**Министерство науки и высшего образования Российской Федерации**

**федеральное государственное автономное образовательное учреждение высшего образования**

**“НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО”  
(УНИВЕРСИТЕТ ИТМО)**

**ЦЕНТР АВТОРИЗОВАННОГО ОБУЧЕНИЯ ИНФОРМАЦИОННЫМ ТЕХНОЛОГИЯМ**

**ИТОГОВАЯ АТТЕСТАЦИОННАЯ РАБОТА**

Веб-приложение для разработки управляющих программ для станков с устройством­ ЧПУ

Автор\_\_\_\_\_\_\_\_\_\_\_Домахин Алексей Владимирович\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия Имя Отчество) (Подпись)

**Центр авторизованного обучения информационным технологиям**

Наименование программы **«Python-разработчик»**

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

(Подпись)

**К защите допустить**

|  |  |  |
| --- | --- | --- |
| *Заместитель директора ЦАО ИТ,*  *к.т.н.* |  | / *Т.В. Зудилова*/ |

Санкт-Петербург, 2023г.

Обучающийся\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Группа 124/02

(Фамилия, И. О.)

Работа принята «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023г.

Работа выполнена с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023г.

Секретарь ИАК \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, И. О.) (подпись)

Листов хранения \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Демонстрационных материалов \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ОГЛАВЛЕНИЕ**

[Введение 1](#_Toc256011441)

[Цель 1](#_Toc256011442)

[Описание системы 1](#_Toc256011443)

[Функции системы 1](#_Toc256011444)

[Структура данных 1](#_Toc256011445)

[Заключение 1](#_Toc256011446)

[Сокращения 1](#_Toc256011447)

[Терминология 1](#_Toc256011448)

[Литература 2](#_Toc256011449)

[Приложение 2](#_Toc256011450)

# Введение

Объектом итоговой аттестационной работы является процесс разработки веб-приложения «CreatorCNC», которое, в свою очередь, отвечает за реализацию управляющих программ на станки с различными устройствами ЧПУ (числовое программное управление). Станки с ЧПУ – это высокотехнологичное оборудование, в котором исполнительные устройства (приводы) управляются автоматически. Приводы, управляемые по определенной программе, приводят в действие рабочий элемент станка – таков принцип работы оборудования с ЧПУ. На сегодняшний день существует обширный парк различных станков и стоек ЧПУ с индивидуальными особенностями.

Актуальность данного проекта заключается в том, что, развитие оборудования с устройствами ЧПУ продвигается гигантскими шагами и в современном мире станки ЧПУ широко применяются в различных отраслях промышленности, как военной, так и гражданской. В следствии этого, спрос на различные редакторы числовых управляющих программ с каждым днем только увеличивается. Данное вэб-приложение поможет начинающим технологам-программистам ЧПУ освоиться в написании управляющих программ на станки с ЧПУ.

Задачи итоговой аттестационной работы:

1. Реализация функционала для разработки управляющих программ для станков с устройством ЧПУ на различные стойки: операция фрезеровки прямоугольника, операция фрезеровки диаметра, операция фрезеровки траекторий, операция резьбофрезерования, операция центровки отверстий, операция сверление отверстий, операция нарезание резьбы метчиком, операция токарной обработки вала и отверстия и точение траекторий.
2. Реализация интерфейса и диалогового окна веб-приложения для работы с пользователем.
3. Разработка базы данных и заполнение ее различными шаблонами фрезерной и токарной обработки.

Проект итоговой аттестационной работы разрабатывался на языке программирования Python. Выбор пал именно на этот язык программирования, так как Python, как показалось, имеет простой и понятный синтаксис для начинающих, что делает его легким в изучении и использовании. Данный язык программирования имеет обширную стандартную библиотеку, которая предоставляет множество функциональных возможностей из коробки, избавляя от необходимости писать сложный код. Так же он имеет популярные внешние библиотеки для разработки вэб-приложений, что как раз подходит для проекта итоговой аттестационной работы.

# Цель

* Изучить основы языка программирования Python, стандартных и внешних библиотек языка.
* Разработать веб-приложение «CreatorCNC», которое отвечает за реализацию управляющих программ на станки с устройством ЧПУ.

# Описание системы

На сегодняшний день существует множество различных библиотек для создания веб-приложений. В языке программирования Python, так же имеются популярные фреймворки для разработки интернет-ресурсов. Ниже перечислены наиболее популярные из них.

* Flask — это микрофреймворк для создания простого и быстрого проекта на языке программирования Python с возможностью масштабирования до сложных приложений. Понятие «микрофреймворк» означает, что в комплекте нет набора инструментов и библиотек, программист может сам установить их в зависимости от задач. Flask используют веб-разработчики на Python. Микрофреймворк подходит для новичков. Позволяет быстро создать веб-приложение, используя только один файл Python. Flask можно использовать при разработке как тренировочных проектов или небольших сайтов, которым не нужен сложный бэкенд, так и API и сложных проектов для электронной коммерции.
* FastAPI — это современный, быстрый (высокопроизводительный) веб-фреймворк для создания API используя Python 3.6+, в основе которого лежит стандартная аннотация типов Python. Ключевые особенности: Скорость: очень высокая производительность, на уровне NodeJS и Go (благодаря Starlette и Pydantic). **Быстрота разработки**: увеличьте скорость разработки примерно на 200–300%. **Меньше ошибок**: сократите примерно на 40% количество ошибок, вызванных человеком (разработчиком). **Интуитивно понятный**: отличная поддержка редактора. **Лёгкость**: разработан так, чтобы его было легко использовать и осваивать. Меньше времени на чтение документации. **Краткость:** сведите к минимуму дублирование кода. Каждый объявленный параметр - определяет несколько функций. Меньше ошибок. **Надежность**: получите готовый к работе код. С автоматической интерактивной документацией.
* Django — это свободный фреймворк для разработки быстрых и безопасных веб-приложений и сайтов на языке Python. Использует шаблон проектирования MVC.

