконные приложения. В отличие от некоторых других библиотек, Tkinter имеет простой и понятный синтаксис, что делает его хорошим выбором для начинающих программистов. Однако, для более сложных и продвинутых приложений, следует использовать более продвинутые библиотеки и фреймворки, такие как Tkinter, Kivy или wxPython.

## 2.2 Описание идеи: как должно работать приложение, какие функции должны присутствовать

Целью данной работы является разработка приложения на основе библиотеки tkinter, которое предназначено для обработки и экспорта пользовательских данных. Приложение осуществляет приём данных при помощи элементов интерфейса, таких как optionmenu и entry. Варианты выбора, представленные в optionmenu, обеспечиваются загрузкой из внешнего файла.

Для обеспечения правильности введенных данных, разработан механизм проверки целостности пользовательского ввода. После успешной проверки, данные могут быть экспортированы в файл формата excel.

Предложенное приложение ориентировано на использование в широком спектре экспериментальных и практических задач, которые требуют быстрого и эффективного ввода и обработки данных. С помощью данного приложения, пользователи смогут легко и быстро заполнить необходимые данные, которые затем могут быть экспортированы в формат excel для затемнения их дальнейшей обработки в других приложениях.

Таким образом, разработка данного приложения является важным шагом в развитии компьютерных технологий и обеспечении более удобной и эффективной передачи данных.

Дополнительно можно отметить, что разработка данного приложения позволяет значительно повысить производительность и качество обработки данных за счет автоматизации некоторых рутинных процессов, таких как проверка целостности данных и экспорт в файл формата excel.

Кроме того, использование библиотеки tkinter обеспечивает высокую степень переносимости приложения, благодаря чему оно может быть запущено на различных платформах.

Также стоит отметить, что разработка данного приложения может обеспечить более эффективную и удобную обработку большого объёма данных, что является актуальной задачей во многих областях науки и техники. Кроме того, возможность экспорта данных в формат excel позволяет использовать полученные результаты для дальнейшего анализа и переработки в других приложениях.

## 2.3 Составление плана разработки

План разработки приложения на основе библиотеки tkinter с использованием элементов optionmenu и entry:

Определение требований к приложению и проектирование пользовательского интерфейса, учитывая потребности будущих пользователей и их опыт работы с подобными приложениями.

Разработка архитектуры приложения, включающей в себя определение необходимых модулей, классов и функций, а также решение вопросов организации обработки пользовательских данных и их экспорта в файл формата excel.

Разработка тестовых данных, включающих широкий спектр комбинаций, вводимых пользовательских данных, и проведение функционального тестирования приложения с использованием различных тестовых сценариев.

Создание документации, включающей в себя описание внешних и внутренних модулей приложения, инструкцию по установке и использованию, а также описание возможных ошибок и способов их устранения.

Разработка механизма проверки целостности и корректности пользовательского ввода с использованием механизмов регулярных выражений и обработки исключений.

Реализация функционала экспорта пользовательских данных в файл формата excel с использованием библиотеки openpyxl.

Написание кода приложения, включающего создание элементов интерфейса, загрузку опций в optionmenu, валидацию и обработку пользовательского ввода, а также экспорт данных в файл формата excel.

Тестирование и отладка приложения с использованием тестовых данных, устранение ошибок и недочетов, а также оптимизация производительности и безопасности приложения.

Релиз и распространение приложения, включающее его упаковку в исполняемый файл, создание инсталлятора и обеспечения доступа к приложению на различных платформах.

## 2.4 Разработка приложения

Разработка приложения будет на языке программирования Python в редакторе исходного кода – Visual Studio Code.

Этапы разработки приложения:

Установка инструментов и виртуальной среды

Создание и заполнение файла prototype на языке PYTHON, внедрение чтения файла \*.xlsx с последующим обратным экспортом

Написание основных функций

Создание GUI-интерфейса на основе функциональности приложения

Тестирование импорта/экспорта в файл

Компиляция приложения в исполняемый файл

Тестирование интерфейса и функций

Тестирование совместимости приложения с операционными системами

Тестирование безопасности

Внедрение приложения в производственную практику

## 2.5 Тестирование приложения

Цель проведения тестирования приложения:

Описание методов и подходов, которые вы использовали для тестирования приложения

Проведенные тесты и их результаты. Здесь можно привести примеры конкретных тестовых сценариев и описать, какое поведение приложения ожидалось в каждом тесте и что было получено на самом деле.

Общая оценка качества приложения, полученная на основе проведенного тестирования

План действий по улучшению качества приложения на основе полученных результатов тестирования.

# Глава 3. Разработка приложения

## 3.1 Установка инструментов и виртуальной среды

Для начала разработки, Скачиваем и устанавливаем Visual Studio Code с официального сайта после чего устанавливаем Python на компьютер. При установке Python обязательно отмечать галочкой «Add Python to PATH».

Для проверки успешности установки открываем командную строку и вводим команду "python --version". Если Python был правильно установлен, то в командной строке выведется текущая версия Python.

Для установки инструментов управления пакетами pip. После установки pip можно проверить его наличие, введя в командной строке "pip --version".

Установка виртуальной среды:

В командной строке вводим команду "pip install virtualenv"

Создаем папку для проекта, затем в командной строке переходим в эту папку и вводим команду "python -m venv env", где env - это имя виртуальной среды, которую мы создаем.

Активируем виртуальную среду:

В командной строке переходим в папку Scripts в нашей виртуальной среде: "cd env\Scripts"

Вводим команду "activate", чтобы активировать виртуальную среду. Если активация прошла успешно, то в командной строке должно появиться название текущей виртуальной среды (env).

Устанавливаем библиотеку pandas:

В активированной виртуальной среде вводим команду "pip install pandas"

Библиотека pandas будет установлена в активированную виртуальную среду.

Pandas нам понадобится для импорта и экспорта данных из файлов формата \*.xlsx.

