ПМ 1. Установка и сопровождение программного и аппаратного обеспечения.

Раздел 1. "Основные понятия и виды программного обеспечения"

Тема 1.1: Введение в программное обеспечение
Определение программного обеспечения
Роль программного обеспечения в современных системах.
Основные виды программного обеспечения.
Жизненный цикл программного обеспечения.
Тенденции развития программного обеспечения.

Определение программного обеспечения

Программное обеспечение (ПО) - это совокупность программ, данных и документации, которые обеспечивают функционирование компьютерной системы или приложения.

Программное обеспечение - это набор инструкций, написанных на языке программирования, которые выполняют определенные задачи на компьютере или другом устройстве. Оно может включать операционные системы, прикладные программы, драйверы устройств и другие компоненты.

Программное обеспечение - это комплект программ, разработанных для решения конкретных задач или предоставления определенных услуг. Это может быть программное обеспечение для управления базами данных, обработки текстов, обмена сообщениями, визуального проектирования и многого другого.

Определение программного обеспечения

Программное обеспечение - это абстрактное понятие, описывающее программы и данные, необходимые для работы компьютерной системы или приложения. Оно включает в себя операционные системы, библиотеки, компиляторы, среды разработки и другие инструменты, которые помогают создавать, запускать и управлять программами.

Программное обеспечение - это совокупность программ и данных, которые обеспечивают функциональность и возможности компьютерной системы или приложения. Оно может быть разработано коммерческими компаниями, сообществами разработчиков или индивидуальными программистами и может быть распространено в виде коммерческих продуктов, свободного программного обеспечения или с открытым исходным кодом.

Роль программного обеспечения в современных системах.

Программное обеспечение играет критическую роль в современных системах и имеет огромное значение во многих областях. Вот некоторые из основных ролей программного обеспечения:

1. Управление и управление ресурсами: Программное обеспечение, особенно операционные системы, позволяет управлять и координировать ресурсы компьютерной системы, такие как процессор, память, дисковое пространство и периферийные устройства. Оно обеспечивает эффективное распределение ресурсов между различными программами и обеспечивает выполнение задач в системе.

2. Разработка и выполнение приложений: Программное обеспечение предоставляет инструменты и среды разработки для создания новых приложений. Разработчики используют программные библиотеки, фреймворки и инструменты для написания кода, отладки программ, тестирования и сборки приложений. Затем программное обеспечение выполняет роль исполнителя, обеспечивая выполнение приложений на компьютере или устройстве конечного пользователя.

Роль программного обеспечения в современных системах.

3.Предоставление функциональности и сервисов: Программное обеспечение предлагает широкий спектр функциональности и сервисов для пользователей. От офисных приложений, таких как текстовые редакторы и электронные таблицы, до приложений для обработки изображений и видео, программное обеспечение обеспечивает возможности для создания, редактирования, обмена и управления различными типами данных.

4.Автоматизация и оптимизация бизнес-процессов: Программное обеспечение играет важную роль в автоматизации и оптимизации бизнес-процессов в различных отраслях. Это может быть программное обеспечение для управления заказами, учета и финансов, управления цепочкой поставок или управления взаимоотношениями с клиентами. Оно помогает компаниям повысить эффективность, сократить издержки и улучшить качество своих операций.

5.Улучшение коммуникации и связи: Программное обеспечение играет важную роль в обеспечении коммуникации и связи между людьми. Это может быть в виде электронной почты, мгновенных сообщений, видеоконференций и социальных сетей. Программное обеспечение для связи облегчает обмен информацией и сотрудничество, улучшая коммуникацию на личном и профессиональном уровне.

Основные виды программного обеспечения.

Программное обеспечение можно разделить на две основные категории: системное и прикладное.

1.Системное программное обеспечение:

1. **Операционные системы**: Это программы, которые управляют аппаратными ресурсами компьютера и предоставляют интерфейс для взаимодействия с пользовательскими приложениями. Примеры: Windows, macOS, Linux.
2. **Драйверы**: Они предназначены для обеспечения взаимодействия между операционной системой и аппаратным обеспечением компьютера, таким как принтеры, сканеры, видеокарты и т.д.
3. **Утилиты**: Эти программы предназначены для обслуживания и оптимизации работы компьютера. Примеры: антивирусы, дисковые утилиты, системные мониторы.
4. **Системные библиотеки**: Это наборы программных компонентов, предоставляющих функциональные возможности для разработки прикладных программ.

Основные виды программного обеспечения.

2. **Прикладное программное обеспечение**:

- **Офисные пакеты**: Включают программы для работы с документами, электронными таблицами, презентациями, базами данных. Примеры: Microsoft Office, Google Workspace (Google Docs, Sheets, Slides).
- **Графические программы**: Используются для создания и редактирования графических изображений и мультимедийных файлов. Примеры: Adobe Photoshop, CorelDRAW.
- **Мультимедийные проигрыватели и редакторы**: Предназначены для воспроизведения и редактирования аудио- и видеофайлов. Примеры: VLC Media Player, Adobe Premiere.
- **Браузеры**: Программы, позволяющие просматривать веб-страницы. Примеры: Google Chrome, Mozilla Firefox.
- **Разработка программного обеспечения**: Включают инструменты для создания прикладных программ, веб-сайтов и мобильных приложений. Примеры: Visual Studio, Android Studio.
- **Игры**: Программы для развлечения и игровых приложений.

Основные виды программного обеспечения.

3. Программное обеспечение для встроенных систем (Embedded Software) представляет собой набор программ, которые управляют функциональностью электронного устройства или системы. Они вшиваются (встраиваются) в микроконтроллеры или процессоры встроенных систем и обеспечивают им работу в соответствии с предназначением.

Основные виды программного обеспечения.

Особенности программного обеспечения для встроенных систем:

1.Ограниченные ресурсы: Программное обеспечение для встроенных систем часто работает на устройствах с ограниченными ресурсами (ограниченным объемом памяти, процессорной мощности и т.д.).

2.Работа в реальном времени: Встроенные системы часто требуют обработки данных в реальном времени и могут быть подвержены строгим временным ограничениям.

3.Стабильность и надежность: Программное обеспечение для встроенных систем должно быть стабильным и надежным, так как неполадки могут иметь серьезные последствия.

4.Оптимизированное управление ресурсами: Важно эффективно использовать ограниченные ресурсы, чтобы обеспечить оптимальную работу устройства.

5.Аппаратная специфика: Встроенные системы часто разрабатываются под конкретные задачи или устройства, поэтому программное обеспечение должно быть адаптировано к аппаратной архитектуре.

6.Минимизация потребления энергии: В некоторых встроенных системах энергопотребление является критическим фактором, и программное обеспечение должно быть спроектировано с учетом эффективности энергопотребления.

Основные виды программного обеспечения.

Примеры встроенных систем:

- Микроконтроллеры в бытовой технике (например, в холодильниках, стиральных машинах).
- Электронные управляющие блоки в автомобилях.
- Промышленные контроллеры для автоматизации производственных процессов.
- Медицинское оборудование.
- Устройства IoT (интернет вещей) и др.

Программное обеспечение для встроенных систем играет ключевую роль в функционировании множества устройств, с которыми мы взаимодействуем ежедневно, и его разработка требует специализированных навыков и знаний.