Проект итоговой аттестационной работы разрабатывался с использованием фреймворка Django. Выбор пал именно на эту библиотеку так как, **Django считается лучшим веб-фреймворком, написанным на Python. Этот инструмент удобно использовать для разработки сайтов, работающих с базами данных.** В Django реализован принцип DRY. Благодаря этому сокращается время создания сайтов. То есть при использовании Django вам не нужно несколько раз переписывать один и тот же код. Фреймворк позволяет создавать сайт из компонентов.

Фреймворк Django написан на языке программирования Python, поэтому его структура соответствует особенностям языка. Создатели реализовали в Django паттерн MVC, и он применяется в текущей версии фреймворка. Архитектура MVC позволяет разработчику работать с визуальным представлением и бизнес-логикой приложения отдельно. Кстати, при работе с Django специалисты чаще используют термин MVT — Model-View-Template или модель-представление-шаблон. Компоненты MVT можно использовать независимо друг от друга.

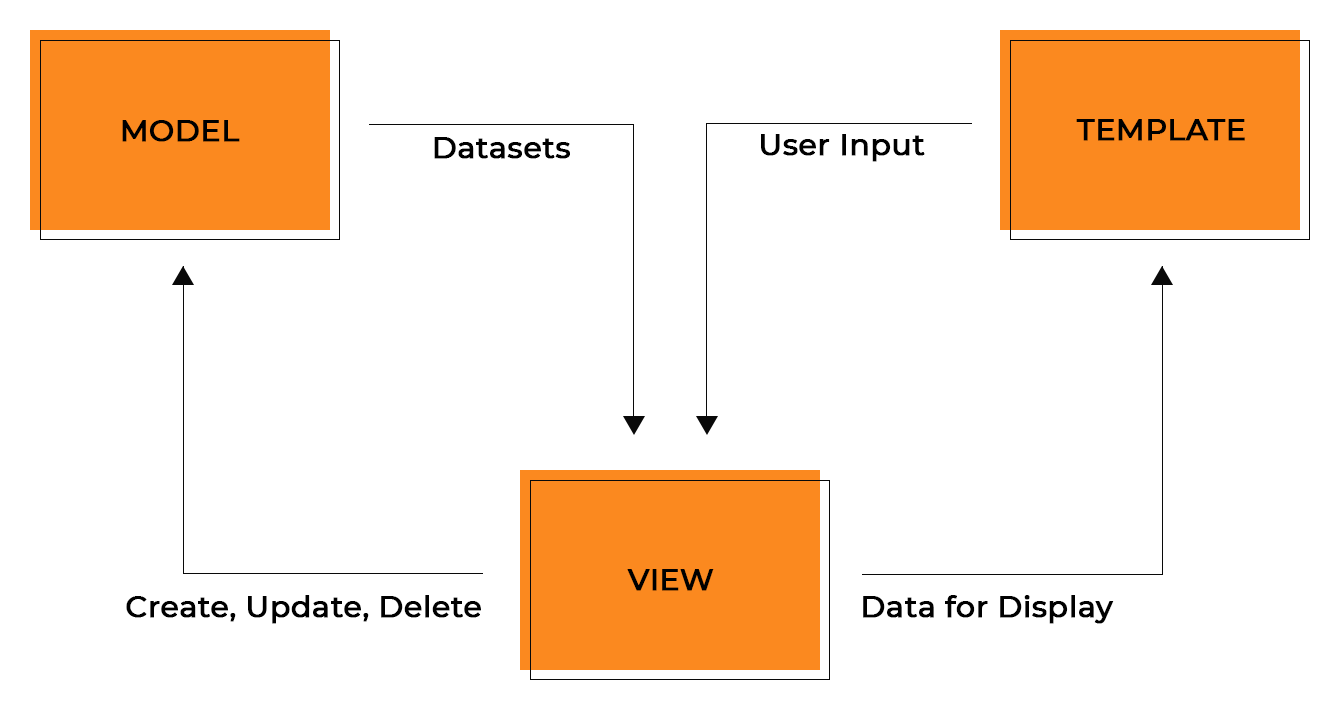


Рисунок 1 – Схема модели Model-View-Template.

В Django реализован мощный движок шаблонов и собственный язык разметки. Шаблоны представляют собой файлы с HTML-кодом, с помощью которого отображаются данные. Содержимое файлов может быть статическим или динамическим. Шаблоны не содержат бизнес-логики. Поэтому они только отображают данные.