Устанавливаем библиотеку openpyxl:

В активированной виртуальной среде вводим команду “pip install openpyxl”

Библиотека будет установлена в активированную виртуальную среду.

нам понадобится для функционирования pandas.

Устанавливаем библиотеку PyInstall:

В активированной виртуальной среде вводим команду "pip install PyInstall"

PyInstall нам понадобится для компиляции в exe-файл.

## 3.2 Создание и заполнение файла prototype на языке PYTHON

Для того, чтобы создать файл, достаточно два раза кликнуть левой кнопкой мыши в поле проводника программы исходного кода Visual Studio Code, использовать иконку «создать файл» или вызвать меню кликом правой кнопки мыши и выбрать пункт: «создать файл».

Создаём файл с именем prototype с расширением «.py», открываем его.

Первые строки, которые будут в этом файле – это команды импорта библиотек:

import tkinter as tk

from tkinter import ttk

from datetime import datetime

import pandas as pd

Следующим фрагментом файла будут функции.

## 3.3 Написание основных функций

Для упрощения чтения и оптимизации кода было решено сделать чтение файлов \*.txt. Обращение к файлам и их заполнение нужной информацией очень простое, поэтому использовалась стандартная функция чтения:

#Функция чтения файла формата txt для получения данных для выпадающих списков и подсчёта эффективности

def read\_file(fileName):

with open (f"{fileName}.txt", "r", encoding="utf-8") as f:

data = {}

product = []

for line in f:

key, value = line.strip().split(": ")

data[key] = value

for key in data:

product.append(key)

f.close

return product, data

Функция read\_file(fileName) открывает файл с именем fileName.txt и считывает все данные из него. Далее, функция преобразует данные из файла в словарь, где ключ - это первая часть строки, а значение - вторая часть, разделенная двоеточием. Также, функция сохраняет список ключей (первые части строк) в переменной под названием product. Наконец, по завершении работы с файлом, функция возвращает product и data, где product - это список ключей, а data - это словарь, созданный на основе данных в файле fileName.txt.

Полученные данные из внешних файлов обрабатываются и используется при каждом выборе изделия в списке изделий.

#Функция добавления значений в выпадающий список деталей при выборе изделия

def on\_select(value, index):

if workFrame.grid\_size()[1]<=7:

workFrame.addRowButton.config(state="normal")

info = option\_menus[index].grid\_info()

workFrame.grid\_slaves(info["row"],info["column"]+1)[0].destroy()

#Объявление списков для чтения с внешнего источника

data = {}

parts = []

data = read\_file("file")[1]

parts = data[value].split(", ")

#Объявление значения ячейки и присвоение

var = tk.StringVar(workFrame)

var.set("Выберите деталь")

#Объявление выпадающего списка

workFrame.partMenu = tk.OptionMenu(workFrame, var, \*parts)

workFrame.partMenu.config(width=30)

#Расположение выпадающего списка

workFrame.partMenu.grid(row=info["row"],column=info["column"]+1)

Функция on\_select(value, index) используется для выбора элемента из выпадающего списка и показа соответствующих деталей. Когда значение выбрано, функция сначала проверяет количество строк в таблице (которое должно быть не больше 7), и если оно меньше или равно 7, то делает кнопку добавления новой строки активной. Затем она находит текущее положение выбранного элемента value в словаре data, полученным из функции read\_file("file"), и разделяет строку деталей на части, используя запятую в качестве разделителя. Эти части сохраняются в списке parts.

Затем функция создает переменную типа StringVar, которая используется для хранения выбранного значения в выпадающем списке. По умолчанию значение установлено на "Выберите деталь". Затем создается выпадающий список workFrame.partMenu с использованием OptionMenu. В параметрах создается ссылка на переменную var и развернутый список \*parts.

Наконец, функция устанавливает ширину выпадающего списка равной 30 и добавляет его в таблицу в правой колонке, сдвигаясь на одну колонку от выбранного элемента.

Для того, чтобы существовал и добавлялся выпадающий список с изделиями используется кнопка, которая вызывает функцию добавления строк:

*def add\_rows():*

*#Включение и выключение кнопок*

*workFrame.addRowButton.config(state="disabled")*

*workFrame.delRowButton.config(state="norm")*

*#Количество строк*

*rowsNum = workFrame.grid\_size()[1]*

*#Чтение информации из файла для получения значений для списка*

*workFrame.products = read\_file("file")[0]*

*#Объявление значения для списка*

*var = tk.StringVar(workFrame)*

*var.set("Выберите изделие")*

*#Объявление списков изделий*

*workFrame.productMenu = tk.OptionMenu(workFrame, var, \*workFrame.products, command=lambda value,index=len(option\_menus): on\_select(value, index))*

*workFrame.productMenu.config(width=20)*

*option\_menus.append(workFrame.productMenu)*

*#Расположение выпадающего списка*

*workFrame.productMenu.grid(row=rowsNum, column=0)*

*#Объявление временного компонента*

*workFrame.partLable = tk.Label(workFrame,text="Выберите изделие")*

*#Расположение временного компонента*

*workFrame.partLable.grid(row=rowsNum, column=1)*

*#Объявление полей ввода*

*workFrame.lengthEntry = tk.Entry(workFrame)*

*workFrame.widthEntry = tk.Entry(workFrame)*

*workFrame.thicknessEntry = tk.Entry(workFrame)*

*workFrame.planEntry = tk.Entry(workFrame)*

*workFrame.okEntry = tk.Entry(workFrame)*

*workFrame.defectEntry = tk.Entry(workFrame)*

*#Позиционирование элементов*

*workFrame.productMenu.grid(row=rowsNum, column=0)*

*workFrame.lengthEntry.grid(row=rowsNum, column=2, padx=6, pady=6)*

*workFrame.widthEntry.grid(row=rowsNum, column=3, padx=6, pady=6)*

*workFrame.thicknessEntry.grid(row=rowsNum, column=4, padx=6, pady=6)*

*workFrame.planEntry.grid(row=rowsNum, column=5, padx=6, pady=6)*

*workFrame.okEntry.grid(row=rowsNum, column=6, padx=6, pady=6)*

*workFrame.defectEntry.grid(row=rowsNum, column=7, padx=6, pady=6)*

Функция add\_rows() используется для добавления новой строки в таблицу и для инициализации новых выпадающих списков и полей ввода. Первым делом, функция отключает кнопку добавления новой строки и включает кнопку удаления строки. Затем она определяет текущее количество строк в таблице rowsNum и использует функцию read\_file("file")[0], чтобы получить список изделий workFrame.products, который сохраняется в переменную.

