

Введение

Веб-приложение для идентификации динамических систем является клиент-серверным инструментальным программным обеспечением с использованием базы данных. Приложение обеспечивает пользователя интерактивным, доступным и дружественным интерфейсом для выполнения различных задач в области идентификации. Приложение позволяет оценивать параметры системы модели на основе входных и выходных данных и анализировать поведение системы.

*Цели данной работы:* облегчение задач идентификации систем, предоставление независимого и доступного программного решения в виде веб-приложения.

*Актуальность бакалаврской работы* заключается в том, что веб-приложение для идентификации систем разрабатывается с концепцией импортозамещения, предоставляя самодостаточное и надежное решение для задач идентификации систем. Существующие решения на рынке, а именно: LabVIEW, MATLAB Simulink, dSPACE SystemDesk, MapleSim находятся под юрисдикцией недружественных стран. В связи с этим возникает острая необходимость в разработке собственных решений, особенно в сфере наукоёмких технологий. Такой подход способствует технологической независимости, обеспечивает доступность инструментов идентификации систем, адаптированных к конкретным потребностям местного сообщества студентов, преподавателей и специалистов в области идентификации систем.

В первом разделе приведены общие сведения о науке идентификации систем. В этом разделе определяются основные термины, такие как: система, динамическая система, идентификация. Рассматриваются типы моделей, описывающие системы. Также уделяется внимание о способах построения модели и методам идентификации.

Во втором разделе описывается процесс разработки веб-приложения. Производится экскурс в терминологическую базу веб-разработки. Кратко обозреваются основные технологии: HTTP, веб-сервер, паттерны проектирования, интерфейс WSGI, фреймворк Flask и непосредственно реализацию приложения.

Стоит отметить достоинства веб-приложения. Доступность – к приложению доступ возможен с любого устройства, подключенного к глобальной сети Интернет, этот аспект устраняет необходимость в установке ПО, а также избавляет пользователя от особых требований к его оборудованию. Возможность создания профиля – это позволяет сохранять идентифицированные модели систем в профиле пользователя.

Данное приложение разработано с открытым исходным кодом, это значит, что исходный код находится в свободном для проверки и модификаций доступе. Таком образом, соблюдение принципов открытого исходного кода в разработке веб-приложения способствует развитию культуры обмена знаниями и даёт пользователям продукта возможность использовать опыт сообщества инженеров-программистов в области идентификации систем.

# Об идентификации динамических систем

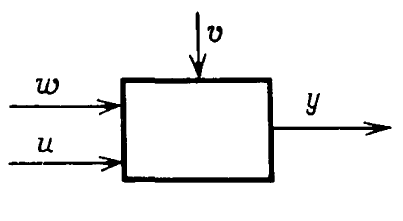
## Общие сведения

Наука идентификации исследует методы, которые позволяют формировать модели динамических систем на основе результатов наблюдений.

Система – это объект, в котором происходит взаимодействие между разнотипными переменными и формируются наблюдаемые сигналы (Льюинг)

Наблюдения представляют из себя входные и выходные данные. Входные данные – это любые управляющие воздействия, в том числе, шумы, представленные в формате временных рядов. Выходные данные – те данные, которые можно зафиксировать, например, приборами.

Динамические системы – такие системы, в которых текущее значение выходного сигнала зависит не только от текущих, но и более ранних значений внешних воздействий.



*y* – выходной сигнал; – неизмеримая помеха; – измеримая помеха;

*u –* входной сигнал управления

Рисунок 1 – Система в общем виде

* 1. **Основные типы моделей и их использование**

В задачах исследования динамических систем необходимо соотнести между собой переменные, которые характеризуют систему. Назовём совокупность предполагаемых связей между наблюдаемыми сигналами **моделью**. Обзорно рассмотрим типы моделей:

1. Умозрительные (субъективные). Наиболее часто применяются в бытовой жизни, в этом типе моделей нет математики. Например, для увеличения скорости езды при управлении велосипедом достаточно знать, что при увеличении частоты и силы вращения педалей, увеличится скорость движения;
2. Графические модели. Такие модели описывают свойства некоторых систем в виде графиков, диаграмм и таблиц. Например, графики статических характеристик могут описать некоторые нелинейные звенья: реле, клапаны. Линейные системы могут быть представлены импульсными реакциями, реакциями на ступенчатое воздействие или частотными характеристиками;
3. Аналитические (или математические) модели. В этих моделях соотношения, описывающие связи между системными переменными, задаются в виде разностных и дифференциальных уравнений. В зависимости от типа используемых разностных или дифференциальных уравнений такие модели, в свою очередь, могут быть непрерывными или дискретными по времени, сосредоточенные или распределенные, детерминированные или стохастические, линейные или нелинейные и т. д.;
4. Программные модели. Программа для ЭВМ также может быть моделью, т. к. она описывает поведение сложных систем на языке программирования. Это описание может быть представлено в виде совокупности взаимодействующих между собой подпрограмм.
   1. **Построение моделей**

Построение моделей основывается на данных наблюдений. Так, например, субъективная модель динамики управления движением велосипеда основана на личном опыте велосипедиста. Графические модели наполняются результатами некоторых измерений.

Для построения математических моделей используют один из двух способов. Первый способ состоит в том, чтобы провести декомпозицию системы на такие подсистемы, свойства которых очевидны из ранее накопленного опыта, например, из известных законов физики или других надежных и доказанных соотношений. Далее необходимо формально на математическом языке объединить эти подсистемы. Полученное описание становится моделью всей системы.

Суть второго способа построения как математических, так и графических моделей заключается в использовании экспериментальных данных. В этом случае ведется фиксация входных и выходных сигналов системы, и модель формируется в результате обработки соответствующих данных. Этот способ называется – **идентификация**.

* 1. **Основные компоненты процедуры идентификации систем**

Введем основные компоненты, которые возникают при проектировании модели системы по данным наблюдения:

1. Данные наблюдений. Входные и выходные данные регистрируются в процессе проведения целенаправленных идентификационных экспериментов, когда пользователь может задать перечень и моменты изменения сигналов.

2. Множество моделей. Множество моделей-кандидатов формируется в результате применения методов моделирования. В этом множестве выполняется поиск наиболее подходящей модели. На этом этапе знание формальных свойств моделей необходимо соединить с априорным знанием, инженерным искусством и интуицией. Набор моделей, у которых параметры рассматриваются прежде всего, как варьируемые средства подстройки моделей к имеющимся данным и не отражают физики процесса, называется *черным ящиком*. Множества моделей с настраиваемыми параметрами, допускающими физическую интерпретацию, называют *серыми ящиками*.

3. Правило оценки степени соответствия испытываемой модели данным наблюдений. Оценка качества модели связана, как правило, с изучением поведения моделей в процессе их использования для воссоздания данных наблюдений. На этом этапе производится выбор из множества моделей-кандидатов наилучшей модели.

# Веб-приложение

## Общие сведения

Веб-приложение – это разновидность программного обеспечения, доступ к которому обеспечивается средствами веб-браузера. Веб-браузер – это программа, которая отправляет запросы, получает и обрабатывает ответы от имени пользователя на его устройстве. Обязательным условием функционирования веб-приложения является наличие стабильного соединения с глобальной сетью Интернет. Одно из главных отличительных особенностей такого вида приложений является кроссплатформенность и независимость от аппаратной конфигурации устройства, у которого есть доступ в сеть.

В этом разделе изложены основные идеи и концепции построения веб-приложения, начиная от описания клиент-серверной архитектуры и краткого обзора протокола HTTP, заканчивая описанием реализации обработчиков запросов фреймворка Flask. При помощи этого фреймворка реализуется веб-приложение для идентификации динамических систем.

## Описание архитектуры веб-приложения

Архитектура программного обеспечения (ПО) – это организация и структура системы, определяющая ее компоненты, их взаимодействие и принципы организации данных. Она определяет основные аспекты системы, такие как разделение ответственности, модульность, коммуникацию и распределение функций.