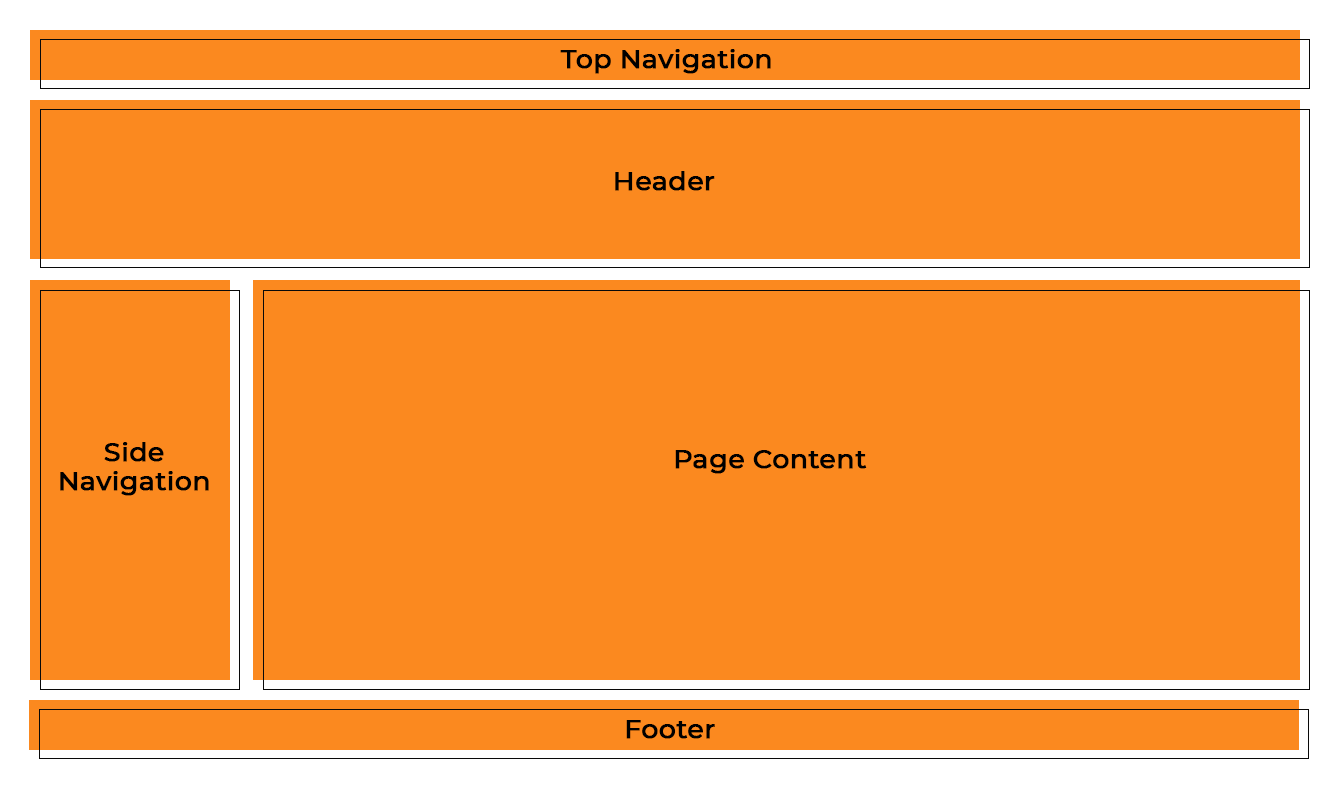


Рисунок 2 – Шаблон страницы

Базовая структура фреймворка включает следующие компоненты:

* URL-маршрутизаторы, перенаправляющие HTTP-запрос от браузера или любого иного веб-клиента в представления;
* представление, которое обрабатывает запрос, обращается к модели и сообщает ей, какие именно данные из БД нужно задействовать, чтобы удовлетворить запрос;
* модель (менеджер базы данных, ORM), «вытаскивающую» нужную информацию из БД и передающую ее представлению;
* HTML-шаблоны, которые используются представлением для демонстрации пользователю полученных от модели данных.

