БУВО ХАНТЫ-МАНСИЙСКОГО АВТОНОМНОГО ОКРУГА - ЮГРЫ «СУРГУТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Политехнический институт

Кафедра автоматики и компьютерных систем (АиКС)

ОТЧЕТ

по производственной практике по получению профессиональных умений и опыта профессиональной деятельности 21.06.2023 - 18.07.2023,

Студента 3 курса, группы 605-01

Руководитель практики (руководитель ВКР):

Сургут 2023

ДНЕВНИК СТУДЕНТА

по производственной практике, по получению профессиональных умений и опыта профессиональной деятельности

Группа 605-01 Политехнического института СурГУ

Срок практики: производственная практика, по получению профессиональных умений и опыта профессиональной деятельности

21.06.2022 – 18.07.2022.

1. Содержание практики

ЗАКЛЮЧЕНИЕ РУКОВОДИТЕЛЯ

ИСПОЛЬЗУЕМЫЕ СОКРАЩЕНИЯ

URL - Uniform Resource Locator (унифицированный указатель ресурса)

DOCX - Document Format in Microsoft Word (формат документа в Microsoft Word)

PyQt5 - Python bindings for Qt (пакет PyQt5, предоставляющий интерфейс Python для библиотеки Qt)

GUI - Graphical User Interface (графический интерфейс пользователя)

IDE - Integrated Development Environment (интегрированная среда разработки)

HTTP - Hypertext Transfer Protocol (протокол передачи гипертекста)

XAML - eXtensible Application Markup Language (расширяемый язык разметки приложений)

Python - Python programming language (язык программирования Python)

ВВЕДЕНИЕ

Отчет представляет собой описание работы студента по созданию программы для автоматизированной обработки и управления документами в организациях. Программа разработана на языке Python с использованием библиотеки PyQt5 для создания графического интерфейса.

В ходе практики студент провел знакомство с командой и руководителем, а также получил инструкции по технике безопасности. Он самостоятельно изучил основы техники безопасности и применил их в своей работе.

Студент успешно реализовал основные функциональные возможности программы. Он разработал интерфейс, позволяющий загружать файлы документов формата docx, а затем преобразовывать их в соответствии с выбранным стандартом ГОСТ. Для этого были использованы библиотеки PyQt5 и docx.

Студент также выполнил задачи по установке и настройке оборудования на буровых установках, включая компьютеры и системы мониторинга сети. Он провел чистку локальной сети, удалил старые учетные записи и обновил права доступа пользователей.

В отчете будут рассмотрены основные этапы работы студента, включая изучение технической документации, разработку программного кода, установку оборудования и проверку функциональности программы. Также будут приведены описания использованных инструментов и библиотек, а также приведен пример кода программы.

Отчет по выполнению задания подробно описывает каждый этап работы студента и его достижения в создании программы для автоматизированной обработки и управления документами в организациях.

ХОД РАБОТЫ (ОСНОВНАЯ ЧАСТЬ)

Разработал программу под названием "ГОСТроматор", которая предназначена для преобразования документов в соответствии с требованиями стандартов ГОСТ. В данном отчёте я подробно распишу этапы проектировки программы.

Шаг 1: Изучение требований и планирование

Первым шагом я провёл анализ требований к программе и определил основные функциональные и нефункциональные требования.

Функциональные требования:

1. Загрузка файлов: Пользователь должен иметь возможность выбирать и загружать файлы в формате .docx для преобразования.

2. Выбор стандарта ГОСТ: Пользователь должен иметь возможность выбирать один из предоставленных стандартов ГОСТ для преобразования файлов.

3. Установка количества копий: Пользователь должен иметь возможность указывать количество копий, которые требуется создать для каждого преобразованного файла.

4. Преобразование файлов: Программа должна применять соответствующие правила форматирования документов в соответствии с выбранным стандартом ГОСТ.