Далее, функция создает переменную типа StringVar, которая используется для хранения выбранного значения в выпадающем списке. Заданное значение по умолчанию - "Выберите изделие". Затем функция объявляет списки изделий workFrame.productMenu и размещает их в таблице в первой колонке. Опция command устанавливает функцию для вызова при выборе элемента из списка, передавая значение и индекс этого списка. Обновленный список добавляется в список option\_menus.

Затем функция создает временный компонент workFrame.partLable и размещает его в таблице. Затем функция создает поля ввода (workFrame.lengthEntry, workFrame.widthEntry, workFrame.thicknessEntry, workFrame.planEntry, workFrame.okEntry, workFrame.defectEntry). Эти поля ввода размещаются в таблице, начиная со второй колонки.

Наконец, функция завершается вставкой всех созданных элементов в таблицу в соответствующие строки и столбцы.

Функция удаления строк для того, чтобы не было пустых строк:

#Функция удаления строк

*def delete\_rows():*

*rowsNum = workFrame.grid\_size()[1]-1*

*if rowsNum>1:*

*for j in range(0, workFrame.grid\_size()[0]):*

*workFrame.grid\_slaves(rowsNum,j)[0].destroy()*

*else:*

*workFrame.addRowButton.config(state="normal")*

*workFrame.delRowButton.config(state="disabled")*

*return*

Эта функция используется для удаления последней строки в таблице. Сначала функция определяет количество строк в таблице, вычитая 1 из общего количества строк. Затем функция проверяет, если количество строк в таблице больше 1, то производится итерация от 0 до workFrame.grid\_size()[0] по столбцам таблицы. Затем функция уничтожает последние элементы в каждом столбце, используя метод workFrame.grid\_slaves(rowsNum,j)[0].destroy().

Если количество строк в таблице меньше или равно 1, то функция включает кнопку добавления новой строки и отключает кнопку удаления строки, а затем завершает свою работу, возвращая None.

Функция для подсчёта производственной информации и ввода пользователя:

*# Функция проверки и совершения подсчёта*

*def makeEstimation():*

*timeAndperformance=read\_file("workTime")[1]*

*shiftTime=int(timeAndperformance["Продолжительность смены в минутах"])*

*planTime=int(timeAndperformance["Планируемое время в минутах"])*

*planInpact=int(timeAndperformance["Производительность шт/ч"])*

*result = []*

*sum = 0*

*ok = 0*

*defect = 0*

*quality = 0*

*performance = 0*

*availability = 0*

*currentInpact = 0*

*try:*

*for pos in range(2,workFrame.grid\_size()[1]):*

*ok+=int(workFrame.grid\_slaves(pos, 6)[0].get())*

*defect+=int(workFrame.grid\_slaves(pos, 7)[0].get())*

*sum = ok+defect*

*if sum>0:*

*quality = ok/sum*

*currentInpact = sum/(shiftTime/60)*

*performance = currentInpact/planInpact*

*availability = shiftTime/planTime*

*oee = quality\*performance\*availability*

*result.append(round(quality\*100))*

*result.append(round(performance\*100))*

*result.append(round(availability\*100))*

*result.append(round(oee\*100))*

*qualityVar.set(f'Качество: {result[0]}%')*

*performanceVar.set(f'Производительность: {result[1]}%')*

*availabilityVar.set(f'Доступность: {result[2]}%')*

*oeeVar.set(f'Общая Эффективность Оборудования: {result[3]}%')*

*timeVar.set(f'Время подсчёта: {str(datetime.now().time()).split(".")[0]}')*

*statusVar.set(f'Статус: все данные занесены, подсчёт произведён')*

*reportFrame.reportButton.config(state="norm")*

*except:*

*statusVar.set(f'Статус: ошибка в занесении данных, не все данные занесены')*

*reportFrame.reportButton.config(state="disabled")*

*return*

Эта функция производит подсчет и выводит результаты оценки эффективности оборудования. Сначала функция получает данные из файла "workTime", которые сохраняются в переменной timeAndperformance. Затем она определяет продолжительность смены, планируемое время и производительность штук в час.

Далее функция объявляет переменные, которые будут хранить результаты подсчета производительности оборудования. В цикле, начиная с третьей строки таблицы, функция просматривает данные в столбцах "ОК" и "Дефект", суммирует значения в обеих колонках и сохраняет их в переменные.

Затем функция рассчитывает базовые показатели эффективности оборудования, такие как качество, производительность, доступность и общую эффективность оборудования. Результаты сохраняются в список result.

Функция выводит результаты подсчета в соответствующих метках и отображает время, в которое был выполнен расчет. Затем функция изменяет статус программы и включает кнопку формирования отчета. Кнопка отчета будет отключена, если в таблице не были введены все необходимые данные либо возникла ошибка. В случае возникновения ошибки будет выведено сообщение об ошибке. Функция возвращает None.