Глобально любое веб-приложение основано на клиент-серверной архитектуре. В такой архитектуре есть устройство, которое обслуживает запросы, и есть устройство, которые эти запросы направляет. Сервер – это устройство, на котором установлены программы, которые обслуживают запросы клиентов. Клиент – это программа пользователя, которая удовлетворяет его потребностям.

На этапе инициации клиент устанавливает соединение с сервером с помощью протокола транспортного уровня TCP. Клиент может использовать одно и то же соединение для работы с сервером или инициировать новое соединение каждый раз. Это зависит от задачи, конфигурации сети и конкретных настроек оборудования. После установки соединения клиент посылает HTTP-сообщение с телом и параметрами запроса. Сервер принимает это сообщение и на основании логики работы формирует и отправляет HTTP-сообщение ответа. Таким образом, организовывается сеанс сессии. Остановимся подробнее на протоколе HTTP.

## Краткий обзор HTTP протокола

### HTTP

В основе клиент-серверного взаимодействия лежит протокол прикладного уровня модели OSI: протокол HTTP (HyperText Transfer Protocol). HTTP был разработан как протокол обмена данными между веб-сервером и веб-браузером. Который используется для передачи между клиентом и сервером (хранителем) файлов HTML, CSS, JS, изображений, аудио, видео, введённых пользователем данных и прочего. Клиент (веб-браузер) отправляет серверу (веб-серверу) запросы и получает от него ответы. Сервер в рамках взаимодействия всегда «ожидает» инициации от клиента.

Одна из ключевых особенностей HTTP заключается в его безсостоятельности (statelessness), что означает, что сервер не сохраняет информацию о предыдущих запросах от клиента. Каждый запрос рассматривается отдельно, без памяти о предыдущем состоянии. Вместо этого, клиент может передавать информацию в запросах, используя заголовки и тело запроса.

Протокол HTTP определен методами запроса, которые указывают тип операции, выполняемой на ресурсе. Некоторые из наиболее распространенных методов включают GET (запрос на получение ресурса), POST (отправка данных на сервер для обработки), PUT (замена или обновление ресурса) и DELETE (удаление ресурса). Вместе с методом запроса указывается URL (Uniform Resource Locator – унифицированный указатель ресурса) или URI (Uniform Resource Identifier)

HTTP также определяет коды состояния, которые указывают результат выполнения запроса. Например, код состояния 200 означает успешный запрос, а код 404 - что запрошенный ресурс не найден. Коды состояния помогают клиенту и серверу взаимодействовать и корректно обрабатывать результаты запросов.

Кроме методов и кодов состояния, HTTP также определяет заголовки, которые предоставляют метаданные о запросе или ответе. Заголовки могут содержать информацию о типе содержимого, длине запроса или ответа, аутентификации и другие дополнительные параметры, необходимые для обработки запроса.

HTTP протокол стал основой для развития веб-технологий и позволил создать мощные веб-приложения и сервисы. Например, с использованием HTTP можно передавать данные в различных форматах, таких как HTML, JSON или XML. Это позволяет клиентским приложениям обмениваться данными с сервером и отображать их в пользовательском интерфейсе.

В целом, HTTP является фундаментальным протоколом, обеспечивающим передачу данных в Интернете. Он обеспечивает структурированное клиент-серверное взаимодействие и предоставляет механизмы для передачи запросов, получения ответов и обработки различных кодов состояния. Благодаря своей простоте и универсальности, HTTP стал неотъемлемой частью современного веб-разработки и обеспечивает передачу данных между клиентами и серверами с высокой эффективностью и надежностью. [[официальная спецификация](https://datatracker.ietf.org/doc/html/rfc9112)]

### HTTPS

HTTPS (Hypertext Transfer Protocol Secure) - это защищенная версия протокола HTTP, обеспечивающая безопасную передачу данных между клиентами и серверами в сети Интернет. Он предоставляет шифрование и аутентификацию, чтобы обеспечить конфиденциальность и целостность данных, передаваемых между клиентом и сервером.