Основные виды программного обеспечения.

4. Инструментальное программное обеспечение — предназначено для использования в ходе проектирования, разработки и сопровождения программ:

- Текстовые редакторы
- Интегрированные среды разработки
- SDK (набор инструментов для разработки программного обеспечения в одном устанавливаемом пакете.)
- Компиляторы
- Интерпретаторы
- Ассемблеры
- Отладчики
- Профилировщики
- Генераторы документации
- Средства анализа покрытия кода
- Средства непрерывной интеграции
- Средства автоматизированного тестирования
- Системы управления версиями и др.

Классификация программного обеспечения

Жизненный цикл программного обеспечения.

Жизненный цикл программного обеспечения (ЖЦ ПО) — это последовательность этапов и процессов, которые проходит программное обеспечение от момента начала его разработки до окончания его использования и вывода из эксплуатации.

ЖЦ ПО охватывает все этапы, начиная с анализа и планирования, проектирования и разработки, тестирования, развертывания и поддержки программного обеспечения. Основные этапы жизненного цикла программного обеспечения:

1. Анализ и планирование: На этом этапе определяются требования к программному обеспечению путем изучения потребностей пользователей и бизнес-целей. Определяются основные функциональные и нефункциональные требования, а также составляется план разработки программного обеспечения.

Жизненный цикл программного обеспечения.

2.Проектирование: На этапе проектирования разрабатывается архитектура программного обеспечения, определяются структура, компоненты и взаимодействие между ними. Разрабатываются диаграммы классов, диаграммы последовательности и другие модели, которые помогают понять организацию программного обеспечения.

3.Разработка: На этом этапе программное обеспечение фактически создается. Программисты пишут исходный код, используя выбранный язык программирования и инструменты разработки. Разработчики следуют определенным методологиям разработки, таким как водопадная модель, инкрементная модель или гибкая модель разработки.

Жизненный цикл программного обеспечения.

4.Тестирование: Этот этап включает проверку программного обеспечения на соответствие требованиям и его отладку. Разработчики проводят функциональное тестирование, модульное тестирование, интеграционное тестирование и системное тестирование, чтобы обнаружить и исправить ошибки и дефекты в программном обеспечении.

5.Развертывание: На этом этапе программное обеспечение готово к установке и использованию. Оно устанавливается на целевые компьютеры или серверы, настраивается и готовится к работе в реальных условиях. Этот этап может включать перенос данных, обучение пользователей и подготовку документации.

Жизненный цикл программного обеспечения.

6.Эксплуатация и поддержка: После успешного развертывания программного обеспечения оно начинает использоваться в реальных условиях. На этом этапе проводится поддержка и обслуживание программного обеспечения, включая исправление ошибок, обновления, улучшения и добавление новых функций в соответствии с потребностями пользователей.

7.Вывод из эксплуатации: В конечном итоге, программное обеспечение может быть выведено из эксплуатации. Это может происходить, если оно устарело, больше не соответствует требованиям или заменено новым программным обеспечением. На этом этапе может проводиться архивация данных, удаление программного обеспечения и его компонентов, а также планирование перехода на новое решение.

Лицензирование и распространение ПО

Виды лицензий:

- **Проприетарная лицензия (Проприетарное ПО):** Это лицензия, которая ограничивает доступ к исходному коду программы и предоставляет права только на использование программы в соответствии с условиями лицензионного соглашения. Пользователь не имеет права изменять, копировать, распространять или продавать программу.
- **Свободная лицензия (Свободное ПО):** Это лицензия, которая предоставляет пользователям право на свободное использование, изучение, изменение и распространение исходного кода программы. Примеры свободных лицензий включают GNU General Public License (GPL) и Apache License.
- **Коммерческая лицензия:** Это лицензия, которая предоставляется за плату и обычно имеет ограничения на использование программы. Пользователи обычно получают ограниченный набор прав в соответствии с условиями лицензии.
- **Академическая (образовательная) лицензия:** Это специальная лицензия, предоставляемая образовательным учреждениям и студентам для использования программного обеспечения в образовательных целях. Обычно она имеет сниженную стоимость или бесплатна.

Лицензирование и распространение ПО

Открытое и закрытое программное обеспечение:

- **Открытое программное обеспечение (Open Source Software, OSS):** Это программное обеспечение, чей исходный код открыт и доступен для просмотра, изменения и распространения. OSS обычно распространяется с лицензиями, позволяющими свободное использование, модификацию и распространение.
- **Закрытое программное обеспечение (Closed Source Software):** Это программное обеспечение, чей исходный код закрыт и недоступен для общественности. Пользователи могут использовать программу в соответствии с условиями лицензии, но не имеют доступа к исходному коду.

Лицензирование и распространение ПО

Открытое и закрытое программное обеспечение:

- **Открытое программное обеспечение (Open Source Software, OSS):** Это программное обеспечение, чей исходный код открыт и доступен для просмотра, изменения и распространения. OSS обычно распространяется с лицензиями, позволяющими свободное использование, модификацию и распространение. Примеры: Linux, Mozilla Firefox, Apache, Mozilla Thunderbird, LibreOffice, MySQL, WordPress, GIMP, Android, Git и пр.
- **Закрытое программное обеспечение (Closed Source Software):** Это программное обеспечение, чей исходный код закрыт и недоступен для общественности. Пользователи могут использовать программу в соответствии с условиями лицензии, но не имеют доступа к исходному коду.

Тенденции развития программного обеспечения.

1.Искусственный интеллект и машинное обучение: Искусственный интеллект (ИИ) и машинное обучение (МО) стали основными тенденциями в развитии программного обеспечения. Применение ИИ и МО позволяет создавать более интеллектуальные и автономные системы, которые способны обрабатывать и анализировать большие объемы данных, распознавать образы, голос и естественный язык, а также принимать решения на основе данных.

2.Облачные вычисления: Облачные вычисления стали все более популярными. Они позволяют предоставлять доступ к вычислительным ресурсам, хранению данных и приложениям через Интернет. Облачные технологии упрощают развертывание, масштабирование и управление программным обеспечением, а также обеспечивают гибкость и доступность для пользователей.

Тенденции развития программного обеспечения.

3.Интернет вещей (IoT): Развитие Интернета вещей приводит к увеличению количества подключенных устройств, которые взаимодействуют друг с другом и с программным обеспечением. Это открывает новые возможности для создания программного обеспечения, которое управляет и координирует работу различных устройств и сенсоров, осуществляет сбор и анализ данных, а также обеспечивает автоматизацию и управление в различных областях, таких как умный дом, индустрия, здравоохранение и транспорт.

4.Распределенные системы и микросервисная архитектура: Распределенные системы и микросервисная архитектура становятся все более популярными в разработке программного обеспечения. Они позволяют создавать сложные приложения, разбитые на небольшие и независимые сервисы, которые могут масштабироваться и развертываться независимо друг от друга. Это способствует более эффективной разработке, обновлению и масштабированию программного обеспечения.

Тенденции развития программного обеспечения.

5.DevOps и непрерывная поставка (CI/CD): DevOps и непрерывная поставка (Continuous Integration/Continuous Delivery, CI/CD) представляют собой методологии и практики разработки программного обеспечения, направленные на автоматизацию и ускорение процесса разработки, тестирования и развертывания программного обеспечения. Они способствуют более быстрой и гибкой поставке программного обеспечения, что позволяет быстрее реагировать на изменения и требования пользователей.