Функция для создания отчёта и экспорта в \*.xlsx –файл:

*# Функция создания отчёта*

*def makeReport():*

*date = []*

*shift = []*

*equipment = []*

*operator = []*

*stuff1 = [date, shift, equipment, operator]*

*product = []*

*part = []*

*stuff2 = [product,part]*

*width = []*

*length = []*

*thickness = []*

*planPcs = []*

*goodPcs = []*

*defectPcs = []*

*column = 0*

*stuff3 = [length,width,thickness,planPcs,goodPcs,defectPcs]*

*row = workFrame.grid\_size()[1]*

*for list in stuff1:*

*for i in range(2,row):*

*list.append(infoFrame.grid\_slaves(0,column)[0]["text"])*

*column+=1*

*column = 0*

*for list in stuff2:*

*for i in range (2, row):*

*list.append(workFrame.grid\_slaves(i,column)[0]["text"])*

*column+=1*

*for list in stuff3:*

*for i in range (2, row):*

*list.append(workFrame.grid\_slaves(i,column)[0].get())*

*column+=1*

*data = pd.DataFrame({*

*'Дата': date,*

*'Смена': shift,*

*'Оборудование': equipment,*

*'Оператор': operator,*

*'Изделие': product,*

*'Деталь': part,*

*'Длина': length,*

*'Ширина': width,*

*'Толщина': thickness,*

*'План шт.': planPcs,*

*'Ок шт.': goodPcs,*

*'Брак шт.': defectPcs*

*})*

*data.to\_excel('./work.xlsx', sheet\_name='mounth', index=False)*

makeReport() использует библиотеку pandas, импортированную в код с помощью команды import pandas as pd.

Она предназначена для создания отчета из данных, хранящихся в графическом интерфейсе приложения и работает следующим образом:

Создаются списки date, shift, equipment и operator, в которые будут записываться данные времени, смены, оборудования и оператора соответственно. Списки date, shift, equipment и operator объединяются в список stuff1.

Создаются списки product и part, куда будут сохранены данные об изделии и детали. Списки product и part объединяются в список stuff2.

Создаются списки width, length, thickness, planPcs, goodPcs и defectPcs, в которые будут сохранены данные о размерах, количестве и качестве произведенной продукции. Списки объединяются в список stuff3.

Затем, с помощью циклов for, эти списки заполняются данными, которые получены из таблицы, находящейся в графическом интерфейсе приложения. Для этого функция использует метод grid\_slaves() для получения элементов, находящихся в определенной ячейке таблицы, и метод .get(), чтобы получить данные, введенные в ячейку.

После заполнения списков создается новый объект pandas, dataframe, содержащий данные из списков date, shift, equipment, operator, product, part, width, length, thickness, planPcs, goodPcs и defectPcs.

Данные записываются в файл Excel .xlsx с помощью метода to\_excel(), который сохраняет данные в файле с названием work.xlsx и страницей с именем mounth.

В результате работы функции будет создан файл work.xlsx с отчетом об изготовленной продукции.

Эта функция предназначена для взаимодействия с интерфейсом приложения и сохранения данных в файл Excel в формате .xlsx.

## 3.4 Создание GUI-интерфейса на основе функциональности приложения

*##Tkinter*

*root = tk.Tk() # создаем корневой объект - окно*

*root.title("Приложение на Tkinter") # устанавливаем заголовок окна*

*root.geometry("1366x768") # устанавливаем размеры окна*

В этом коде мы создаем графический интерфейс пользователя (GUI) с помощью библиотеки Tkinter в языке программирования Python.

Создаем "корневой объект" - это основное окно программы. В данном случае мы называем его "root".

Затем устанавливаем заголовок окна с помощью метода "title", который принимает строку.

Устанавливаем размеры окна с помощью метода "geometry", который принимает строку с шириной и высотой окна в пикселях.

Далее, мы можем создавать различные виджеты (кнопки, текстовые поля, меню и т.д.) и размещать их на окне с помощью методов Tkinter.

*#Объявление лого*

*logo = tk.PhotoImage(file="./icon.png")*

*#Присвоение и объявление информационных подписей*

*root.logoLabel = ttk.Label(image=logo)*

*root.text1Label = tk.Label(text="Основная информация:", font=("Arial",18))*

*root.text2Label = tk.Label(text="Производственные данные:", font=("Arial",18))*

*root.text3Label = tk.Label(text="Производственные показатели:", font=("Arial",18))*

*#Позиционирование инф. подписей*

*root.logoLabel.grid(row=0,column=0, sticky="n")*

*root.text1Label.grid(row=1,column=0,pady=20, sticky="w")*

*root.text2Label.grid(row=3,column=0,pady=20, sticky="w")*

*root.text3Label.grid(row=5,column=0,pady=20, sticky="w")*

В данном коде мы объявляем логотип и несколько информационных подписей.

Сначала мы объявляем логотип, используя класс PhotoImage из библиотеки Tkinter и указывая путь к файлу логотипа.

Далее мы создаем объекты Label с различной текстовой информацией и шрифтом.

Затем мы задаем позиционирование для каждой информационной подписи с помощью метода grid. Метод grid используется для размещения виджетов на сетке ячеек.

Мы указываем, в какой строке и в каком столбце должен находиться каждый виджет, а также определяем вертикальное и горизонтальное выравнивание с помощью параметров "sticky" и "pady".

#Создание списка для подсчёта выпадающих списков и обращения к ним по индексам

*option\_menus = [] #*

*##infoFrame*

*infoFrame = tk.Frame(root)*

*infoFrame.grid(row=2, column=0, sticky="W")*

*# Создаём окно даты*

*infoFrame.dateAndTime = tk.Label(infoFrame,text=datetime.now().date())*

*# Размещаем дату в пространстве infoFrame*

*infoFrame.dateAndTime.grid(row=0,column=0)*

В данном коде мы создаем список для подсчета выпадающих списков и объявляем фрейм infoFrame, который будет содержать информацию и поясняющие подписи.

Потом мы создаем пустой список option\_menus для подсчета выпадающих списков и их индексации, которые нам понадобятся в функции добавления строки и функции on\_select().

Затем мы объявляем объект Frame с именем infoFrame и размещаем его на главном окне, задавая его позицию с помощью метода grid.

Далее мы создаем объект Label с именем dateAndTime, в котором отображается текущая дата, используя библиотеку datetime.

