Geekbrains

Backend - разработка web - приложения информационного сайта на языке программирования Python с использованием фреймворка Django

Программа: «Python - разработчик»

Специализация: «Backend - разработчик»

Касимов И. Т.

Магнитогорск

2024

Глава 1. Программирование и его направления

* 1. Что такое программирование

Программирование - процесс создания и модификации компьютерных программ. Мы пользуемся запрограммированной техникой каждый день. Обычно программирование выглядит как написание программного кода на каком-нибудь языке программирования. С его помощью разработчики создают сайты, приложения, умные устройства, роботов и разнообразные цифровые сервисы.

В более широком смысле программирование - любое создание инструкций для исполнителя. Инструкции пишут в виде кода на различных языках для исполнителя. Исполнитель - устройства выполняющие эти инструкции. Когда человек программирует, он, по сути, переводит задачи для машины на язык, который ей понятен. Инструкции, написанные на этом языке, могут запускаться и выполняться - компьютер им следует. Так получаются программы. Люди общаются друг с другом на естественных языках, разработчики же с компьютерами взаимодействуют на языках программирования. Написанный программистом код - это алгоритм действий, который должно выполнить устройство. Таким образом, язык программирования помогает человеку записать понятную компьютеру последовательность операций.

Все виды языков программирования, суть которых сводится к преобразованию понятных человеку команд в машинный код, обладают строгим синтаксисом. К примеру, конец строки должен заканчиваться точкой с запятой. Такое правило позволяет компьютеру отделять команды друг от друга.

В современном мире программирование стало неотъемлемой частью IT - индустрии, и его роль постоянно растет. Основной целью программирования является написание программ, которые выполняют определенные задачи или решают определенные проблемы. Программы могут быть разработаны для различных платформ и устройств, начиная от компьютеров и мобильных устройств, и заканчивая встроенными системами и сетевыми устройствами.

Программирование включает в себя не только написание кода, но и планирование, анализ, тестирование и отладку программного обеспечения. В процессе программирования программисты используют различные инструменты и технологии для создания эффективных и надежных программ.

В современном мире программирование играет ключевую роль в различных отраслях, таких как разработка программного обеспечения, веб-разработка, машинное обучение, искусственный интеллект, кибербезопасность и другие. Понимание основ программирования является важным навыком для специалистов IT - индустрии и позволяет создавать инновационные решения и продукты.

Таким образом, программирование представляет собой творческий процесс, направленный на создание программного обеспечения для удовлетворения потребностей пользователей и решения различных задач. В рамках дипломной работы по программированию можно изучить различные методы разработки программ, алгоритмы, структуры данных, технологии программирования и другие аспекты, связанные с созданием качественного программного обеспечения.

1.2 Направление языков программирования

Направления языков программирования охватывают широкий спектр технологий и методов, используемых для разработки программного обеспечения. Различные языки программирования имеют свои уникальные особенности, преимущества и области применения, что делает их подходящими для различных задач и проектов. В рамках данного исследования рассмотрим основные направления языков программирования и их роль в современной IT - индустрии.

Одним из основных направлений языков программирования являются процедурные языки. Они используются для описания последовательности действий, выполняемых компьютером для достижения определенной цели. Примерами процедурных языков являются C, Pascal, и BASIC. Эти языки часто используются для написания системного программного обеспечения, операционных систем, и встроенных приложений.

Другим важным направлением языков программирования являются объектно-ориентированные языки. В данном подходе программа строится из объектов, которые содержат данные и методы для их обработки. Языки, такие как Java, C++, и Python, широко используются для разработки сложных приложений, веб-сервисов, и игр.

Функциональное программирование также представляет важное направление языков программирования. Основным принципом этого подхода является работа с функциями как основными строительными блоками программы. Языки, такие как Haskell, Lisp, и Scala, позволяют писать более выразительный и функциональный код, что способствует улучшению производительности и читаемости программ.

Еще одним важным направлением языков программирования являются скриптовые языки. Они обеспечивают быструю разработку и исполнение программы без предварительной компиляции. Примерами скриптовых языков являются JavaScript, PHP, и Ruby. Эти языки широко используются для создания веб - страниц, обработки данных, и автоматизации задач.

Таким образом, направления языков программирования представляют собой разнообразные подходы к разработке программного обеспечения, каждое из которых имеет свои особенности и преимущества. Изучение и понимание различных языков программирования позволяет разработчикам выбирать наиболее подходящий язык для конкретного проекта и достигать оптимальных результатов. В дальнейшем исследовании будут рассмотрены более детально особенности каждого из направлений языков программирования и их влияние на развитие современной IT - индустрии.

Глава 2. Виды разработки в программировании.

2.1 Основные виды.

1. Backend - разработчик. Backend Developer - наиболее популярный тип разработчиков. Они создают серверно - административную часть продукта, то есть пишут код для внутренней стороны сайта, которую не видит пользователь. Такой программист разрабатывает фундамент проекта и настраивает его работу.