6.Безопасность программного обеспечения: С ростом угроз в области кибербезопасности все большее внимание уделяется защите программного обеспечения от взломов и нарушений. Разработчики программного обеспечения ставят перед собой задачу создания безопасных систем, применяя передовые методы шифрования, аутентификации и контроля доступа. Также внедряются практики тестирования на безопасность и аудита программного обеспечения.

7. Блокчейн и децентрализованные приложения.

Искусственный интеллект

Машинное обучение (Machine Learning): Машинное обучение является одной из самых активно развивающихся областей искусственного интеллекта.

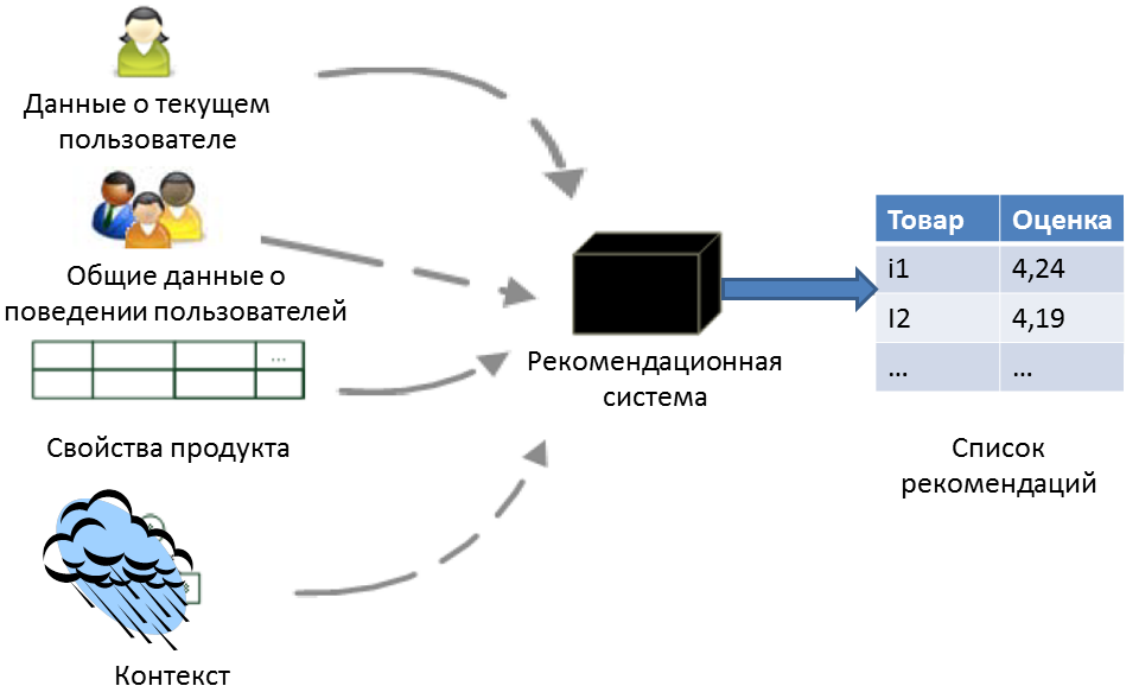
Оно представляет собой методологию, при которой компьютерные системы могут обучаться на основе данных и опыта, а не явно программироваться.

Машинное обучение включает в себя такие подходы, как нейронные сети, алгоритмы классификации, регрессии, кластеризации и многое другое.

Некоторые примеры использования машинного обучения:

Рекомендательные системы:

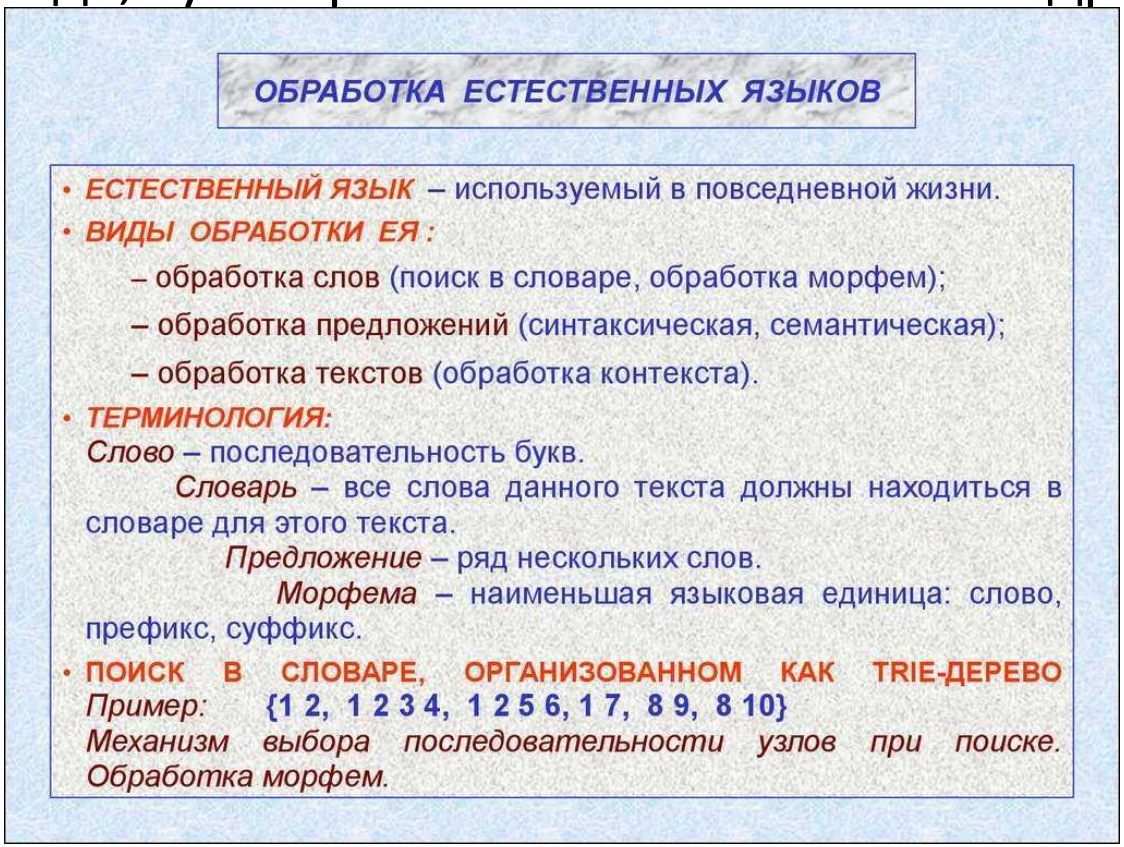
Платформы, такие как Netflix, Amazon, и Spotify, используют ML для предложения пользователям контента и продуктов, которые могут заинтересовать их.