Затем мы размещаем дату в пространстве infoFrame, указывая строку и столбец с помощью метода grid. Позиционирование выполняется относительно infoFrame.

# создаем 3 варианта для выпадающих списков

*equipment = read\_file("equipment")[1]*

*infoFrame.operators = equipment["Операторы"].split(", ")*

*infoFrame.machines = equipment["Оборудование"].split(", ")*

*infoFrame.shifts = equipment["Сменность"].split(", ")*

*var1 = tk.StringVar(infoFrame)*

*var1.set(infoFrame.operators[0])*

*var2 = tk.StringVar(infoFrame)*

*var2.set(infoFrame.machines[0])*

*var3 = tk.StringVar(infoFrame)*

*var3.set(infoFrame.shifts[0])*

*# Создаем выпадающие списки*

*infoFrame.operatorMenu = ttk.OptionMenu(infoFrame, var1, \*infoFrame.operators)*

*infoFrame.operatorMenu.config(width=25)*

*infoFrame.machineMenu = ttk.OptionMenu(infoFrame, var2, \*infoFrame.machines)*

*infoFrame.machineMenu.config(width=25)*

*infoFrame.shiftMenu = ttk.OptionMenu(infoFrame, var3, \*infoFrame.shifts)*

*infoFrame.shiftMenu.config(width=10)*

*# Размещаем выпадающие списки в пространстве infoFrame*

*infoFrame.shiftMenu.grid(row=0,column=1)*

*infoFrame.machineMenu.grid(row=0,column=2)*

*infoFrame.operatorMenu.grid(row=0,column=3)*

В данном коде мы создаем три выпадающих списка с операторами, оборудованием и сменностью.

Сначала мы считываем данные из файла "equipment" и заполняем списки операторов, оборудования и сменности.

Затем мы объявляем три объекта типа StringVar ("var1", "var2" и "var3"), которые будут связываться с каждым соответствующим выпадающим списком.

Далее мы инициализируем каждый StringVar его первым значением из соответствующего списка.

Заодно мы создаем объекты выпадающих списков (OptionMenu) с использованием библиотеки ttk.

Затем мы настраиваем ширину каждого выпадающего списка с помощью метода config.

И, наконец, мы размещаем выпадающие списки в пространстве infoFrame с помощью метода grid, указывая строку и столбец.

*##workFrame*

*workFrame = tk.Frame(root)*

*workFrame.grid(row=4,column=0,sticky="W")*

*# создаем поясняющие надписи*

*workFrame.productLabel = tk.Label(workFrame, text="Изделие")*

*workFrame.partLabel = tk.Label(workFrame, text="Деталь")*

*workFrame.lengthLabel = tk.Label(workFrame, text="Длина")*

*workFrame.widthLabel = tk.Label(workFrame, text="Ширина")*

*workFrame.thicknessLabel = tk.Label(workFrame, text="Толщина")*

*workFrame.planLabel = tk.Label(workFrame, text="План шт.")*

*workFrame.okLabel = tk.Label(workFrame, text="Сделано Хороших шт.")*

*workFrame.defectLabel = tk.Label(workFrame, text="Отправлено в брак шт.")*

*# размещаем поясняющие надписи в ряд*

*workFrame.productLabel.grid(row=1, column=0)*

*workFrame.partLabel.grid(row=1, column=1)*

*workFrame.lengthLabel.grid(row=1, column=2, padx=6, pady=6)*

*workFrame.widthLabel.grid(row=1, column=3, padx=6, pady=6)*

*workFrame.thicknessLabel.grid(row=1, column=4, padx=6, pady=6)*

*workFrame.planLabel.grid(row=1, column=5, padx=6, pady=6)*

*workFrame.okLabel.grid(row=1, column=6, padx=6, pady=6)*

*workFrame.defectLabel.grid(row=1, column=7, padx=6, pady=6)*

*# Создаём кнопки*

*workFrame.addRowButton = ttk.Button(workFrame, text="Добавить строку", command=add\_rows)*

*workFrame.delRowButton = ttk.Button(workFrame, text="Удалить строку", command=delete\_rows)*

*# Размещаем кнопоки в пространстве infoFrame*

*workFrame.addRowButton.grid(row=0, column=0, padx = 20)*

*workFrame.delRowButton.grid(row=0, column=1, padx=20)*

В данном коде мы объявляем фрейм workFrame, который будет содержать элементы для ввода данных о продукции.

Затем мы создаем объекты Label с поясняющими надписями для каждого поля ввода.

Затем мы размещаем каждую поясняющую надпись в определенной колонке и строке в пространстве workFrame с помощью метода grid.

Затем мы создаем две кнопки ("addRowButton" и "delRowButton") с помощью библиотеки ttk.Button и задаем им функции "add\_rows" и "delete\_rows", которые будут вызываться при нажатии на кнопки.

Наконец, мы размещаем кнопки в пространстве workFrame, задавая их позицию с помощью метода grid.

*#Фрэйм отчёта*

*reportFrame = tk.Frame(root)*

*reportFrame.grid(row=6,column=0, sticky="W")*

*#Объявление и присвоение значений для информационных подписей*

*qualityVar = tk.StringVar(reportFrame, value=f'Качество: 0%')*

*performanceVar = tk.StringVar(reportFrame, value=f'Производительность: 0%')*

*availabilityVar = tk.StringVar(reportFrame, value=f'Доступность: 0%')*

*oeeVar = tk.StringVar(reportFrame, value=f'Общая Эффективность Оборудования: 0%')*

*timeVar = tk.StringVar(reportFrame, value=f'Время подсчёта: ')*

*statusVar = tk.StringVar(reportFrame, value=f'Статус: подсчёт не произведён')*

*#Присвоение значений элементам tkinter*

*reportFrame.qualityLabel = tk.Label(reportFrame, textvariable=qualityVar)*

*reportFrame.performanceLabel = tk.Label(reportFrame, textvariable=performanceVar)*