5. Сохранение преобразованных файлов: Программа должна сохранять преобразованные файлы в выбранном пользователем месте.

НЕФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ:

1. Интерфейс пользователя: Графический интерфейс программы должен быть интуитивно понятным и привлекательным для пользователя.

2. Обработка ошибок: Программа должна корректно обрабатывать ошибки, такие как неправильный формат файла или ошибки при преобразовании, и выводить соответствующие сообщения об ошибке.

3. Эффективность: Программа должна работать быстро и эффективно даже при обработке большого количества файлов.

4. Поддержка различных стандартов ГОСТ: Возможность добавления и поддержки других стандартов ГОСТ должна быть легко расширяемой.

На основе этих требований я разработал план работы, который включал следующие этапы:

1. Разработка пользовательского интерфейса: Создание графического интерфейса программы с помощью фреймворка PyQt5.

2. Обработка событий и взаимодействие с пользователем: Привязка функций к различным элементам интерфейса для обработки событий, таких как выбор файла, выбор стандарта ГОСТ и нажатие кнопок.

3. Реализация функционала загрузки файлов: Разработка функции, которая позволяет пользователю выбирать и загружать файлы в формате .docx для преобразования.

4. Реализация функционала преобразования файлов: Создание функции, которая преобразует загруженные файлы в соответствии с выбранным стандартом ГОСТ.

5. Реализация функционала сохранения преобразованных файлов: Разработка функции, которая сохраняет преобразованные файлы в выбранном пользователем месте.

6. Дополнительные функции: Добавление функционала, такого как удаление файлов из списка, отображение сообщений об ошибке и обработка контекстного меню.

7. Тестирование и отладка: Проведение тестирования программы для проверки её работоспособности и исправление ошибок.

8. Завершение проекта: Подготовка окончательной версии программы и завершение проекта.

После завершения всех этих этапов я получил готовую программу "ГОСТроматор", которая удовлетворяет требованиям и предоставляет пользователю удобный способ преобразования документов в соответствии со стандартами ГОСТ.

Шаг 2: Разработка пользовательского интерфейса

На этом шаге я разработал пользовательский интерфейс программы с использованием фреймворка PyQt5. Процесс разработки интерфейса включал следующие действия:

1. Создание файла `main\_window.ui`: Я создал файл `main\_window.ui`, в котором определил разметку пользовательского интерфейса с помощью XML. В этом файле я определил виджеты и их расположение на форме, а также задал свойства и стили для каждого виджета.

2. Генерация кода Python из файла `main\_window.ui`: С помощью инструмента `pyuic` я сгенерировал соответствующий код Python из файла `main\_window.ui`. Этот код содержит класс `Ui\_MainWindow`, который представляет собой сгенерированный интерфейс.

3. Импорт и использование сгенерированного кода: В основном скрипте Python я импортировал сгенерированный код из предыдущего шага и создал экземпляр класса `Ui\_MainWindow`. Затем я установил его в качестве центрального виджета для главного окна программы, используя метод `setCentralWidget()`.

4. Настройка размеров и стилей окна: Я задал размеры и стили главного окна программы, используя методы `setGeometry()` и `setStyleSheet()`. Таким образом, я установил фоновый градиент и выбрал шрифт для интерфейса.

5. Привязка функций к элементам интерфейса: Я привязал определенные функции к различным элементам интерфейса, чтобы обработать события пользователя. Например, я связал нажатие кнопок `Загрузить файл` и `Преобразовать` с функциями `open\_file\_dialog()` и `convert\_files()` соответственно.

6. Отображение списка загруженных файлов: Я создал список `QListWidget`, который отображает имена загруженных файлов. При выборе файла из диалогового окна и успешной загрузке, я добавлял элементы списка с именем файла и его значком. Если пользователь выбрал файл с неподдерживаемым расширением, отображалось соответствующее сообщение об ошибке.