Основной механизм безопасности HTTPS - это использование SSL (Secure Sockets Layer) или его последующего преемника, TLS (Transport Layer Security). SSL/TLS используют асимметричное и симметричное шифрование для защиты данных. При установлении соединения между клиентом и сервером происходит, так называемое, "рукопожатие", где клиент и сервер обмениваются сертификатами и устанавливают общий секретный ключ для шифрования и расшифровки данных.

Протокол HTTPS также использует порт 443, в отличие от порта 80, который используется протоколом HTTP. При использовании HTTPS веб-сайты имеют URL, начинающийся с "https://" вместо "http://".

Одним из ключевых преимуществ HTTPS является обеспечение конфиденциальности пользовательских данных. Все данные, передаваемые между клиентом и сервером, включая логины, пароли, кредитные карты и другую чувствительную информацию, шифруются, что делает их невозможными для прочтения злоумышленниками. Это особенно важно при передаче данных через общедоступные или ненадежные сети, такие как общественные Wi-Fi сети.

HTTPS также обеспечивает аутентификацию (подтверждение подлинности) сервера. Клиент может быть уверен, что он связывается с правильным сервером, так как сервер предоставляет свой цифровой сертификат, выданный надежным удостоверяющим центром. Это защищает от атак "Man-in-the-Middle", при которых злоумышленник пытается подменить сервер и перехватить данные между клиентом и сервером.

Использование HTTPS становится все более широко распространенным, особенно для веб-приложений, где требуется защита личных данных пользователей. Браузеры и поисковые системы активно поощряют использование HTTPS, и многие сайты уже перешли на этот протокол. Это способствует повышению безопасности и доверия пользователей, обеспечивая защиту их конфиденциальной информации при взаимодействии с веб-сайтами. [[1](https://help.reg.ru/support/ssl-sertifikaty/obshchaya-informatsiya-po-ssl/chto-takoye-protokol-https-i-printsipy-yego-raboty)]

## Веб-сервер

Веб-сервер представляет собой программное обеспечение, предназначенное для обслуживания запросов от клиентских устройств и предоставления веб-ресурсов посредством протокола HTTP (Hypertext Transfer Protocol). Веб-серверы играют важную роль в современной информационной инфраструктуре, обеспечивая доступ к веб-страницам, приложениям и другим ресурсам через глобальную сеть Интернет.

Основная функция веб-сервера заключается в обработке и отклике на HTTP-запросы от клиентов. Когда клиент отправляет запрос на получение определенного ресурса, веб-сервер принимает этот запрос и определяет, какой ресурс был запрошен. Затем сервер обращается к хранилищу данных, где хранятся запрошенные ресурсы, и возвращает клиенту соответствующий HTTP-ответ, содержащий запрошенную информацию или указывающий на ошибку.

Веб-серверы могут также выполнять другие функции, такие как аутентификация и авторизация пользователей, управление сеансами, обработка форм, кеширование ресурсов для повышения производительности, шифрование данных с использованием протокола SSL/TLS для обеспечения безопасной передачи информации.

Одним из распространенных вариантов использования веб-сервера является его комбинация с прокси-сервером. Прокси-сервер действует как посредник между клиентом и сервером, перенаправляя запросы от клиента к соответствующему серверу и возвращая ответы обратно клиенту. Прокси-серверы могут выполнять различные функции, включая кэширование, балансировку нагрузки, фильтрацию содержимого, защиту от DDoS-атак (distributed denial-of-service attack) – распределенная атака с целью довести сервер до состояния отказа в обслуживании.