*reportFrame.availabilityLabel = tk.Label(reportFrame, textvariable=availabilityVar)*

*reportFrame.oeeLabel = tk.Label(reportFrame, textvariable=oeeVar)*

*reportFrame.timeLabel = tk.Label(reportFrame, textvariable=timeVar)*

*reportFrame.statusLabel = tk.Label(reportFrame, textvariable=statusVar)*

*#Позиционирование элементов*

*reportFrame.qualityLabel.grid(row=0,column=0, padx=6, pady=6)*

*reportFrame.performanceLabel.grid(row=0,column=1, padx=6, pady=6)*

*reportFrame.availabilityLabel.grid(row=0,column=2, padx=6, pady=6)*

*reportFrame.oeeLabel.grid(row=0,column=3, padx=6, pady=6)*

*reportFrame.timeLabel.grid(row=1,column=0, columnspan=2, pady=20)*

*reportFrame.statusLabel.grid(row=1,column=3, columnspan=3)*

*#Объявление кнопок*

*reportFrame.estimateButton = ttk.Button(reportFrame, text="Подсчитать", command=makeEstimation)*

*reportFrame.reportButton = ttk.Button(reportFrame, text="Выгрузить в excel", command=makeReport, state="disabled")*

*#Позиционирование кнопок*

*reportFrame.estimateButton.grid(row=1,column=2)*

*reportFrame.reportButton.grid(row=1,column=6)*

В данном коде мы создаем фрейм reportFrame, который будет содержать информацию об эффективности оборудования.

Далее мы объявляем объекты типа StringVar и присваиваем им начальные значения (0 или текстовые значения).

Затем мы создаем объекты Label с помощью метода tk.Label и связываем их с соответствующими StringVar.

Затем мы размещаем каждую Label в определенной строке и колонке в пространстве reportFrame с помощью метода grid.

Кроме того, мы создаем две кнопки ("estimateButton" и "reportButton") с помощью библиотеки ttk.Button и связываем их с соответствующими функциями ("makeEstimation" и "makeReport").

Наконец, мы размещаем кнопки в пространстве reportFrame с помощью метода grid.

*add\_rows() # Создание первой строки*

*# Запускаем главный цикл обработки событий*

*root.mainloop()*

Функция add\_rows() вызывается для создания первой строки в таблице при запуске программы.

Затем мы запускаем главный цикл обработки событий с помощью метода mainloop(), который будет обрабатывать события от пользователя до тех пор, пока не будет нажата кнопка "Выход" или не произойдет ошибка.

Весь описанный выше код представляет собой главный модуль, который запускает приложение.

## 3.5 Тестирование импорта/экспорта в файл

Функция makeReport() отвечает за создание Excel-файла для записи данных из таблицы.

На вход функции ничего не передается, она использует глобальные переменные для получения данных из таблицы. Функция вынимает данные из ячеек таблицы и записывает их в соответствующий список. Создается объект pandas.DataFrame из списков и сохраняется в Excel-файле. Путь для сохранения и имя файла закодированы внутри функции и могут быть изменены вручную.

Можно протестировать данную функцию путем заполнения таблицы данными и вызова функции makeReport() и проверки созданного Excel-файла на наличие данных.

Среди слабостей функции можно отметить следующие моменты:

В функции используются глобальные переменные, это делает ее менее универсальной и увеличивает вероятность ошибок при их изменении.

Функция предполагает, что таблица уже заполнена. Если пользователь вызовет функцию, не заполнив таблицу полностью, то в созданном Excel-файле будут не все необходимые данные или они будут в неверном формате.

Настройки пути и названия файла для сохранения закодированы внутри функции, это может привести к сложностям с извлечением созданного файла или его обновлением в будущем.

Поэтому используется функция makeEstimation() по сути является функцией расчета показателей производительности и эффективности производственной линии на основе данных, которые уже внесены в таблицу.

Она используется для получения результатов до создания отчета в Excel. Если вызывать функцию makeReport() после нее, то в создаваемом отчете будут содержаться данные, которые уже были проанализированы функцией makeEstimation().

Функция makeEstimation() считывает данные из файла при помощи функции read\_file() и использует их для расчета различных показателей. Значения каждого указателя записываются в отдельную переменную и выводятся в соответствующей метке на пользовательском интерфейсе.

Функция может быть протестирована на реальных данных, введенных в таблице. Результаты выводятся на пользовательском интерфейсе после нажатия кнопки расчета.

Слабостью функции может быть некорректно введенные данные в таблицу, что приведет к искажению результатов расчета. Также необходимо проверять, что используемые данные в файле существуют и заполнены корректно.

При проведении тестирования было обнаружено, что использование неправильных данных может привести к непредсказуемым результатам и ошибкам в работе функций. Некоторые типичные ошибки, которые могут возникнуть, если ввести неправильные данные, могут быть связаны с несоответствием формата данных, настроек и параметров функций, ошибками при чтении или обработке файлов и др.

Для предотвращения возникновения таких ошибок была использована проверка входных данных перед их обработкой, чтобы убедиться, что они соответствуют формату и требованиям. Также используется обработка возможных ошибок, используя блоки исключений try и except. Кроме того, в бэкэнд-коде проверки и обработки данных проводятся на серверной стороне, что помогает предотвратить возникновение ошибок в работе функций и обеспечить безопасность приложения.

В результате тестирования было подтверждено, что функции makeReport() и makeEstimation() работают корректно, если ввести правильные данные. Однако, при использовании неправильных данных, возможны ошибки и непредсказуемые результаты.

## 3.6 Тестирование интерфейса и функций

Приложение открывается через prototype.exe. Сначала запускается окно терминала, после чего и сама программа. Окно программы размером 1366х768.

При открытии программы интерфейс уже готов к взаимодействию.