Некоторые примеры использования машинного обучения:

Анализ текста и обработка естественного языка (NLP):

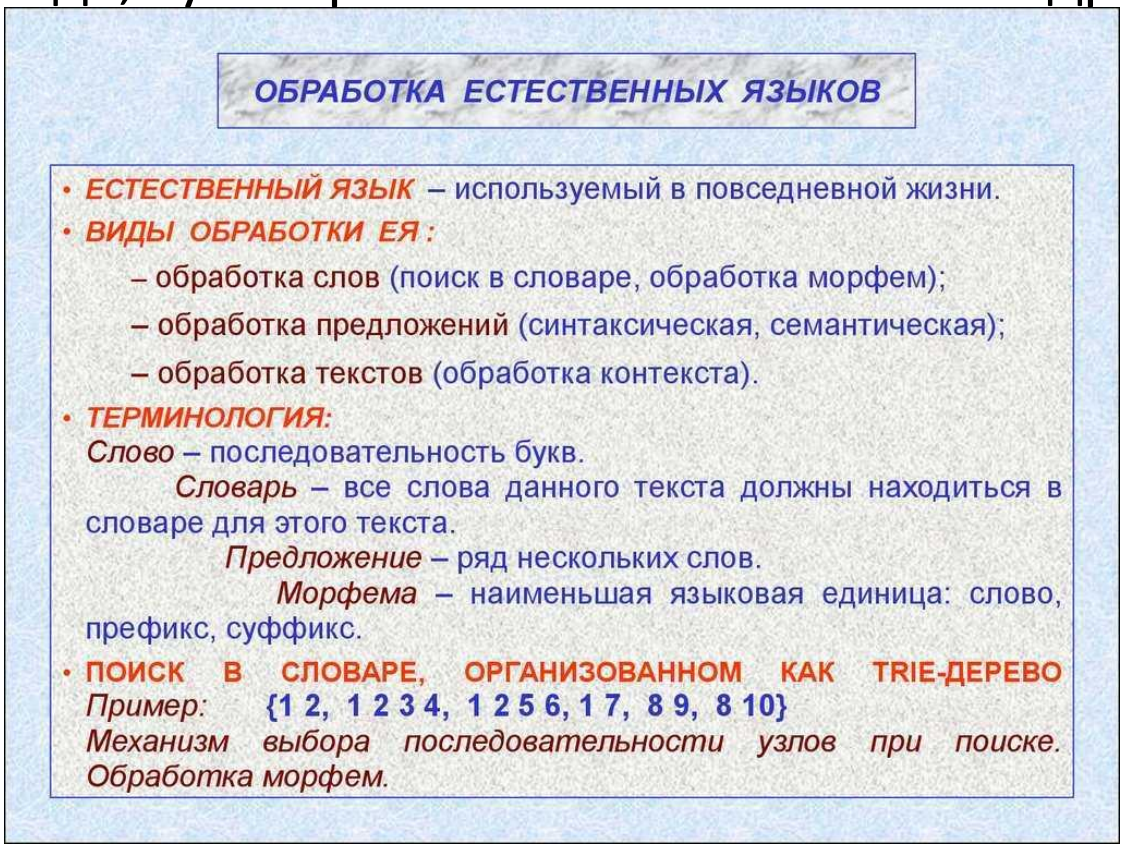
•Алгоритмы машинного обучения могут анализировать тексты, понимать их смысл, выделять ключевую информацию, а также выполнять задачи автоматического перевода, суммирования текстов и многое другое.



Некоторые примеры использования машинного обучения:

Анализ текста и обработка естественного языка (NLP):

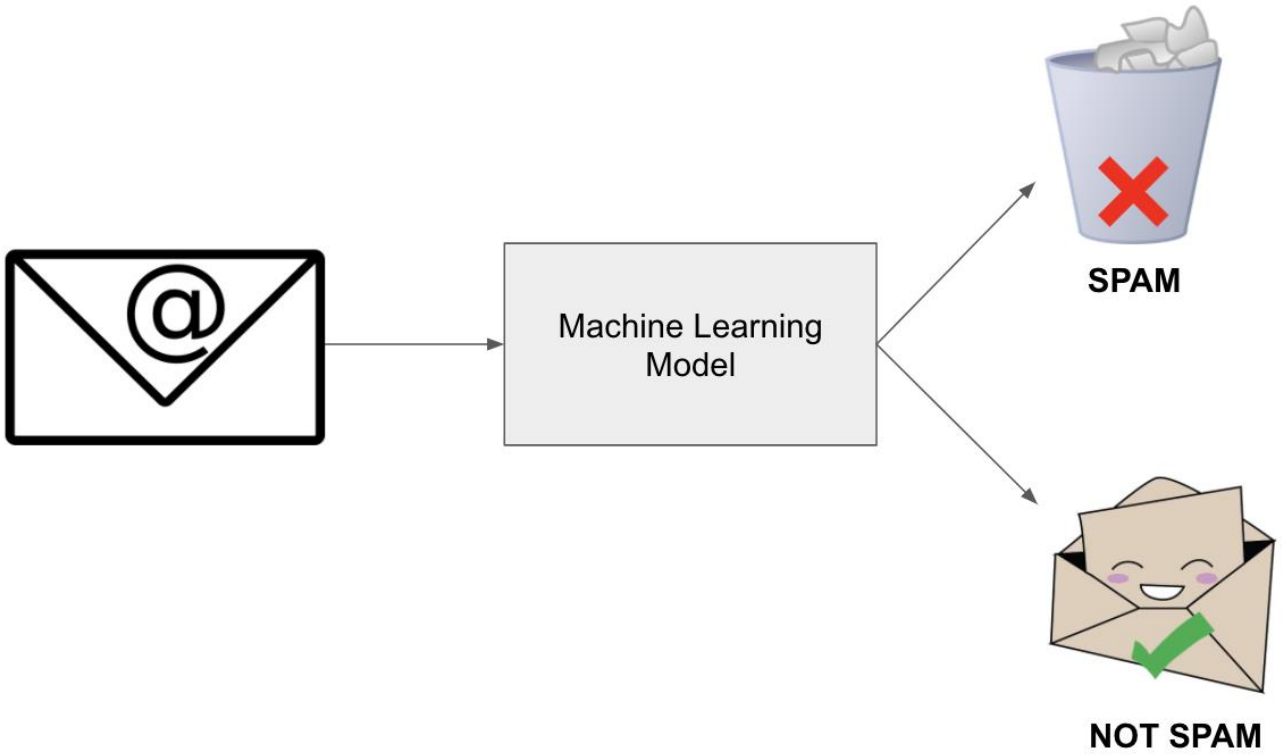
•Алгоритмы машинного обучения могут анализировать тексты, понимать их смысл, выделять ключевую информацию, а также выполнять задачи автоматического перевода, суммирования текстов и многое другое.



Некоторые примеры использования машинного обучения:

Классификация и категоризация:

•ML используется для автоматической классификации данных. Например, фильтрация спама в электронной почте или классификация новостей по темам.