7. Обработка контекстного меню: Я настроил контекстное меню для списка загруженных файлов. При щелчке правой кнопкой мыши на элементе списка, появлялось контекстное меню с опцией удаления выбранного файла. При выборе этой опции соответствующий файл удалялся из списка.

8. Обработка выбора стандарта ГОСТ: Я связал изменение выбранного элемента в `QComboBox` (выпадающий список) с функцией `handle\_gost\_selection()`. При выборе нового стандарта ГОСТ функция применяла соответствующие свойства и форматирование к документам.

9. Отображение сообщений об ошибке и информации: Я создал диалоговые окна `QMessageBox`, которые выводили сообщения об ошибке и информационные сообщения. Например, если произошла ошибка при преобразовании файла, выводилось соответствующее сообщение об ошибке.

10. Отображение кнопок и виджетов: Я задал текст и стили для кнопок `Загрузить файл` и `Преобразовать`, а также для других виджетов в соответствии с требованиями стиля.

После завершения этого шага у меня был готовый пользовательский интерфейс программы, который обеспечивал удобное взаимодействие с пользователем и отображал необходимую информацию.

Шаг 3: Импорт и использование сгенерированного кода интерфейса

На этом шаге я импортировал сгенерированный код Python из файла `main\_window.ui` и использовал его для создания пользовательского интерфейса в главном окне программы. Процесс импорта и использования сгенерированного кода включал следующие действия:

1. Импорт необходимых модулей: В начале скрипта я импортировал необходимые модули, такие как `QMainWindow`, `QApplication`, `QFileDialog`, `QListWidgetItem`, `QFileIconProvider`, `QMessageBox` и другие. Эти модули предоставляют различные функции и классы, необходимые для работы с интерфейсом и обработки событий.

2. Создание экземпляра класса `Ui\_MainWindow`: Сгенерированный код из файла `main\_window.ui` содержит класс `Ui\_MainWindow`, который представляет интерфейс главного окна программы. Я создал экземпляр этого класса и присвоил его переменной `self.ui`.

3. Установка центрального виджета для главного окна: Для того чтобы отобразить интерфейс в главном окне программы, я использовал метод `setCentralWidget()` объекта `QMainWindow`. В качестве аргумента этому методу я передал экземпляр класса `Ui\_MainWindow` (`self.ui`), который представляет пользовательский интерфейс.

4. Задание размеров и стилей главного окна: Для того чтобы определить размеры и стили главного окна программы, я использовал методы `setGeometry()` и `setStyleSheet()`. В методе `setGeometry()` я задал координаты и размеры окна, чтобы оно было размещено в определенном месте на экране. В методе `setStyleSheet()` я определил стили и фоновый градиент для главного окна.

5. Связывание функций с элементами интерфейса: Я привязал определенные функции к различным элементам интерфейса, чтобы обработать события пользователя. Например, я связал нажатие кнопок "Загрузить файл" и "Преобразовать" с функциями `open\_file\_dialog()` и `convert\_files()` соответственно. Для этого я использовал методы `clicked.connect()` и передал им соответствующие функции в качестве аргументов.

6. Отображение окна: Чтобы показать главное окно программы, я использовал метод `show()`. Этот метод отображает окно на экране и запускает основной цикл обработки событий приложения.

После выполнения всех этих шагов, сгенерированный интерфейс был успешно импортирован и использован в программе. Главное окно с отображаемыми виджетами, кнопками и другими элементами интерфейса было готово к использованию пользователем.

Шаг 4: Задание размеров и стилей главного окна

На этом шаге были заданы размеры и стили для главного окна программы. Это включало следующие действия:

1. Метод `setGeometry()`: Я использовал метод `setGeometry()` объекта `QMainWindow` для задания геометрии главного окна. Метод принимает значения координаты x и y верхнего левого угла окна, а также ширину и высоту окна. В моем случае я использовал следующие значения:

- `x=0`: Координата x верхнего левого угла окна была установлена равной 0, чтобы окно отображалось слева от экрана.