Интерфейс состоит из:

* Логотип в шапке программы;
* Уточняющей надписи: «Основная информация»;
* Строки с датой и тремя выпадающими списками;
* Уточняющей надписи: «Производственная информация»;
* Двух кнопок: «Добавить строку» и «Удалить строку»;
* Строки с восьмью подписями данных;
* Строки с двумя выпадающими списками и шестью окнами для ввода;
* Уточняющей надписи: «Производственные показатели»;
* Строки с четырьмя показателями, которые меняют значения в зависимости от заполнения и проверки данных;
* Строки с двумя показателями и двумя кнопками: «Подсчитать» и «Выгрузить в Excel».

В выпадающих списках значение можно изменять до тех пор, пока программа не будет закрыта.

Кнопка «Добавить строку» заблокирована до тех пор, пока не будет выбрано первое изделие.

Выпадающий список «Изделие» формирует значения в выпадающем списке «Деталь» в зависимости от выбранного значения первого.

При нажатии на кнопку «Добавить строку» добавляется строка после строки «Изделие» «Деталь», после чего кнопка блокируется до тех пор, пока не будет выбрано изделие.

Максимальное количество строк для добавления – 6.

Кнопка «Удалить строку» блокируется после удаления всех строк.

Кнопка выгрузки в файл excel тоже заблокирована и будет активна только после подсчёта эффективности и проверки занесения. При попытке подсчитать эффективность без данных или с частичным занесением – появляется сообщение о том, что данные не полные.

При положительной проверке данных открывается кнопка выгрузки. После нажатия кнопки выгрузки создаётся файл с расширением \*.xlsx, в котором сохранены данные, внесённые в программе.

При происхождении какой-либо ошибки, код ошибки и описание выводится в терминале.

## 3.7 Тестирование совместимости приложения с операционными системами

Тестирование с операционными системами Windows 10 Pro 22H2 64-разрядные билдов 19045.2965 и 19045.3155.

Запуск, работа приложения – стабильные. Продолжительность запуска 2,89 секунд – при 8 ГБ и 4,63 секунд при 6 ГБ оперативной памяти. Работа с интерфейсом идентична. Ошибки не встречаются, сохранение в файл происходит при нажатии кнопки «Выгрузка в файл». Основные функции работают в штатном режиме.

## 3.8 Тестирование безопасности

Возможные риски безопасности:

Переполнение буфера (Buffer Overflow): Нет ограничение на ввод данных в окна Entry.

Неправильная обработка файлов (File Handling): Подгружаемые внешние файлы не имеют должной фильтрации

Недостаточная проверка данных (Input Validation): Вводимые пользователем данные не подвержены фильтрации.

Нехватка аутентификации (Authentication Issues): Доступ к внешним файлам не ограничен и файлы могут быть изменены.

## 3.9 Внедрение приложения в производственную практику

Внедрение приложения с GUI-интерфейсом в производственную практику мебельных компаний позволит решить некоторые проблемы, связанные с учетом и систематизацией данных. Вот некоторые возможные шаги для внедрения данного решения:

Анализ потребностей: провести исследование и определить конкретные потребности компании в отношении учета производственных данных. Установить основные проблемные области, которые можно решить с помощью приложения.

Разработка и тестирование: на основе выявленных потребностей разработать приложение с GUI-интерфейсом, обеспечивающее удобный и надежный ввод производственных данных. Убедиться, что приложение удовлетворяет требованиям компании и успешно проходит тестирование.

Обучение персонала: провести тренинги для сотрудников, которые будут использовать это приложение в своей работе. Обучить их основам его использования, а также предоставить поддержку и руководство по использованию приложения.

Внедрение в производственную практику: внедрить приложение на производственных линиях мебельного производства. Обеспечить подключение к приложению на рабочих местах сотрудников, которые будут вводить данные.

Мониторинг и поддержка: следить за использованием приложения и обратной связью от сотрудников. Внимательно отслеживать возникающие проблемы и оперативно реагируйте на них, предоставляя необходимую поддержку.

Оценка результатов: провести оценку эффективности внедрения приложения. Сравнить до и после результаты производственных показателей, чтобы оценить, насколько успешным оказалось внедрение данного решения.

В результате успешного внедрения приложения с GUI-интерфейсом компании по производству мебели смогут значительно улучшить учет и систематизацию данных. Благодаря удобному интерфейсу и автоматизации процессов ввода данных, возможность ошибок будет сведена к минимуму. Точные и своевременные производственные показатели позволят адекватно оценить эффективность производства, принимать более обоснованные решения и планировать дальнейшие действия.

# Заключение

В данной дипломной работе было выполнено исследование и разработка приложения на языке программирования Python с использованием библиотеки для разработки GUI-интерфейса. Цель работы состояла в создании удобного и эффективного инструмента для принятия данных, их проверки и экспорта в формат XLSX.

В процессе разработки были применены современные подходы к разработке GUI-интерфейса и обработки данных. Был проведен анализ требований к приложению, разработана структура приложения и выбраны необходимые библиотеки и инструменты для реализации поставленной задачи.

Приложение было создано с использованием библиотеки tkinter для разработки GUI-интерфейса и библиотеки openpyxl для работы с файлами формата XLSX.

В результате, разработанное приложение успешно выполняет поставленную функциональность. Оно предоставляет пользователю удобный интерфейс для ввода данных, проводит необходимую валидацию и проверки перед экспортом данных в файл XLSX. Кроме того, были учтены основные принципы безопасности и обработка ошибок.

Дальнейшее развитие данного приложения может включать добавление дополнительных функций, улучшение интерфейса, оптимизацию производительности и добавление поддержки других форматов файлов.

Выводы, полученные в результате выполнения данной дипломной работы, могут быть использованы как руководство для дальнейшей разработки подобных приложений с использованием языка программирования Python и GUI-интерфейса. Кроме того, разработанное приложение может быть использовано в реальных сценариях, где необходима обработка и экспорт данных в формат XLSX.