Основная цель веб-сервера заключается в обеспечении доступа к веб-ресурсам и предоставлении эффективного и безопасного взаимодействия между клиентами и серверами. Он играет важную роль в функционировании веб-сайтов, приложений и других онлайн-сервисов, предоставляя пользователю необходимые данные и обеспечивая плавное взаимодействие в сети Интернет.[[1](https://doka.guide/tools/web-server/), [2](https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_web_server), [3](https://lectureswww.readthedocs.io/5.web.server/web.server.html)]

*Источники о прокси-серверах:* [Ист. 1](https://www.linuxbabe.com/it-knowledge/differences-between-forward-proxy-and-reverse-proxy) (en)[Ист. 2](https://trends.rbc.ru/trends/industry/61c192a19a79475f186a0296)

## Виды веб-приложений

Все веб-приложения разрабатываются с помощью одних и тех же технологий: HTML, CSS и JavaScript. Однако есть много вариаций организации работы приложения. Выбор способа зависит от цели приложения. Основных подхода всего два: многостраничные приложения и Single Page Applications (одностраничные приложения), но каждый из них делится на подвиды.

### Многостраничные приложения

Многостраничные приложения состоят в основном из статичных страниц. В таких страницах нет изменяемого контента. Их просто разрабатывать, но если страниц становится много (сотни или тысячи), или данные на странице меняются, то придётся генерировать их в режиме реального времени. Для этого нужно затрачивать программные и аппаратные ресурсы: подключать серверные мощности и писать дополнительный код и на каждый переход от одной страницы к другой нужно генерировать и загружать новую страницу, а это занимает время.

Рассмотрим подвиды многостраничных приложений: приложение с набором готовых страниц (файлы расширения .HTML), и приложение, с множеством страниц, которые генерируются на сервере.

*Совокупность готовых статичных страниц*, которые связаны между собой ссылками друг на друга, также имеют право называться веб-приложением, но точнее их будет назвать сайтом. Как правило, такие сайты содержат в себе данные, которые не меняются или слабо меняются с течением времени: это может быть персональный блог или страница с индивидуальными достижениями. Для организации доступа к таким страницам используются веб-сервера, которые по запросу пользователя вернут ему нужный готовый ресурс.

*В случае приложения с динамической генерацией HTML* файлов есть возможность размещать на страницах нестатичные данные, то есть такие, которые меняются с течением времени в зависимости от действий пользователя или в зависимости от внешних изменений. В данной работе используется именно такой подход с помощью шаблонизатора Jinja2.

Jinja2 – это библиотека, написанная на языке программирования Python, которая используется для генерации динамических веб-страниц и других текстовых файлов, позволяя разработчикам разделять логику приложения и представление данных.

Основная цель Jinja2 состоит в том, чтобы упростить процесс создания и управления шаблонами. Шаблоны в Jinja2 являются текстовыми файлами, в которых могут содержаться HTML-разметка и специальные конструкции, называемые выражениями Jinja2. Выражения Jinja2 позволяют вставлять переменные, логические операции, циклы и другие управляющие конструкции внутрь шаблонов.

Jinja2 поддерживает множество полезных функций, таких как условные операторы (if-else), циклы (for), фильтры данных, наследование шаблонов и многое другое. Эти возможности позволяют разработчикам создавать динамические страницы, адаптирующиеся к различным данным и сценариям использования.

Важной особенностью Jinja2 является его отделение логики от представления. Это означает, что разработчики могут создавать шаблоны, содержащие минимальное количество логики и фокусирующиеся на отображении данных. Такой подход повышает читаемость и поддержку кода, упрощает внесение изменений в представления и улучшает повторное использование кода.

Jinja2 также обладает мощными инструментами для обработки ошибок и исключений. Он предоставляет механизмы для обработки ошибок в шаблонах, таких как отсутствие переменных или некорректные выражения, и позволяет разработчикам определить собственные стратегии обработки ошибок.

Использование Jinja2 в проектах Python обеспечивает гибкость и удобство при создании веб-приложений. Он интегрируется с различными фреймворками, такими как Flask и Django, и может быть использован для разработки различных типов проектов - от простых веб-страниц до сложных и масштабируемых приложений.