Некоторые примеры использования машинного обучения:

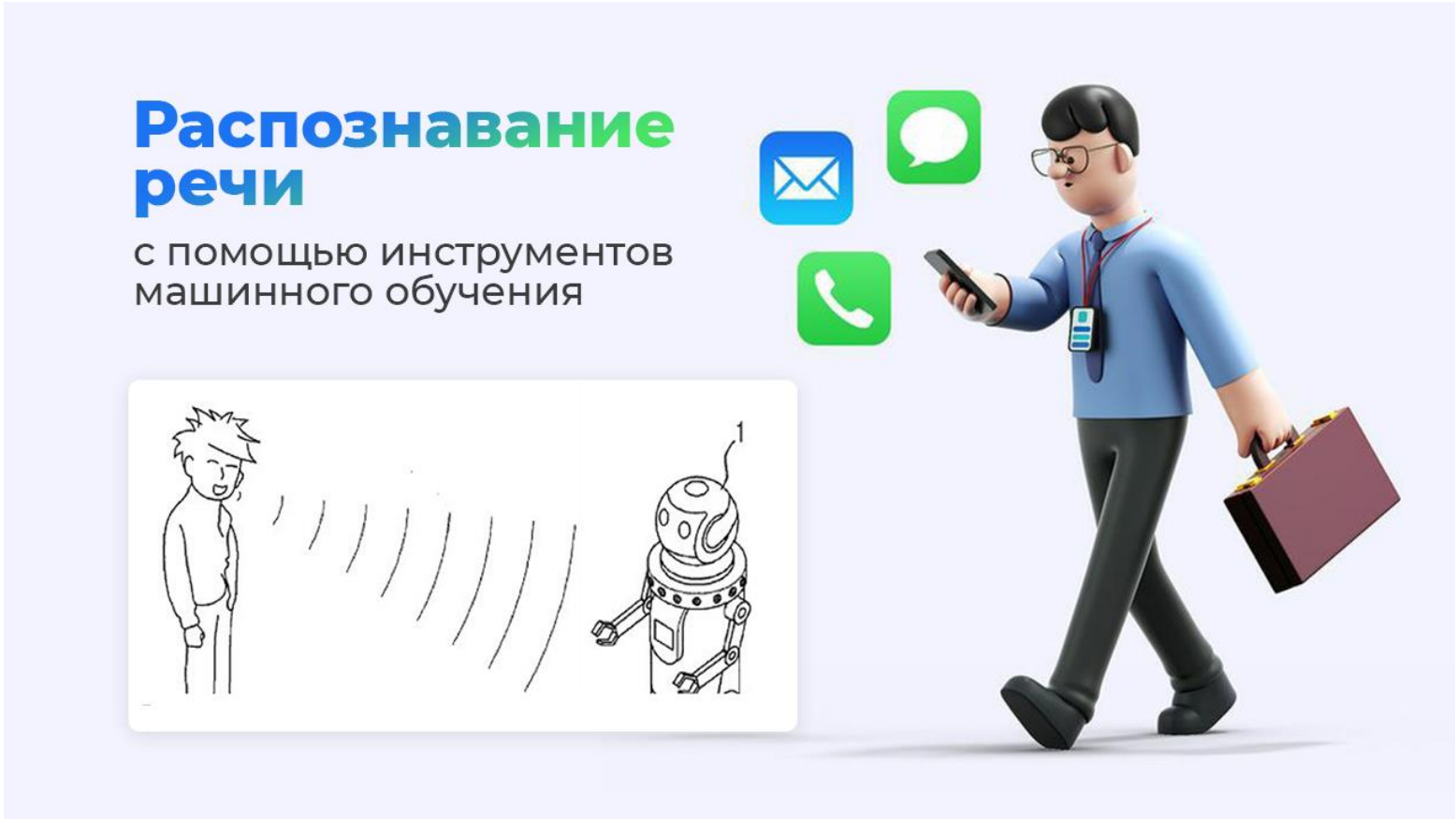
Обработка изображений и компьютерное зрение:

•Системы машинного обучения могут анализировать и обрабатывать изображения, например, распознавать объекты, лица, анализировать медицинские снимки и т.д.

Некоторые примеры использования машинного обучения:

Распознавание речи:

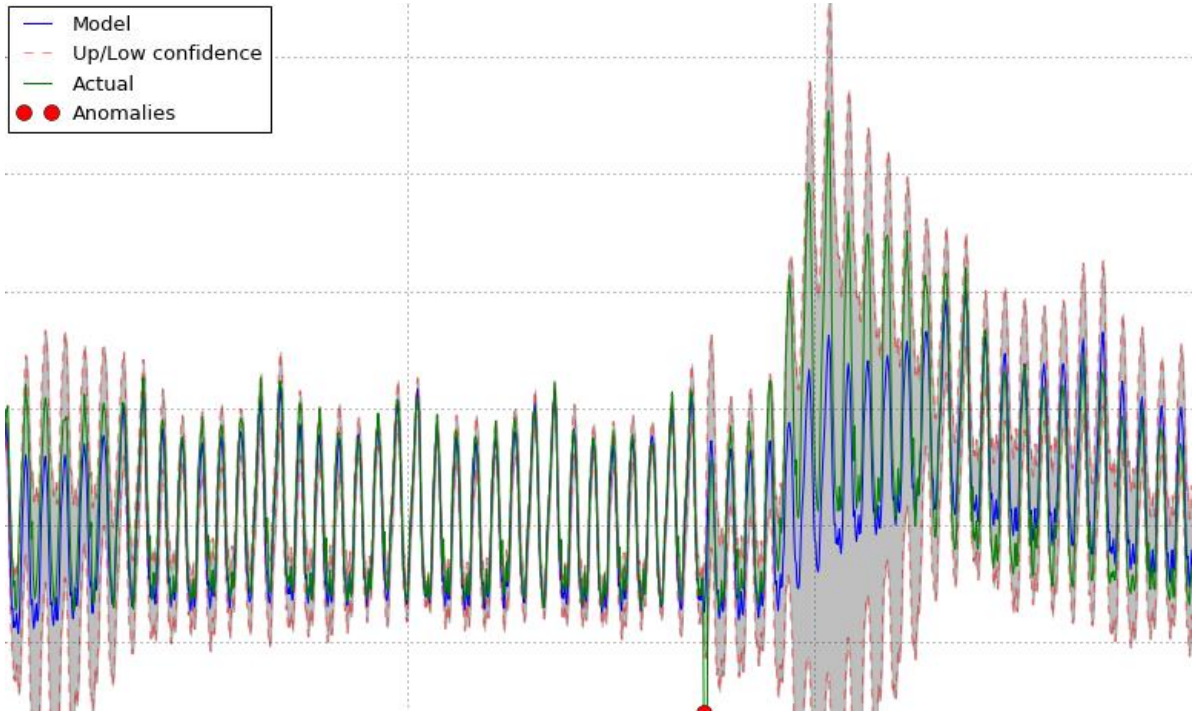
ML используется в системах распознавания и синтеза речи, позволяя компьютерам интерпретировать и генерировать устную речь.



Некоторые примеры использования машинного обучения:

Прогнозирование и анализ временных рядов:

Применяется в прогнозировании финансовых рынков, погоды, анализе потребительского поведения и других областях.



Некоторые примеры использования машинного обучения:

Медицинская диагностика и анализ:

ML может помогать в анализе медицинских изображений (например, рентгеновских снимков), прогнозировании диагнозов и разработке индивидуальных планов лечения.

Некоторые примеры использования машинного обучения:

Автоматизация производства и промышленности:

•Внедрение ML позволяет оптимизировать производственные процессы, контролировать качество продукции и предсказывать сбои оборудования.