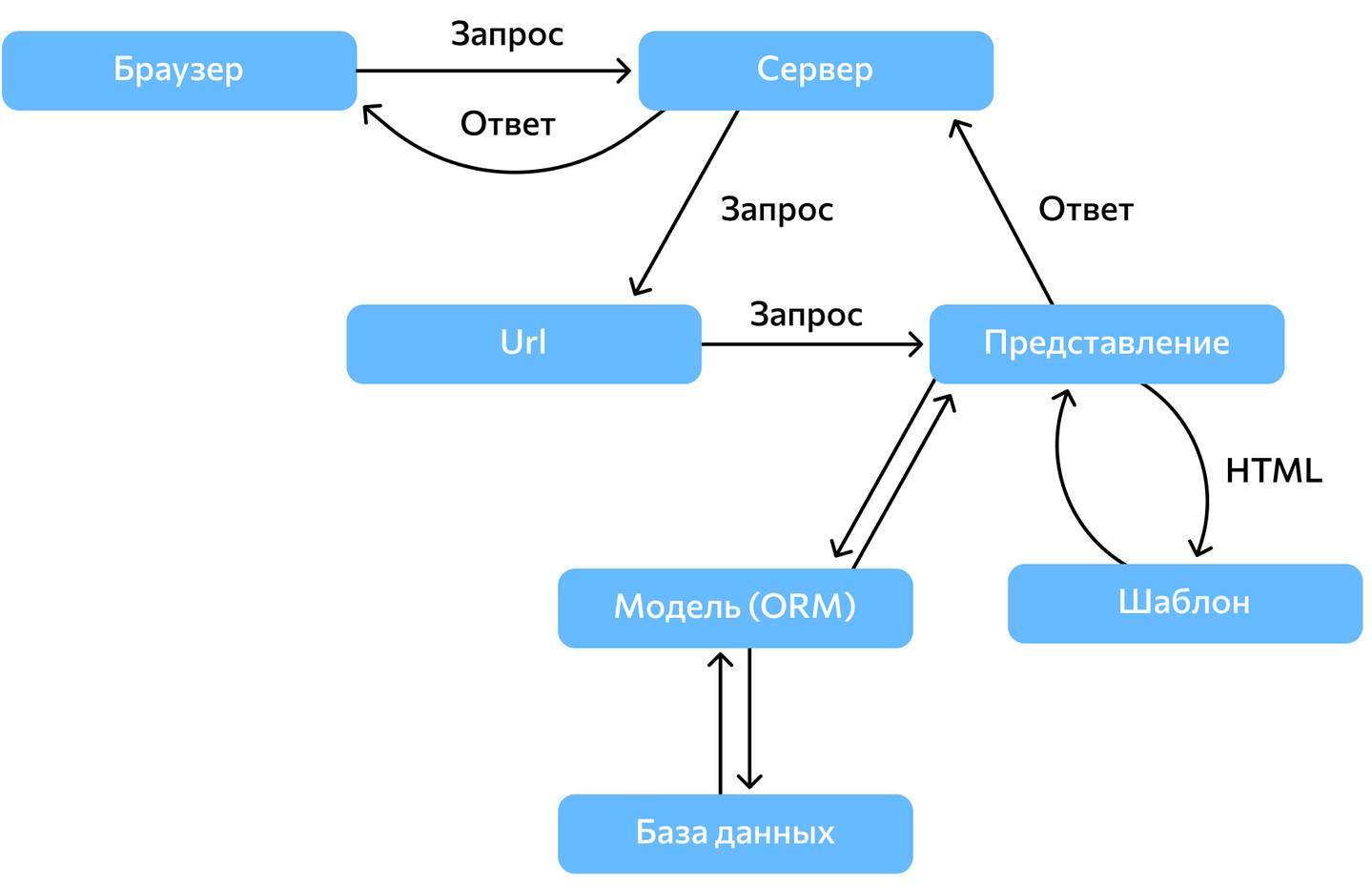


Рисунок 3 - схема работы приложения на Django

Разработка веб-приложения «CreatorCNC» проводилось с использованием фреймворке Django по нескольким причинам:

* Полнота - Фреймворк Django спроектирован по принципу «Все включено». Разработчик может с его помощью создать веб-приложение без сторонних компонентов. Это полезно для начинающих программистов, так как им не приходится отвлекаться на поиск дополнительных инструментов для решения типовых задач.
* Надежность - Django поддерживается и развивается Django Software Foundation, а также сообществом сторонних разработчиков. Фреймворк обновляется и совершенствуется, проверяется на ошибки. Созданные приложения стабильные и содержат меньше багов по сравнению с разработанными на других фреймворках, CRM или написанными с нуля.
* Масштабируемость - Архитектура Django строится на независимости составляющих частей. Любой компонент можно заменить или модифицировать, не затрагивая другие. Возможности Django позволяют, как разрабатывать приложения, так и расширять их при увеличении трафика и нагрузки.
* Безопасность - У Django есть встроенные инструменты защиты от распространенных хакерских атак. Также он позволяет эффективно распределять доступ к данным среди пользователей разного уровня. Это повышает безопасность продукта и стабильность его работы.

# Руководство пользователя.

В начале, для использования веб-приложения в полном объеме нужно авторизоваться (рис.4) или зарегистрироваться (рис. 5), если не существует личного аккаунта.

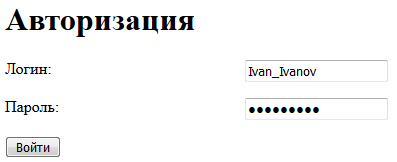


Рисунок 4 – Окно авторизации пользователя.

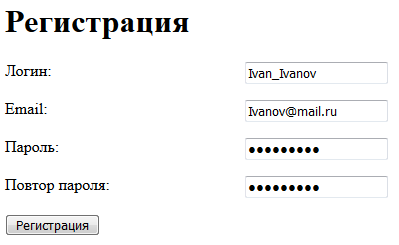


Рисунок 5 – Окно регистрации пользователя.