### Одностраничные приложения

Одностраничные приложения (SPA – Single Page Application) дают возможность разрабатывать клиентские приложения со сложной логикой с помощью JavaScript. В этом подходе отрисовкой содержимого на странице управляет JavaScript. При помощи этого языка программирования можно написать функции, которые будут посылать запросы веб-серверу на получение данных, затем они будут размещены на странице в соответствующих местах. Переходы между экранами будут мгновенными, и пользователь сразу увидит результат своих действий. Однако такой подход создаёт новые проблемы, а именно: большая нагрузка на браузер, вследствие этого падение производительности.

Разработка таких приложений часто сложнее, поэтому в данной выпускной работе вид веб-приложения – многостраничный, но с размещением на них нестатичных элементов, например, интерактивный график переходного процесса. Этот выбор себя оправдывает, так как само приложение не нагружено большим количеством страниц.

## Архитектура и паттерны проектирования веб-приложений

**Архитектура приложения** — это набор решений о том, как модули приложения будут связаны друг с другом и с внешними компонентами. **Паттерны проектирования** – шаблонное решение частной архитектурной проблемы. Важно спроектировать веб-приложения согласно определенном паттерну, исходя из архитектурного решения. Рассмотрим паттерн проектирования MVC, который используется в веб-приложении для идентификации динамических систем. [[1](https://doka.guide/tools/architecture-and-design-patterns/), [2](https://doka.guide/tools/architecture-mvc/)]

### Паттерн проектирования MVC

MVC (Model-View-Controller) — структура приложения, в которой за данные, их обработку и их вывод отвечают три разных сущности:

* Модель (model) отвечает за данные и их структуру, cодержит в себе схему данных приложения;
* Представление (view) отвечает за отображение данных, показывает данные в понятном для пользователя виде;
* Контроллер (controller) или обработчик – отвечает за обработку данных.

Таким образом разделяется ответственность между разными программными модулями. Важно упомянуть о потоке данных между этими сущностями (см. рис. 1):

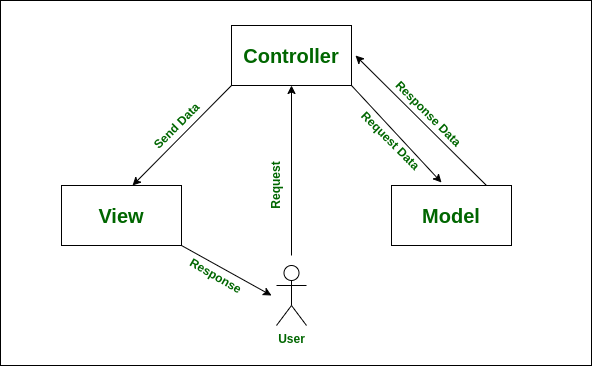
* от пользователя передаются представлению;
* от представления — контроллеру;
* через контроллер обновляется модель;
* модель уведомляет представление о том, что что-то изменилось.

Рисунок 1 – Паттерн проектирования MVC

### Архитектурный стиль RESTful API

Это архитектурный стиль, который используется для создания веб-сервисов. Возможно, так и должно быть. Но стиль и инструмент – это разное. Инструментом пользуются для создания. А стиль – это на что должен быть похож результат. Как минимум. Он опирается на принципы REST (Representational State Transfer) и включает в себя использование HTTP-методов (GET, POST, PUT, DELETE) для управления ресурсами.

*Информация в этом подразделе будет дописываться.*

## Технологический стэк

Фреймворк – набор готовых программных пакетов, программных модулей для облегчения труда программистов и экономии времени разработки программного обеспечения. Фреймворк конкретизирует архитектурное решение и позволяет реализовать выбранный паттерн проектирования. Рассмотрим один из фреймворков.