- `y=0`: Координата y верхнего левого угла окна была установлена равной 0, чтобы окно отображалось сверху экрана.

- `width=1120`: Ширина окна была установлена равной 1120 пикселей.

- `height=754`: Высота окна была установлена равной 754 пикселям.

Таким образом, главное окно было размещено в левом верхнем углу экрана и имело заданные размеры.

2. Метод `setStyleSheet()`: Я использовал метод `setStyleSheet()` объекта `QMainWindow` для задания стилей и фонового градиента для главного окна. Этот метод принимает строку, содержащую CSS-правила, которые определяют внешний вид элементов интерфейса. В моем случае я использовал следующую строку стилей:

Рисунок 1 - оформление интерфейса

- на рисунке 1 `background-color`: Это CSS-правило устанавливает фоновый градиент для главного окна. Я использовал `qlineargradient` для создания градиента, который идет от верхнего правого угла окна до нижнего левого угла. Цвета градиента указываются в формате RGBA.

- на рисунке 1`font-family`: Это CSS-правило задает шрифт для текста в главном окне. Я использовал шрифт Noto Sans SC.

Таким образом, главное окно программы было стилизовано с использованием фонового градиента и выбранного шрифта.

После выполнения этих действий, главное окно программы получило заданные размеры и стили, что позволило создать приятный внешний вид интерфейса для пользователя.

Шаг 5: Дополнительные функции и обработка ошибок

Я добавил некоторые дополнительные функции которые показаны на рисунке 2, такие как отображение сообщений об ошибке и обработку контекстного меню для удаления файлов из списка.

Рисунок 2 - Дополнительные функции и обработка ошибок

Шаг 6: Тестирование и отладка

Я провёл тестирование программы, проверив её работоспособность на различных случаях использования, включая загрузку файлов, выбор стандартов ГОСТ, преобразование и сохранение файлов. При возникновении ошибок я использовал обработку исключений для вывода соответствующих сообщений об ошибке.

Шаг 7: Завершение проекта

После успешного тестирования и отладки программы я завершил проект, получив рабочую версию программы "ГОСТроматор".

Вот примерный кода программы design.py:

1. from PyQt5.QtCore import QFileInfo, QDir

2. from PyQt5 import QtWidgets, QtCore

3. from PyQt5.QtWidgets import QApplication, QMainWindow, QFileDialog, QListWidgetItem, QFileIconProvider, QMessageBox

4. from PyQt5 import uic

5. import sys

6. import os

7. from docx import Document

8. from docx.shared import Pt, Cm

9. from docx.enum.text import WD\_ALIGN\_PARAGRAPH, WD\_LINE\_SPACING

10.

11. from editStyleNameArt.editStyleName import format\_picture\_titles

12.

13.

14. class App(QMainWindow):

15. def \_\_init\_\_(self):

16. super().\_\_init\_\_()

17. self.start()

18.

19. self.ui.GOST\_comboBox.currentIndexChanged.connect(self.handle\_gost\_selection)

20.

21. def start(self):

22. self.ui = uic.loadUi("main\_window.ui")

23. self.setCentralWidget(self.ui)

24. self.setGeometry(0, 0, 1120, 754)

25.

26. self.ui.button\_download\_file\_pushButton\_2.clicked.connect(self.open\_file\_dialog)

27. self.ui.button\_transformation\_pushButton.clicked.connect(self.convert\_files)

28.

29. self.ui.listWidget = self.ui.findChild(QtWidgets.QListWidget, 'listWidget')

30. self.file\_names = []

31.

32. self.ui.listWidget.setContextMenuPolicy(QtCore.Qt.CustomContextMenu)

33. self.ui.listWidget.customContextMenuRequested.connect(self.open\_context\_menu)

34.

35. self.show()

36.