Некоторые примеры использования машинного обучения:

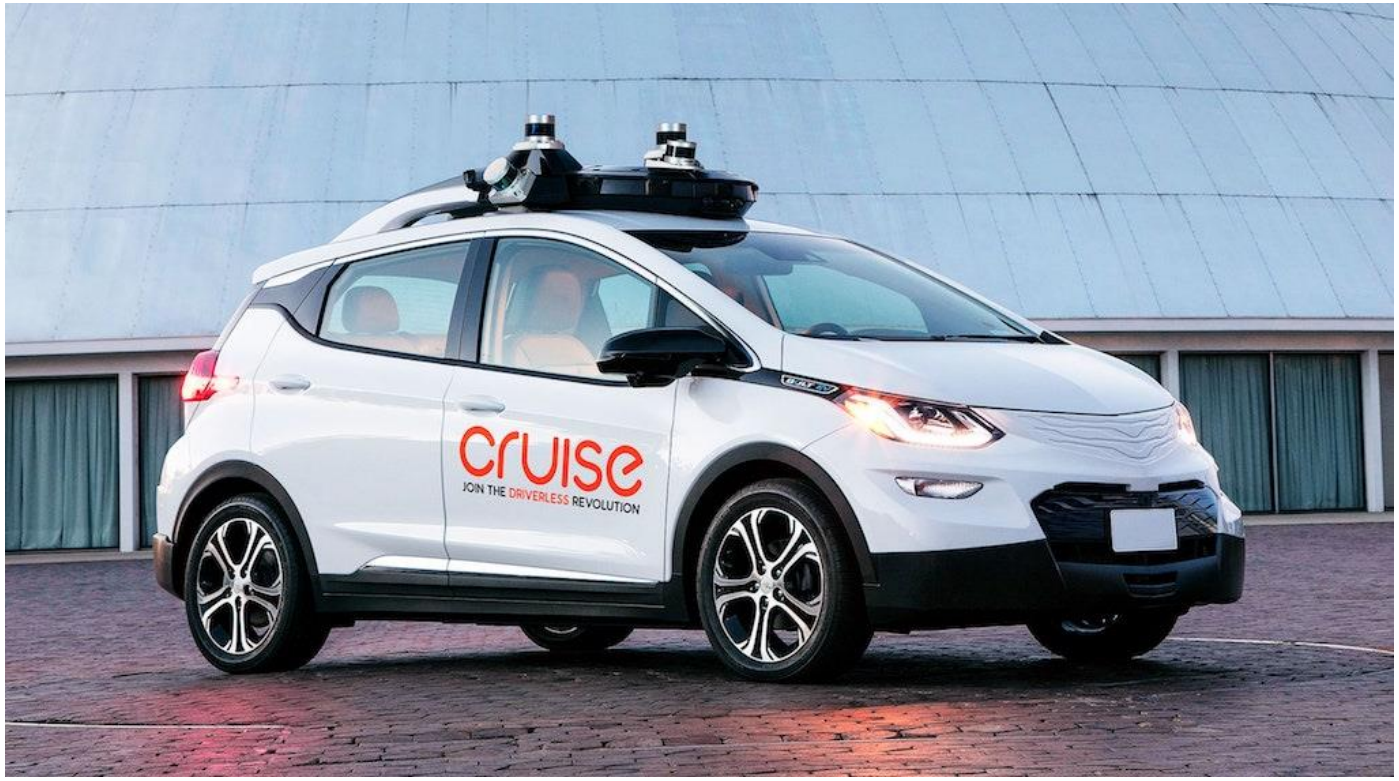
Автоматизация производства и промышленности:

•Внедрение ML позволяет оптимизировать производственные процессы, контролировать качество продукции и предсказывать сбои оборудования.

Некоторые примеры использования машинного обучения:

Автономные транспортные средства:

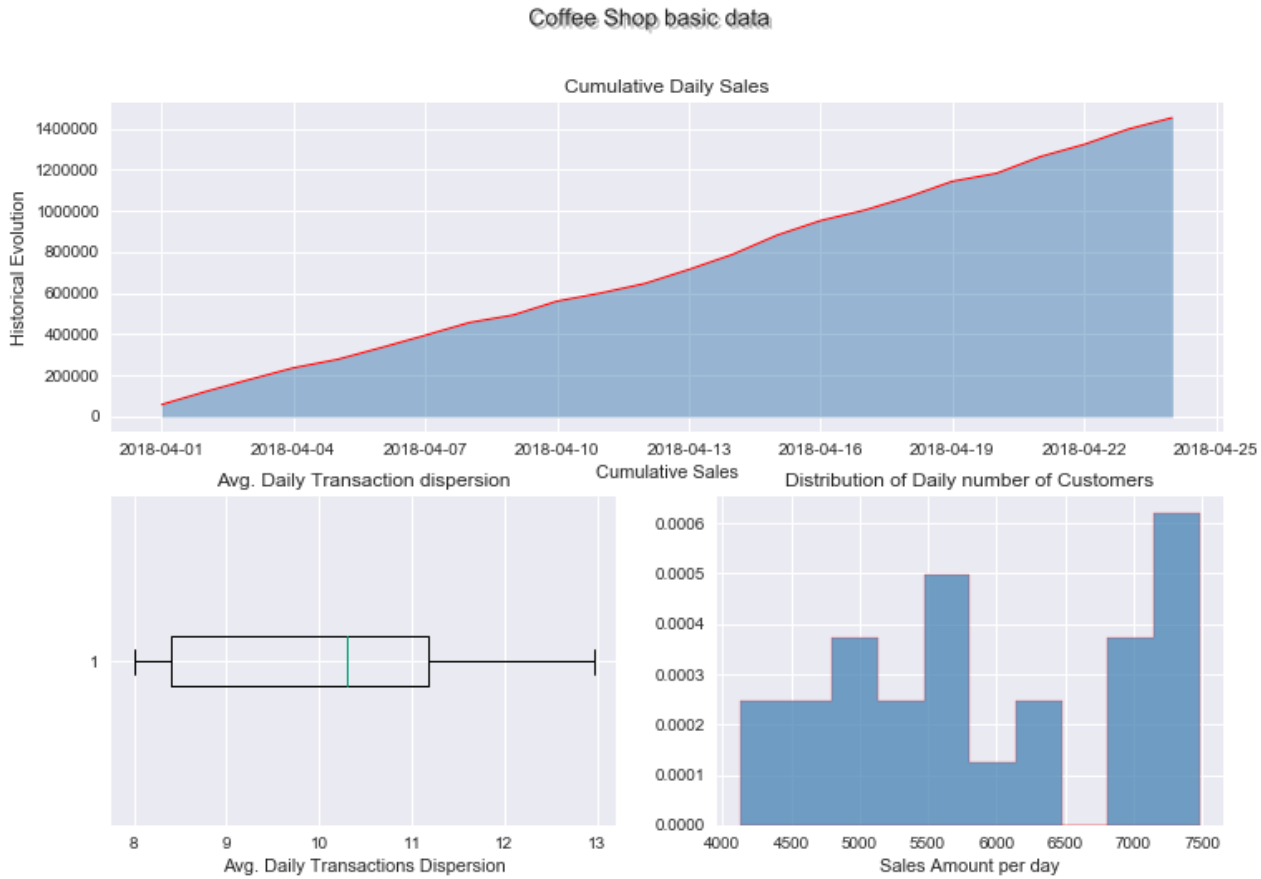
•ML используется в системах управления автопилотами и анализе данных для безопасной и эффективной работы беспилотных транспортных средств.