*Информация в этом подразделе будет дописываться.*

### WSGI

***Источники про WSGI:***

[Основной источник](https://peps.python.org/pep-0333/); [Дополнительный источник 1;](https://habr.com/ru/articles/426957/) [Доп. источник 2;](http://citforum.ru/programming/python/wsgi/) [Доп. источник 3;](https://lectureswww.readthedocs.io/5.web.server/wsgi.html)

***Сценарий:***

*Можно расписать кучу страниц, что такое WSGI, объяснить его по источнику 3 и источнику из PEP 333, затем сказать, что Flask всё это реализует и перейти туды*

***Своими словами:***

*WSGI – это стандарт, по которому взаимодействуют программы, написанные на языке программирования Python c веб-сервером. Был создан для унификации доступа к веб-серверам. Решает проблему совместимости библиотек и фреймворков с веб-серверами.*

WSGI (Web Server Gateway Interface) – промежуточное программное обеспечение.

Пару слов о WSGI надо рассказать.

### Flask

Flask – это минималистичный фреймворк для создания веб-приложений на языке Python. Он предоставляет широкий спектр возможностей и инструментов для быстрой и качественной разработки отзывчивых приложений.

Вот некоторые ключевые особенности данного фреймворка: Здесь и выше: наверное, отступы можно уменьшить. Выполню в качестве примера:

1. Легкость использования: Flask имеет простую и интуитивно понятную структуру, которая делает его легким в освоении и использовании. Он предоставляет только базовый функционал, позволяя разработчикам гибко выбирать сторонние инструменты и библиотеки для своих проектов. Или какой-то промежуточный вариант. Но… тут решение принимаете Вы
2. Маршрутизация URL: Flask предлагает простой способ определения URL-маршрутов и связывания их с соответствующими функциями-обработчиками. Это позволяет определить, как приложение должно отвечать на запросы по определенным URL-адресам.
3. Встроенный сервер разработки: Flask поставляется с встроенным сервером разработки, который упрощает запуск и тестирование веб-приложений без необходимости настройки отдельного веб-сервера.
4. Расширяемость: Flask имеет модульную архитектуру, позволяющую разработчикам добавлять функциональные возможности с помощью расширений (extensions). Существуют множество расширений для Flask, которые обеспечивают поддержку баз данных, авторизацию, формы, обработку файлов, асинхронные запросы и многое другое.
5. Шаблонизация: Flask предоставляет инструменты для использования шаблонов, которые упрощают создание динамических веб-страниц. Шаблонизация позволяет разделить логику и представление веб-приложения, что делает код более удобочитаемым и обеспечивает повторное использование кода. Такой движок шаблонизации называется Jinja2.
6. Поддержка HTTP-методов: Flask обрабатывает различные HTTP-методы, такие как GET, POST, PUT, DELETE и другие, что позволяет легко определять обработчики для разных типов запросов.
7. Отладочные возможности: Flask предлагает отладочные инструменты, которые помогают разработчикам в процессе отладки приложений. Это включает в себя режим разработки, отображение отладочной информации при ошибке и возможность интерактивной отладки.
8. Flask является очень популярным фреймворком для разработки веб-приложений на Python благодаря своей простоте и гибкости. Он хорошо подходит для небольших и средних проектов, где требуется быстрый старт и минимальная настройка.

*Информация в этом подразделе будет дописываться*

**Обоснование выбора технологического стека!!!**

## Реализация

### Основные преимущества

Основные функции включают в себя: удобный интерфейс пользователя – веб-приложение обладает интуитивно понятным интерфейсом, который упрощает процесс идентификации системы; легкий импорт данных – пользователи могут легко внести данные входных и выходных сигналов, как правило, в виде временных рядов; оценка модели системы – приложение содержит алгоритмы и методы для оценки параметров модели, например, с помощью метода наименьших квадратов, метод максимального правдоподобия или с помощью метода идентификации подпространства; экспорт модели в виде коэффициентов полинома числителя и знаменателя передаточной функции, вывод передаточной функции пользователю; оценка качества идентификации – приложение позволяет оценивать достоверность моделей, отображая основной критерий качества: функционал. Вдобавок, возможно провести валидацию модели с помощью методов остаточного анализа и оценки соответствия модели; вывод графиков, а именно: кривые входных и выходных сигналов, кривая переходного процесса, графики частотных характеристик.