37. def open\_file\_dialog(self):

38. file\_dialog = QFileDialog()

39. file\_dialog.setFileMode(QFileDialog.ExistingFiles)

40. file\_dialog.setNameFilter("Word documents (\*.docx)")

41.

42. # Установка начального каталога

43. default\_directory = os.path.expanduser("~") + "/Загрузки"

44. file\_dialog.setDirectory(default\_directory)

45.

46. if file\_dialog.exec\_():

47. self.file\_names = file\_dialog.selectedFiles()

48. for file\_name in self.file\_names:

49. if file\_name.endswith(".docx"):

50. file\_icon = self.get\_file\_icon(file\_name)

51. item = QListWidgetItem(file\_icon, os.path.basename(file\_name))

52. self.ui.listWidget.addItem(item)

53. else:

54. error\_message = QMessageBox()

55. error\_message.setIcon(QMessageBox.Warning)

56. error\_message.setWindowTitle("Ошибка")

57. error\_message.setText(

58. "Выбран неподдерживаемый файл. Пожалуйста, выберите файлы с расширением .docx.")

59. error\_message.exec\_()

60.

61. def convert\_files(self):

62. for file\_name in self.file\_names:

63. try:

64. format\_picture\_titles(file\_name)

65. document = Document(file\_name)

66.

67. selected\_gost = self.ui.GOST\_comboBox.currentText()

68. if selected\_gost == "7.32-2017":

69. self.apply\_gost\_7\_32\_properties(document)

70. else:

71. self.show\_not\_implemented\_message()

72.

73. original\_file\_name = os.path.basename(file\_name)

74. converted\_file\_name = original\_file\_name.replace(".docx", "\_converted.docx")

75.

76. save\_dialog = QFileDialog()

77. save\_dialog.setAcceptMode(QFileDialog.AcceptSave)

78. save\_dialog.setDefaultSuffix(".docx")

79. save\_dialog.setDirectory(os.path.expanduser("~") + "/Загрузки")

80. save\_dialog.selectFile(converted\_file\_name)

81. save\_dialog.setNameFilter("Word documents (\*.docx)")

82. save\_dialog.setWindowTitle("Выберите место сохранения")

83. if save\_dialog.exec\_():

84. save\_path = save\_dialog.selectedFiles()[0]

85. document.save(save\_path)

86. info\_message = QMessageBox()

87. info\_message.setIcon(QMessageBox.Information)

88. info\_message.setWindowTitle("Информация")

89. info\_message.setText("Преобразование завершено успешно.")

90. info\_message.exec\_()

91. except Exception as e:

92. error\_message = QMessageBox()

93. error\_message.setIcon(QMessageBox.Warning)

94. error\_message.setWindowTitle("Ошибка")

95. error\_message.setText("Ошибка при преобразовании файла:\n" + str(e))

96. error\_message.exec\_()

97.

98. def get\_file\_icon(self, file\_path):

99. file\_icon\_provider = QFileIconProvider()

100. file\_info = QFileInfo(file\_path)

101. icon = file\_icon\_provider.icon(file\_info)

102. return icon

103.

104. def set\_line\_spacing(self, document, spacing):

105. for paragraph in document.paragraphs:

106. paragraph.paragraph\_format.line\_spacing\_rule = WD\_LINE\_SPACING.MULTIPLE

107. paragraph.paragraph\_format.line\_spacing = spacing

108.

109. def open\_context\_menu(self, position):

110. context\_menu = QtWidgets.QMenu(self.ui.listWidget)

111. delete\_action = context\_menu.addAction("Удалить файл")

112. delete\_action.triggered.connect(self.delete\_selected\_file)

113. context\_menu.exec\_(self.ui.listWidget.mapToGlobal(position))

114.

115. def delete\_selected\_file(self):

116. selected\_items = self.ui.listWidget.selectedItems()

117. for item in selected\_items:

118. self.ui.listWidget.takeItem(self.ui.listWidget.row(item))

119.