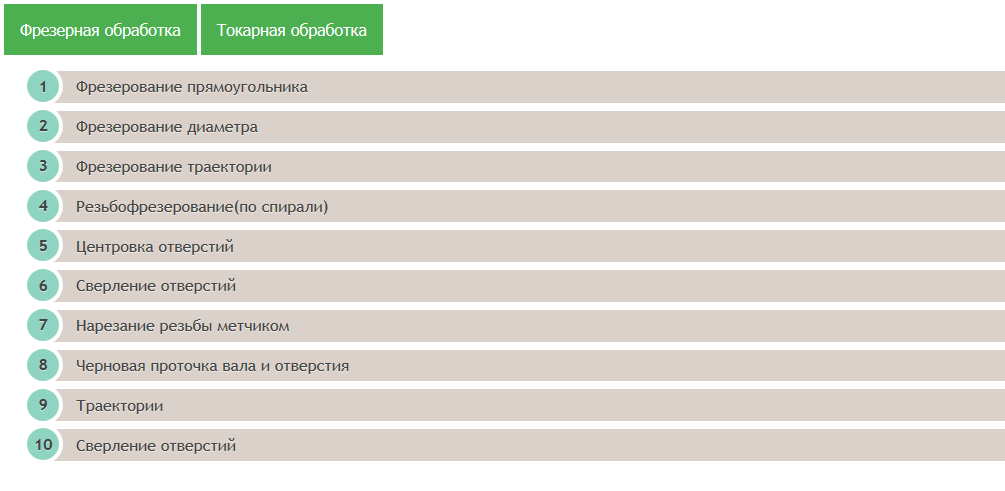
На главной странице (рис. 6) выбрать операцию механообработки.

Рисунок 6 – Главная страница.

Далее согласно схеме (рис. 8) заполнить параметры для фрезеровки прямоугольника (рис.7).

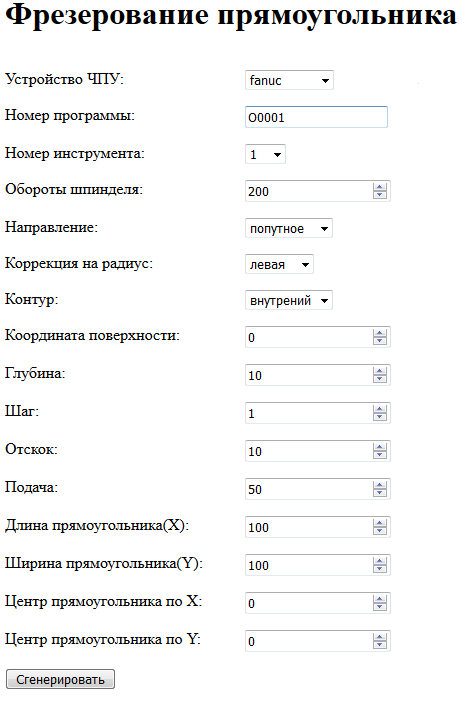


Рисунок 7 – Параметры операции «Фрезерование прямоугольника»

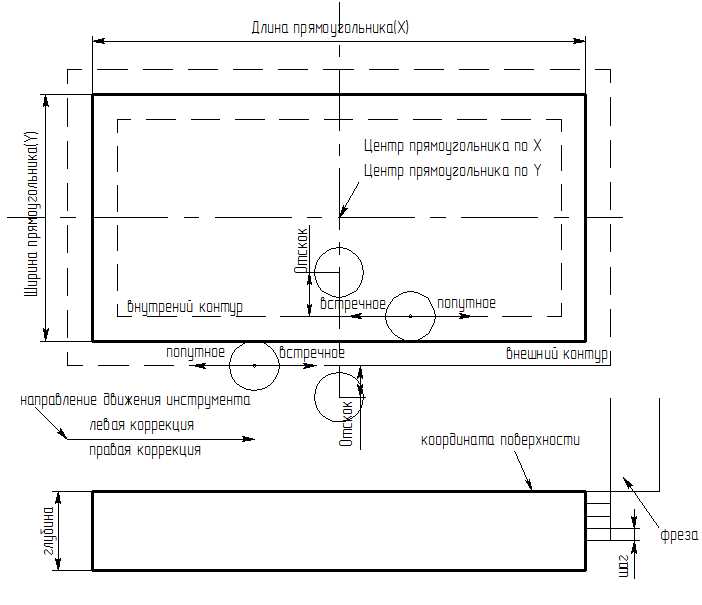


Рисунок 8 – Схема операции «Фрезерование прямоугольника»

# Устройство ЧПУ – комплексная система управления со встроенным промышленным персональным компьютером, предназначенная для синхронной работы всех узлов станка.

**Номер программы** – название или номер управляющее программы, которая отправится на станок ЧПУ. Номер нужно указывать с учетом особенностей станка, например, для стойки Fanuc, обязательно впереди «O» и затем 4 цифры (О0000).

**Номер инструмента** – позиция режущего инструмента в магазине станка ЧПУ.

Обороты шпинделя(n, об/мин) – частота вращения шпинделя, число оборотов режущего инструмента, совершаемое за минуту.

**Направление** – направление движения режущего инструмента. Попутное – фрезерование, при котором направление движения заготовки совпадает с направлением вектора скорости резания. Встречное – фрезерование, при котором направление движения заготовки и вектор скорости резания направлены в разные стороны.

**Коррекция на радиус** – автоматическая коррекция радиуса инструмента, при котором происходит смещение инструмента относительно исходной траектории вправо или влево.

**Контур** – выбор внешнего или внутреннего контура операции.

**Координата поверхности** – это начало обработки по Z относительно системы координат.

**Глубина** – глубина прямоугольника, отсчитываемая от координаты поверхности.

**Шаг** – величина движения инструмента от точки координаты поверхности до глубины.

**Отскок** – величина подхода и отхода режущего инструмента от основного контура. Нужно учитывать диаметр инструмента.

**Подача** – скорость перемещения режущего инструмента за единицу времени.

**Длина прямоугольника (Х)** – длина прямоугольника по оси X в миллиметрах.

**Длина прямоугольника (Y)** – длина прямоугольника по оси Y в миллиметрах.

**Центр прямоугольника по X** – координата по оси «х» относительно центра системы координат.

**Центр прямоугольника по Y** – координата по оси «y» относительно центра системы координат.

Далее пользователю нужно нажать на кнопку «Сгенерировать». И на выходе получается управляющая программа, с представленным кодом, в соответствие с выбранной стойкой ЧПУ.