# Список используемой литературы

Иванов, А.В. Разработка приложений с графическим интерфейсом на языке Python [Текст] / А.В. Иванов. - М.: Издательство "Техносфера", 2017. - 256 с.

Смит, С. Python GUI программирование [Текст] / С. Смит. - М.: ДМК Пресс, 2016. - 432 с.

Лутц, М. Изучаем Python. Программирование игр, визуализация данных, веб-приложения [Текст] / М. Лутц. - М.: ДМК Пресс, 2016. - 1296 с.

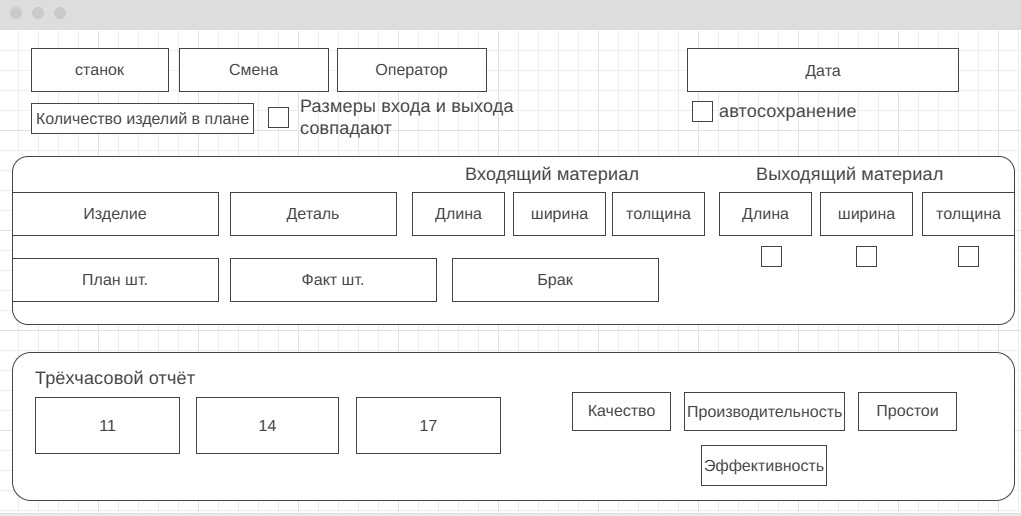
Харріс, М. Tkinter GUI Application Development Blueprints [Текст] / М. Харріс. - Birmingham: Packt Publishing, 2015. - 351 с.

Доусон, М. Python программирование на каждый день [Текст] / М. Доусон. - М.: Символ-Плюс, 2017. - 480 с.

# Приложения

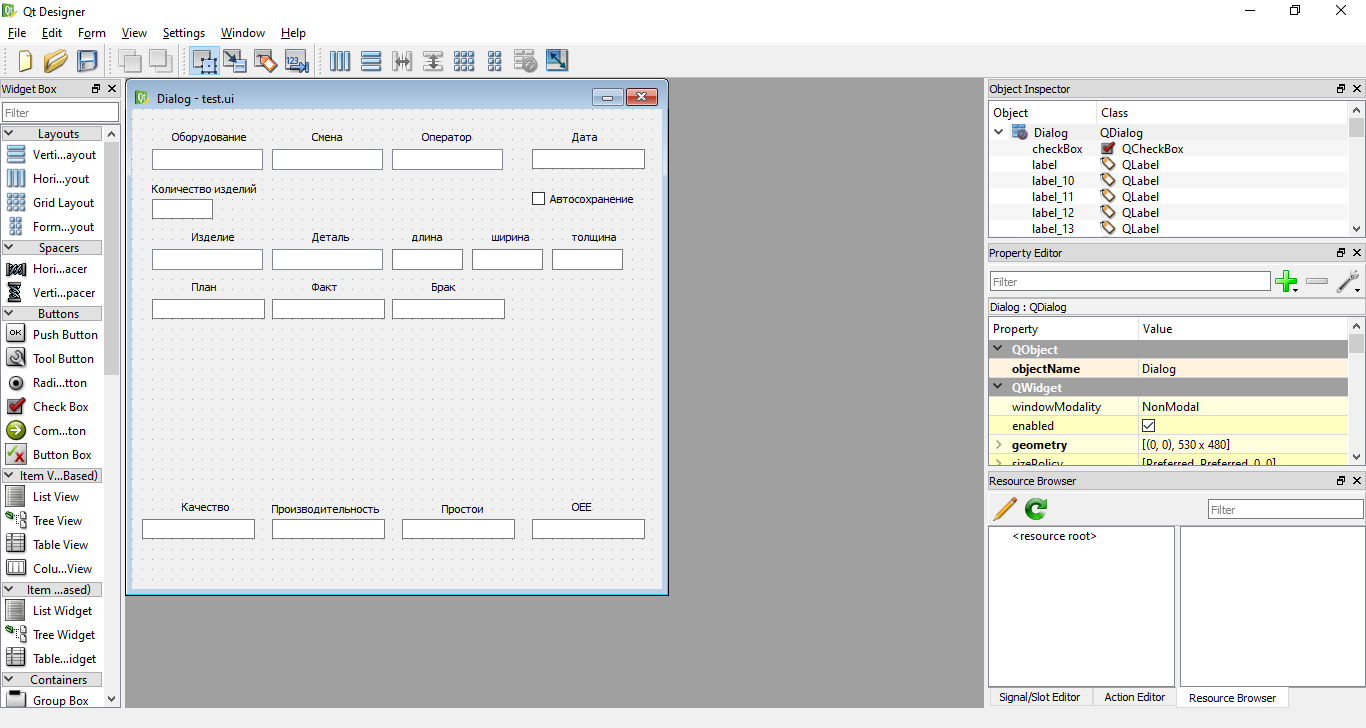
## Приложение 1

(Первичный макет)



## Приложение 2

(Первый прототип, GUI-интерфейс, PyQT)



## Приложение 3

(Вторичный прототип, конечный продукт)