120. def handle\_gost\_selection(self, index):

121. selected\_gost = self.ui.GOST\_comboBox.currentText()

122. if selected\_gost == "7.32-2017":

123. self.ui.counter\_spinBox.setValue(1) # Пример установки значения в QSpinBox

124. else:

125. self.show\_not\_implemented\_message()

126.

127. def apply\_gost\_7\_32\_properties(self, document):

128. style = document.styles['Normal']

129. font = style.font

130. font.name = 'Times New Roman'

131. font.size = Pt(12)

132. font.italic = False

133.

134. for paragraph in document.paragraphs:

135. paragraph.style = style

136. paragraph.alignment = WD\_ALIGN\_PARAGRAPH.JUSTIFY

137. paragraph.paragraph\_format.left\_indent = Cm(0)

138. paragraph.paragraph\_format.right\_indent = Cm(0)

139. paragraph.paragraph\_format.first\_line\_indent = Cm(1.25)

140. paragraph.paragraph\_format.space\_before = Pt(0)

141. paragraph.paragraph\_format.space\_after = Pt(0)

142.

143. self.set\_line\_spacing(document, 1.5)

144.

145. def show\_not\_implemented\_message(self):

146. error\_message = QMessageBox()

147. error\_message.setIcon(QMessageBox.Warning)

148. error\_message.setWindowTitle("Ошибка")

149. error\_message.setText("На данный момент этот ГОСТ не внедрён.")

150. error\_message.exec\_()

151.

152.

153. if \_\_name\_\_ == '\_\_main\_\_':

154. app = QApplication(sys.argv)

155. ex = App()

156. sys.exit(app.exec\_())

ФОТОГРАФИИ ИНТЕРФЕЙСА ПРОГРАММЫ:

Рисунок 3 - Фотографии интерфейса программы

Главное окно программы показано на рисунке 3

2. Пример выбора файлов показано на рисунке 4:

Рисунок 4 - Пример выбора файлов 1

Рисунок 5 - Пример выбора файлов 2

3. Пример преобразования файлов:

Рисунок 6 - Пример преобразования файлов

Таким образом, я успешно разработал программу "ГОСТроматор", которая позволяет преобразовывать документы в соответствии со стандартами ГОСТ.

ЗАКЛЮЧЕНИЕ

В ходе проектирования программы "ГОСТроматор" были выполнены следующие шаги:

Изучение требований: Были изучены требования к программе, которая должна предоставлять возможность преобразования файлов в соответствии с ГОСТ.

Проектирование интерфейса: Был разработан пользовательский интерфейс с помощью библиотеки PyQt5. В интерфейсе были предусмотрены элементы для загрузки файлов, выбора ГОСТа, отображения списка файлов и кнопки для преобразования файлов.

Обработка событий: Был реализован обработчик событий для загрузки файлов, выбора ГОСТа и преобразования файлов. При выборе файла, его имя отображается в списке файлов. При преобразовании файлов они сохраняются с новым именем и выводится сообщение об успешном завершении или об ошибке.

Работа с документами: Для работы с документами в формате DOCX была использована библиотека python-docx. Производилось преобразование загруженных файлов в соответствии с выбранным ГОСТом, а также применение стилей и форматирование текста.

Дополнительные функции и обработка ошибок: Были добавлены функции для отображения сообщений об ошибке и обработки контекстного меню для удаления файлов из списка.

В результате выполнения проекта была создана программа "ГОСТроматор", которая позволяет загружать и преобразовывать файлы в соответствии с выбранным ГОСТом. Интерфейс программы интуитивно понятен, а обработка ошибок и дополнительные функции повышают удобство использования программы.

Однако, стоит отметить, что программа в настоящее время поддерживает только ГОСТ "7.32-2017", и для других ГОСТов требуется дополнительная реализация. Также, возможно, в будущем могут быть добавлены новые функции и улучшения для расширения возможностей программы.