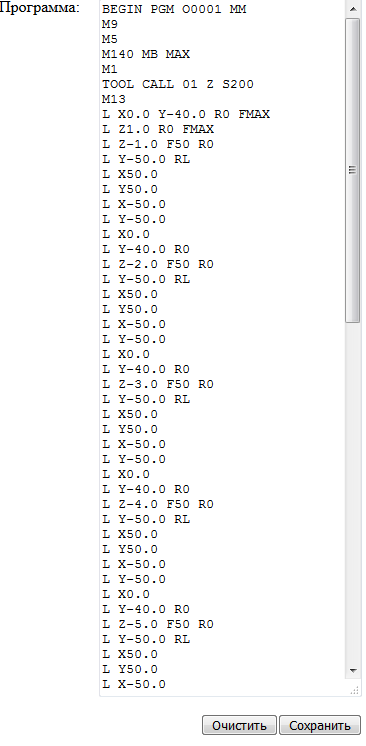
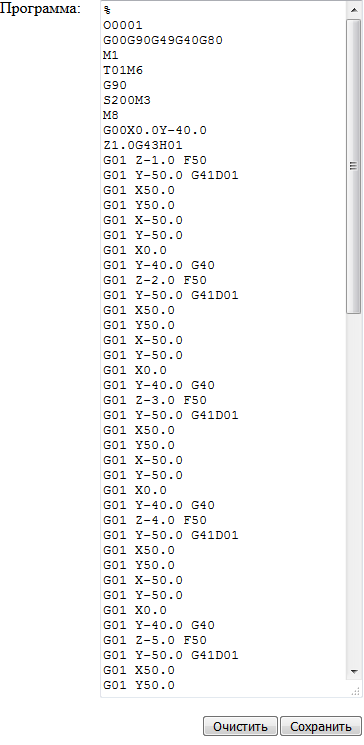


Рисунок 9 – Управляющая программ на стойку Fanuc(слева) и на стойку Heidenhain(справа).

# Затем, можно сохранить управляющую программ, если есть в этом необходимость.

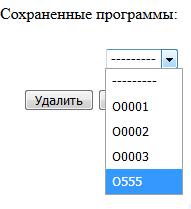


Рисунок 10 – Окно выбора сохраненных управляющих программ.

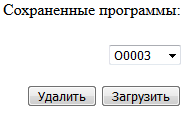


Рисунок 11 – Окно загрузки сохраненных программ.

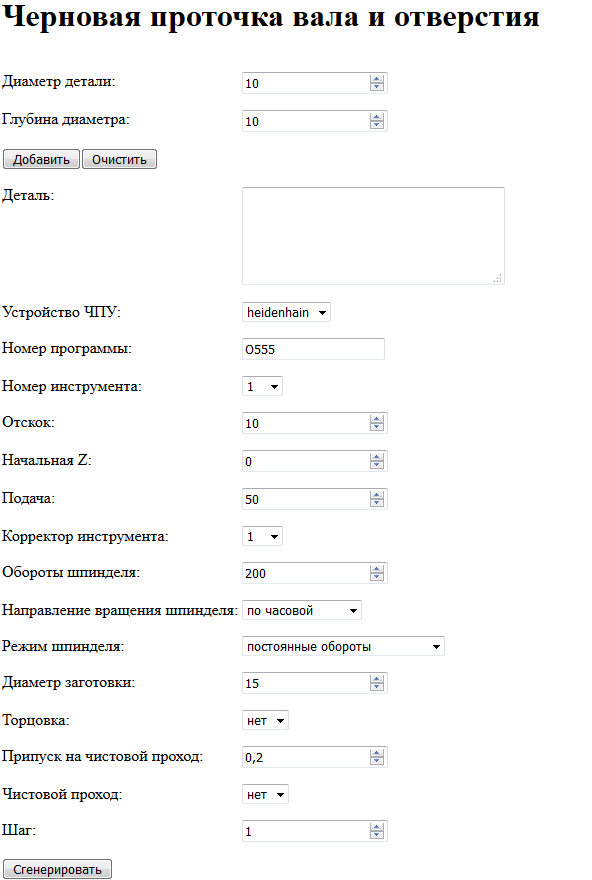


Рисунок 11 - Параметры операции «Фрезерование прямоугольника»

**Диаметр детали** – нужно вписать диаметр детали.

**Глубина диаметра** – нужно вписать глубину диаметра.

**Кнопка «Добавить»** – добавятся параметры в поле деталь на основе параметров «Диаметр детали» и «Глубина диаметра», может быть несколько.

**Начальная Z** – Координата, от которой начинается отсчет операции.

**Направление вращения шпинделя** – Направление движения шпинделя вокруг своей оси.

**Режим шпинделя** – Постоянные обороты - вращение шпинделя независимо от диаметра детали. Постоянная скорость резания – станок автоматически поддерживает обороты, необходимые для того, чтобы резка происходила с желаемой скоростью резания.