Некоторые примеры использования машинного обучения:

Финансовый анализ и рисковое управление:

Применяется для прогнозирования рыночных трендов, анализа портфелей инвесторов и управления рисками.



Облачные технологии:

Облачные технологии представляют собой модель предоставления и использования вычислительных ресурсов (включая хранение данных, обработку, сетевые ресурсы и т.д.) через интернет.

Они позволяют компаниям арендовать виртуальные ресурсы вместо того, чтобы иметь собственные физические серверы и инфраструктуру.

Основные характеристики облачных технологий:

1. **Доступность по требованию (On-Demand Access):** Пользователи могут получать доступ к ресурсам в любое время и в любом месте через интернет.

2. **Масштабируемость (Scalability):** Пользователи могут увеличивать или уменьшать объем ресурсов в зависимости от потребностей.

3. **Распределенность (Distribution):** Облачные ресурсы распределены по множеству серверов и центров обработки данных (ЦОД), что обеспечивает отказоустойчивость и высокую доступность.

4. **Экономическая эффективность (Cost Efficiency):** Компании могут снизить капитальные затраты на аппаратное обеспечение и инфраструктуру.

Примеры облачных услуг:

- **Инфраструктура как сервис (IaaS):** Предоставляет виртуальные серверы, хранилища данных и сетевые ресурсы для создания собственной инфраструктуры.
- **Платформа как сервис (PaaS):** Предоставляет платформу и средства разработки для создания, тестирования и развертывания приложений.
- **Программное обеспечение как сервис (SaaS):** Предоставляет готовые к использованию приложения через интернет.

SaaS (Software as a Service):

SaaS - это модель предоставления программного обеспечения, при которой приложения хранятся и обслуживаются в удаленных центрах обработки данных, а пользователи получают доступ к ним через интернет.

Преимущества SaaS:

1.Легкость в использовании: Пользователи могут начать использовать приложение сразу же, не требуя сложной установки и настройки.

2.Регулярные обновления: Поставщик SaaS обычно обновляет и поддерживает приложение, обеспечивая доступ к последним версиям.

3.Доступность с разных устройств: Приложения SaaS могут быть запущены с любого устройства с подключением к интернету.

4.Экономическая эффективность: Пользователи платят за использование приложения по мере необходимости, минимизируя начальные капитальные затраты.

Примеры SaaS-приложений включают Google Workspace, Microsoft Office 365, Salesforce, Dropbox, Figma и многие другие.

SaaS (Software as a Service):

Примеры популярных платформ облачных технологий включают Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), и многие другие.

Они предлагают широкий спектр облачных услуг, включая хостинг виртуальных серверов, хранилище данных, аналитические инструменты, и многие другие.

Интернет вещей (IoT)

Интернет вещей (IoT) представляет собой концепцию, в которой физические объекты, оборудованные датчиками, сетевым подключением и возможностью обмена данными, могут взаимодействовать друг с другом и с окружающей средой.

Эти объекты могут быть различными устройствами, включая домашние электроприборы, автомобили, промышленное оборудование, медицинские устройства и многое другое.

Основная идея Интернета вещей заключается в том, чтобы объединить физический и цифровой мир, позволяя объектам собирать данные, обмениваться информацией и принимать решения на основе анализа полученной информации.

Это создает возможность для автоматизации, оптимизации процессов и повышения эффективности в различных сферах деятельности.

Интернет вещей (IoT)

Важные аспекты Интернета вещей:

1.Сенсоры и актуаторы: Объекты в Интернете вещей оснащены различными сенсорами для сбора данных из окружающей среды. Эти сенсоры могут измерять такие параметры, как температура, влажность, освещенность, движение и т. д. Актуаторы используются для выполнения действий на основе полученных данных, например, управления освещением, регулировки температуры или выполнения определенных операций.

2.Сетевое подключение: Объекты в Интернете вещей подключены к сети, обычно через беспроводные технологии, такие как Wi-Fi, Bluetooth, Zigbee или сотовая связь. Это позволяет им обмениваться данными с другими устройствами и передавать информацию на удаленные серверы для анализа и обработки.

Интернет вещей (IoT)

3.Облачные вычисления и аналитика данных: Данные, собранные устройствами IoT, могут быть переданы на облачные серверы для хранения и анализа. Облачные вычисления обеспечивают высокую масштабируемость и доступность данных, а аналитика данных позволяет получать ценные инсайты и принимать информированные решения на основе собранной информации.

4.Безопасность и конфиденциальность: Интернет вещей ставит перед собой задачу обеспечения безопасности и конфиденциальности данных, передаваемых и хранимых устройствами IoT. Это важный аспект, учитывая, что объекты IoT могут собирать и обрабатывать чувствительную информацию, например, медицинские данные или данные о потреблении энергии.

Применение Интернета вещей охватывает множество областей, таких как умные дома, умные города, промышленность, здравоохранение, транспорт, сельское хозяйство и многое другое. Он открывает новые возможности для повышения эффективности, комфорта и безопасности в нашей повседневной жизни и рабочей среде.

Распределенные системы и микросервисная архитектура:

Распределенные системы и микросервисная архитектура - это два важных понятия, связанных с разработкой программного обеспечения.

Распределенная система - это совокупность автономных компонентов, работающих на нескольких компьютерах или устройствах, которые взаимодействуют и сотрудничают для решения общей задачи.

В распределенных системах компоненты могут выполняться на разных физических серверах, в разных локациях и могут использовать различные типы сетевого взаимодействия для обмена данными.

Распределенные системы и микросервисная архитектура:

Особенности распределенных систем:

1.Прозрачность: Распределенные системы стремятся предоставить единый интерфейс и скрыть детали распределения от пользователей и приложений. Это означает, что пользователь или приложение не должны знать, где и как выполняются различные компоненты системы.

2.Согласованность: Распределенные системы должны поддерживать согласованность данных между различными компонентами. Это достигается с помощью применения алгоритмов согласования и методов репликации данных.

3.Отказоустойчивость: Распределенные системы должны быть устойчивы к отказам компонентов. Если один компонент выходит из строя, другие компоненты должны продолжать работать без проблем.

4.Масштабируемость: Распределенные системы должны быть масштабируемыми, то есть способными обрабатывать растущую нагрузку путем добавления новых компонентов или увеличения ресурсов существующих компонентов.

Распределенные системы и микросервисная архитектура:

Микросервисная архитектура:
Микросервисная архитектура - это подход к разработке программного обеспечения, в котором приложение разбивается на небольшие независимые сервисы, называемые микросервисами.

Каждый микросервис представляет собой отдельно развертываемую и масштабируемую единицу, которая выполняет специфическую функцию и взаимодействует с другими микросервисами через сетевые вызовы.

Распределенные системы и микросервисная архитектура:

Особенности микросервисной архитектуры:

1.Разделение на службы: Приложение разбивается на отдельные микросервисы, каждый из которых отвечает за определенную функциональность. Это позволяет разработчикам сосредоточиться на конкретных сервисах и обеспечивает независимость развертывания и масштабирования каждого сервиса.