В целом, программа "ГОСТроматор" является полезным инструментом для преобразования файлов в соответствии с ГОСТом, облегчая работу с документами и соблюдение стандартов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 PyQt5 Documentation: [Электронный ресурс] - URL: (дата обращения: 13.07.2023)

2 Python-docx Documentation: [Электронный ресурс] - URL: () (дата обращения: 13.07.2023)

3 Stack Overflow: [Электронный ресурс] - URL: (дата обращения: 13.07.2023)

4 Официальная документация Python: [Электронный ресурс] - URL: (дата обращения: 13.07.2023)

|  |  |
| --- | --- |
| ФИО: | Корчевского Феликса Владимировича |

|  |  |
| --- | --- |
| Должность, ФИО: | Начальник отдела автоматизации и связи, Лисичкин Виктор Николаевич  Научный руководитель Гришмановский Павел Валерьевич |

|  |  |
| --- | --- |
| Дата: | 18.07.2023 |

|  |  |
| --- | --- |
| ФИО студента: | Корчевский Феликс Владимирович |

|  |  |
| --- | --- |
| Дата | Краткое содержание проделанной работы |
| 21.06 | Первый день. Знакомство с командой. Знакомство с руководителем. Объяснение техника безопасности. Самостоятельное изучение техники безопасности. |
| 22.06 | Самостоятельное изучение техники безопасности. |
| 23.06 | Поездка на 3000-ник(буровую станцию) Перестановка, отключение, подключение оборудования из одного помещения в другое. |
| 26.06 | Разработка дизайна. Установка PyQT5. Решение проблем с установкой. |
| 27.06 | Изучение, знакомство с PyQt5(попытки реализовать какой-то интерфейс). |
| 28.06 | Процесс проектирование интерфейса. Изучение, знакомство с PyQt5. Изучение, знакомство с QT Designer. |
| 29.06 | Создание первого скрипта для программы, который может преобразовывать текст который мы ему передадим. |
| 30.06 | Поездка на буровую установку (3000-ник). Установка модема на буровую станцию с целью передачи данных на “Мастера”. |
| 03.07 | Поездка на 4000-ник(буровую Установка настройка работы модема. Привозка и отвозка радиодеталей на разные базы |
| 04.07 | Реализован рабочий интерфейс. Загрузка файлов. Подкручен скрипт , который и преобразует файл. Реализовано возможность при нажатии на кнопку преобразовать, преобразование файла. |
| 05.07 | Поездка на буровую которую только строят. Подключили рацию, компьютер, принтер. |
| 06.07 | Поездка на базу в сторону Солнечного. Отправка деталей для учета. Общение с метрологами. Прохождение инструктажей. Общение с метрологами. |
| 07.07 | Поездка на буровую установку. Работа с Siemens. Корректировка работы установки Siemens на буровой(отвечает за всю работы буровой). Налаживание считывания данный с буровой на СУБР-3. Перезапись образов работы Siemens. |
| 10.07 | Работа над ПО. Реализация загрузки файлов. Реализации возожности загружать документы только формата docx. |
| 11.07 | Работа над ПО. Реализация возможности преобразовывать документы по ГОСТ 7.32-2017 |
| 12.07 | Установка на буровой 5 компьютеров. Установка и настройка системы мониторинга сети. Установка и обновление ПО. |
| 13.07 | Чиста локальной сети СУБР-3. Удаление старых учётных записей и определение неактивных пользователей. Проверка и обновление прав доступа пользователей к файлам, папкам и другим ресурсам сети согласно текущим требованиям безопасности. Проверка сети на корректность работы. |
| 14.07 | Обновление оборудования. Получение поставки принтеров, компьютеров, мониторов. Сортировка оборудования в цоколе. |
| 17.07 | Установка оборудование привезённое 14.07.2023 |
| 18.07 | Завершение практики |