**Торцовка** – нужна ли торцовка в начале операции.

**Чистовой проход** – нужен ли чистовой проход в конце операции.

Данное веб–приложение быстро и надежно позволит разработать управляющую программу на станки с ЧПУ, как начинающему инженер-программисту так и более опытному. Легко и быстро позволит, перевести сделанную управляющую программу, по необходимости, с одной стойки ЧПУ на другую.

# Функции системы

1. Создание управляющей программы для фрезеровки прямоугольника.
   1. Получение список строк начала управляющей программы и выбор инструмента.
   2. Получение строк подвода режущего инструмента.
      1. Расчет координат для подвода режущего инструмента
      2. Подстановка в шаблон координат подвода режущего инструмента.
   3. Получение шаблона с подставленным значением коррекции на радиус.
   4. Получение шаблона с фрезеровкой прямоугольника с подставленными значениями.(get\_milling)
   5. Получение список строк конца управляющей программы.
2. Создание управляющей программы для фрезеровки траекторий.
   1. Получение список строк начала управляющей программы и выбор инструмента.
   2. Получение строк подвода режущего инструмента.
      1. Расчет координат для подвода режущего инструмента
      2. Подстановка в шаблон координат подвода режущего инструмента.
   3. Получение шаблона с фрезеровкой траекторий с подставленными значениями.(get\_milling)
   4. Получение список строк конца управляющей программы.

3.Создание управляющей программы для сверления отверстий

3.1. Получение список строк начала управляющей программы и выбор инструмента.

3.2. Подстановка в шаблон координат подвода режущего инструмента.

3.3. Получение шаблона со сверлением отверстий с подставленными значениями(hole\_making).

3.4. Получение список строк конца управляющей программы.

# 4. Создание управляющей программы операции «Черновая проточка вала и отверстия».

4.1. Получение список строк начала управляющей программы и выбор инструмента.

4.2. Подстановка в шаблон координат подвода режущего инструмента.

4.3. Получение списка строк дополнительной операции «торцовка», если необходимо.

4.4. Получение шаблона цикла черновой обработки.

4.5. Получение шаблона контура.

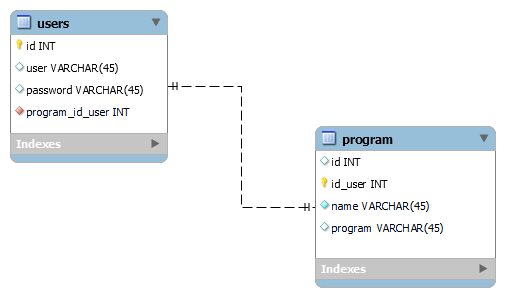
4.6. Получение списка строк с дополнительной операции «чистовой проход».

4.7. Подстановка в шаблон координат отвода режущего инструмента.

4.8. Получение список строк конца управляющей программы.

# 

# Структура данных



# Заключение

Изучил основы языка программирования Python, стандартных и внешних библиотек языка. Разработал веб-приложение «CreatorCNC», которое отвечает за реализацию управляющих программ на станки с устройством ЧПУ.

В процессе разработки веб–приложения возникали некоторые трудности, а именно не мог получить данные из поля формы для подстановки в другое поле этой же формы, с последующей их обработкой. Решена была проблема путем выявлением нажатой кнопки на стадии сервера и подстановкой данных в поле формы.

Проект «CreatorCNC можно расширять и добавлять различные операции, например: электроэрозионной обработки, лазерной обработки, и т.д. Так же можно добавить симулятор движения режущего инструмента на базе полученной управляющей программы.

# Литература

Марк Лутц, «Изучаем Python» том 2- М: Вильямс, 2020. – 700с.

Пол Бэрри, «Изучаем программирование на Python» - М:Эксмо, 2017.-624с.

Антонио Меле, «Django 4 в примерах. Разрабатывайте мощные и надежные веб-приложения на Python с нуля» - М: ДМК-Пресс, 2023. – 800с.

[www.hexlet.ru](http://www.hexlet.ru) – образовательная платформа [электронный ресурс] URL: <https://ru.hexlet.io/blog/posts/pochemu-django-luchshiy-freymvork-dlya-razrabotki-saytov>

www.tiangolo.com – интернет площадка [электронный ресурс] URL: <https://fastapi.tiangolo.com/ru/>

[www.skilfactory.ru](http://www.skilfactory.ru) – образовательная платформа [электронный ресурс] URL: https://blog.skillfactory.ru/glossary/flask/

# Приложение

class SubMoveMill:

"""Основной класс для создания управляюще программы"""

def \_\_init\_\_(self, data):

self.flag = True

self.message = ""

self.number\_tool = data["num\_tool"].zfill(2)

self.name\_program = data["name\_program"]

self.cncdevice = data["cncDevice"]

self.samples = self.connect\_db(self.cncdevice, self.machine)

self.info = Info()

try:

self.spin\_rot = int(data["spindle\_rotation"])

except ValueError:

self.flag = False

self.message = self.info.errors["basic\_msg"]

def supply(self, parametrs: tuple) -> str:

"""

:param parametrs: дполнительные параметры.

:return: Возвращает подвод и включение коррекции на длину инструмента.