2.Легковесность и гибкость: Микросервисы являются небольшими и легковесными, что облегчает их разработку, тестирование и развертывание. Они также могут быть написаны на различных языках программирования и использовать различные технологии по выбору.

3.Независимость и отказоустойчивость: Каждый микросервис может разрабатываться, развертываться и масштабироваться независимно от других микросервисов. Это обеспечивает высокую отказоустойчивость, поскольку отказ одного сервиса не влияет на работу остальных.

4.Горизонтальное масштабирование: Микросервисы могут быть горизонтально масштабируемыми, то есть можно добавлять дополнительные экземпляры сервисов для обработки увеличивающейся нагрузки. Это позволяет эффективно масштабировать приложение без необходимости масштабирования всей системы.

5.Улучшенная модульность и обновление: Микросервисная архитектура способствует модульности приложения, что упрощает его сопровождение и обновление. Каждый сервис может быть разрабатываем, тестируем и развертываться отдельно, без необходимости влияния на остальные сервисы.

Распределенные системы и микросервисная архитектура:

Микросервисная архитектура позволяет компаниям создавать гибкие, масштабируемые и отказоустойчивые приложения, которые легко масштабировать и поддерживать.

Однако она также вносит сложности в управление и координацию между различными сервисами, требуя хорошо спроектированной инфраструктуры и коммуникации между сервисами.

DevOps и непрерывная поставка:

DevOps и непрерывная поставка (Continuous Delivery) являются практиками разработки программного обеспечения, направленными на автоматизацию процессов разработки, тестирования и доставки приложений.

DevOps - это методология разработки программного обеспечения, объединяющая разработчиков (Dev) и операционную команду (Ops), чтобы обеспечить сотрудничество, коммуникацию и автоматизацию во всех этапах жизненного цикла приложения: от разработки и тестирования до развертывания и эксплуатации.

DevOps и непрерывная поставка:

Основные принципы DevOps включают:

1.Сотрудничество и коммуникация: DevOps ставит целью снизить барьеры между разработчиками и операционной командой, поощряя интеграцию и сотрудничество. Коммуникация и обмен знаниями помогают улучшить качество и эффективность процессов разработки и эксплуатации.

2.Автоматизация: DevOps стремится к автоматизации повторяющихся и рутинных задач, таких как сборка, тестирование, развертывание и мониторинг приложений. Автоматизация упрощает процессы, устраняет ошибки, повышает эффективность и снижает время доставки нового функционала.

DevOps и непрерывная поставка:

3.Непрерывное улучшение: DevOps поощряет постоянное улучшение путем обратной связи, мониторинга и анализа метрик производительности. Команды стремятся к постоянному совершенствованию процессов, инструментов и практик с целью достижения большей эффективности и качества.

4.Использование облачных и автоматизированных инструментов: DevOps активно использует облачные ресурсы и инструменты автоматизации для упрощения разработки, тестирования, развертывания и масштабирования приложений. Это включает в себя инфраструктуру как код, контейнеризацию, оркестрацию контейнеров и многое другое.

Непрерывная поставка :

Непрерывная поставка (Continuous Delivery): Непрерывная поставка - это практика, в рамках которой разработчики стремятся доставлять новый функционал и исправления ошибок в приложение с минимальными задержками и рисками.

Она основана на автоматической сборке, тестировании и развертывании приложений с использованием набора автоматизированных инструментов и процессов.

Непрерывная поставка :

Основные принципы непрерывной поставки включают:

1.Автоматизированная сборка и тестирование: Разработчики используют инструменты для автоматической сборки и тестирования приложения на каждом этапе разработки. Это включает модульное тестирование, интеграционное тестирование и тестирование производительности.

2.Развертывание приложения в любой момент: При использовании непрерывной поставки приложение может быть развернуто в любой момент времени. Это позволяет быстро доставать новые функции и исправления ошибок, минимизируя время задержки между разработкой и развертыванием.

Непрерывная поставка:

3.Автоматизированное развертывание: Непрерывная поставка включает автоматизированное развертывание приложений с использованием инструментов, таких как системы управления конфигурациями, контейнеризация и оркестрация. Это обеспечивает быстрое и надежное развертывание приложений в различных средах, включая тестовую, предпродакшн и продакшн.

4.Непрерывный мониторинг: В рамках непрерывной поставки обеспечивается постоянный мониторинг приложения в производственной среде. Это позволяет оперативно обнаруживать проблемы и устранять их, а также собирать данные для анализа производительности и использования ресурсов.

Непрерывная поставка :

Преимущества DevOps и непрерывной поставки включают более быструю итерацию разработки, улучшенное качество программного обеспечения, снижение рисков и более эффективное использование ресурсов.

Они позволяют командам разработчиков и операций тесно взаимодействовать, ускоряя процессы и доставку нового функционала пользователям.

Важно отметить, что DevOps и непрерывная поставка - это не только набор инструментов и процессов, но и культурный и организационный подход.

Они требуют совместной работы и коммуникации между различными командами в организации, а также готовности к изменениям и постоянному совершенствованию.

Блокчейн:

Блокчейн - это распределенная база данных, в которой информация хранится в виде непрерывно растущих блоков, связанных между собой с помощью криптографических методов.

Каждый блок содержит набор транзакций или записей данных, а также хэш предыдущего блока, что обеспечивает целостность и невозможность изменения предыдущих блоков без изменения всей цепи блоков.

Одной из ключевых особенностей блокчейна является децентрализация. Традиционно данные хранятся в централизованных системах, где управление и контроль осуществляются центральными организациями или учреждениями.

В блокчейне же данные распределены между множеством участников сети, называемых узлами. Каждый узел хранит полную копию блокчейна и принимает участие в подтверждении и проверке транзакций.

Блокчейн :

Основные характеристики децентрализованных приложений включают:

1.Прозрачность и невозможность подделки: Благодаря применению криптографии и хранению данных в блокчейне, DApps обеспечивают прозрачность операций и невозможность подделки или изменения данных без согласия большинства узлов в сети.

2.Управление с использованием смарт-контрактов: DApps используют смарт-контракты - программные коды, которые выполняются автоматически при выполнении определенных условий. Смарт-контракты позволяют автоматизировать и контролировать выполнение операций без посредников.

Блокчейн :

2.Участие сообщества: Децентрализованные приложения основываются на участии сообщества узлов. Разработчики и пользователи участвуют в принятии решений и развитии приложения через процессы голосования или консенсусные механизмы.

3.Открытость и свобода доступа: Децентрализованные приложения могут быть доступны для всех пользователей без ограничений и цензуры. Это позволяет создавать приложения, которые не зависят от определенных организаций или правительств.

Блокчейн :

Примеры децентрализованных приложений включают криптовалюты, такие как Bitcoin и Ethereum, децентрализованные биржи, децентрализованные социальные сети (Диаспора (diasporafoundation.org), SocialX (socialx.network), Minds (wefunder.com/minds)), онлайн-рынки и многое другое.

Они стремятся к созданию более открытых, прозрачных и инклюзивных систем, где управление и контроль лежат в руках пользователей.

Заключение:

В данной лекции мы рассмотрели важные аспекты введения в программное обеспечение.

Мы изучили основные понятия и принципы разработки программного обеспечения, а также рассмотрели его жизненный цикл и тенденции развития.