"""

return self.samples["supply"].format(\*parametrs)

def choose\_tool(self, parametrs) -> str:

"""Возвращает шаблон с подставленными значениями вызова иструмента"""

return self.samples["choose\_tool"].format(\*parametrs)

def start\_program(self) -> str:

"""Возвращает шаблон с подставленными значениями старта управляющей программы"""

return self.samples["start\_program"].format(self.name\_program)

def end\_program(self) -> str:

"""Возвращает шаблон с подставленными значениями конца управляющей программы"""

return self.samples["end\_program"].format(self.name\_program)

def connect\_db(self, cncdevice: str, machine) -> dict:

"""Подключает к базе данных.

:param cncdevice: Название нужной стойки станка ЧПУ

:return: Возвращает словарь с шаблонами нужной стойки ЧПУ.

"""

with sq.connect("C:\python3.7\django\programcnc\db.sqlite3") as con:

cur = con.cursor()

cur.execute(f"SELECT name, sample FROM {cncdevice}\_{machine}")

result = cur.fetchall()

cur.close()

return {key: value for (key, value) in result}

def show\_message(self):

"""Вывод ошибки"""

print(self.message)

def begin\_program(self) -> List[str]:

"""Возвращает список строк управляющей программы начала программы и выбор инструмента"""

text\_out =[]

text\_out.append(self.start\_program())

text\_out.append(self.choose\_tool((self.number\_tool, self.spin\_rot)))

return text\_out

class BasicMill(sm.SubMoveMill):

"""Базовый класс для фрезерования"""

def \_\_init\_\_(self, data):

self.machine = "milling"

super().\_\_init\_\_(data)

self.direction = data["direction"]

self.rad\_comp = data["radius\_compensation"]

self.contour = data["contour"]

try:

self.start\_height = float(data["start\_height"])

self.depth = abs(float(data["depth"]))

self.rebound = abs(float(data["rebound"]))

self.step = abs(float(data["step"]))

self.feed = abs(int(data["feed"]))

except ValueError:

self.flag = False

self.message = self.info.errors["basic\_msg\_mill"]

def get\_radius\_compensation(self) -> str:

"""Возвращает шаблон с подставлеными значениями - компенсации на радиус интсрумента"""

name = f"radius\_compensation\_{self.rad\_comp}"

return self.samples[name].format(self.number\_tool)

def get\_supply(self, centr\_y: float, size: float) -> float:

"""

:param centr\_y: Точка отвода по у.

:param size: Размер контура.

:return:Возвращает значение у подвода инструмента перед началом обработки"

"""""

if self.contour == "inner":

sup\_y = centr\_y - size / 2 + self.rebound

else:

sup\_y = centr\_y - size / 2 - self.rebound

return sup\_y

def begin\_milling(self, text\_out: List[str], centr\_x: float, centr\_y: float, height: float, size: float):

"""

:param text\_out: Строка текста управляюще программы для вывода.

:param centr\_x: Центр отверстия по х.

:param centr\_y: Центр отверстия по у.

:param height: Величина начала фрезеровки.

:param size: Размер контура.

:return: Возвращает текст управляющей программы с шаблоном, в который добавлены значения

и возвращает величину подвода по y.

"""

sup\_y = self.get\_supply(centr\_y, size)

text\_out.append(self.supply((centr\_x, sup\_y, height + self.step, self.number\_tool)))

return text\_out, sup\_y

def get\_milling(self, text\_out: List[str], name: str, height: float, parametrs: List[float]) -> List[str]:

"""

:param text\_out: Строка текста управляюще программы для вывода.

:param name:Название шаблона.

:param height: Величина начала обработки.

:param parametrs: Дополнительные параметры.

:return: Возвращает список строк управляющей программы,с добавленными шаблоном.

"""

count = height

while count > height - self.depth + self.step:

count -= self.step

text\_out.append(self.samples[name].format(count, \*parametrs))

text\_out.append(self.samples[name].format(height - self.depth, \*parametrs))

return text\_out

class Rectangle(bm.BasicMill):

"""Класс фрезеровки прямоугольника в управляющей прогрмме"""

def \_\_init\_\_(self, data):

super().\_\_init\_\_(data)

self.operation = "rectangle"

try:

self.centr\_x = float(data["centr\_x"])

self.centr\_y = float(data["centr\_y"])

self.long = abs(float(data["long"]))

self.width = abs(float(data["width"]))

except ValueError:

self.flag = False

self.message = self.info.errors["rectangle\_msg"]

def create(self) -> str:

"""Возвращает текст управляющей программы"""

text\_out = self.begin\_program()

text\_out, sup\_y = self.begin\_milling(text\_out, self.centr\_x, self.centr\_y, self.start\_height, self.width)

name = f"{self.operation}\_{self.contour}\_{self.direction}\_bottom"

g\_rad\_comp = self.get\_radius\_compensation()

coord\_y\_down = self.centr\_y - self.width/2

coord\_y\_up = self.centr\_y + self.width/2

coord\_x\_left = self.centr\_x - self.long/2

coord\_x\_right = self.centr\_x + self.long/2

parametrs = [self.feed,

coord\_y\_down,

coord\_y\_up,

coord\_x\_left,

coord\_x\_right,

self.centr\_x,

sup\_y,

g\_rad\_comp]

text\_out = self.get\_milling(text\_out, name, self.start\_height, parametrs)

text\_out.append(self.end\_program())

return "".join(text\_out)