За что отвечает: база данных, архитектура, логика продукта, системы обработки и хранения данных.

Языки и технологии: Python, С++, Java, PHP, Ruby, Go. Хороший бэкенд-разработчик также должен разбираться в базе данных MySQl, PostgreSQL или NoSQL.

1. Frontend - разработчик. Входят в тройку самых востребованных программистов. Он отвечает за пользовательскую сторону приложения и работают на стороне клиента. Цель frontend - разработчика - сделать удобный и интуитивно понятный интерфейс для пользователя. Например, интерактивные переходы страниц, всплывающее меню, меняющие цвет кнопки — всем этим занимается frontend - developer. Также в его задачи входит правильное отображение сайта или приложения на компьютере и на разных электронных гаджетах. Ну а если в проекте отсутствует верстальщик, берет его задачи на себя - работает с HTML-кодом.

За что отвечает: пользовательский интерфейс, кросс-браузерные ошибки, верстка шаблона сайта, адаптивная и мобильная верстка.

Языки и технологии: JavaScript, CSS, интерфейсные среды (React, jQuery или Angular), HTML, препроцессоры SASS/LESS. Дополнительные знания: серверные технологии, основы SEO-оптимизации, веб-шрифты, графические редакторы.

1. Fullstack - разработчик. Это универсальный программист полного цикла, который сочетает в себе знания frontend - и backend -разработчика. Fullstack - разработчик полностью разрабатывает web - проект или сайт: от программно-административной части до клиентского интерфейса. Такие разработчики нужны для создания полного проекта и востребованы на рынке.

За что отвечает: пользовательская и серверная часть сайта, кроссплатформенные приложения, интеграция сервисов на frontend - и backend.

Языки и технологии: JavaScript для браузерной части, PHP, Java или Python для серверной части, HTML, CSS, базы данных, фреймворки.

1. Мобильный разработчик. К ним относятся программисты, которые делают приложения для мобильных устройств ОС Android и iOS. Помимо телефонов, к таким устройствам относятся: GPS - навигаторы, умные часы, электронные книги.

Как правило, Android - программисты разбираются в обеих частях приложения, отвечая за полный цикл разработки, поэтому строгого деления на frontend - и backend у них нет.

За что отвечает: полный цикл разработки приложений для мобильных устройств.

Языки для Android - разработчика: Java, Kotlin, OpenGL, Android SDK.

Языки для iOS-разработчика: Objective-C, Swift, CoreData, CoreGraphics.

1. Разработчик видеоигр. Программисты, которые разрабатывают видеоигры, а именно:

* desktop - приложения (программы, которые работают под управлением ОС и не зависят от других приложений);
* мобильные игры;
* ролевые онлайн-игры MMORG.

Они работают над ПО, которое предназначено для ОС Windows, Apple OSX или Linux. Некоторые Game-разработчики создают бизнес-приложения для конкретной компании.

За что отвечает: игровая разработка, веб-дизайн, обновления.

Языки и технологии: C/С++, С#, Java, Open GL/DirectX, игровые движки (Unity, Unreal Engine, Torque), графические библиотеки.

1. Инженер - программист. Разрабатывает программы для разных устройств на предприятиях и заводах: станки ЧПУ, конвейеры, хлебопечки и другие. Основное отличие от остальных разработчиков - техническая экспертность с опытом более 10 лет. Инженер-программист обладает глубокими знаниями дискретной математики и физики, понимает промышленные и технологические процессы.

За что отвечает: автоматизация производства и программирование внутренних устройств.

Языки и технологии: C, C++, C#, Delphi, Assembler, технические знания.

1. DevOps - инженер. Одна из самых сложных позиций в ИТ-рекрутменте, потому что не всегда понятно, чем занимается такой специалист. DevOps - это методика повышения качества программного обеспечения, а DevOps - инженер использует ее для синхронизации всех этапов создания ИТ-продукта и совмещает в себе разработчика, тестировщика, менеджера и сисадмина. Он контролирует и автоматизирует работу разработчиков и других ИТ - специалистов, которые связаны с продуктом.

За что отвечает: кодирование, тестирование, запуск приложения, автоматизация, внедрение программных инструментов.

Языки и технологии: PHP, Perl, Ruby, Python, С++, Cloud Automation (Azure, GCP, Alibaba), Jira, системы мониторинга сетевых устройств, ОС Windows/Linux, ПО для автоматизации (Docker, Jenkins, Puppet).

1. Embedded - разработчик. Относительно новая и узкая специализация разработчиков, которые работают со встроенными устройствами, то есть с девайсами и гаджетами, в которых есть аппаратная платформа. Например, микроконтроллеры, встроенные программы и устройства. Embedded - developer умеет создавать ПО, а также должен разбираться в физических процессах электрических компонентов.

За что отвечает: разработка ПО, тестирование и отладка, оптимизация оборудования, разработка решений для диагностики сбоев устройств.

Языки и технологии: C/C ++, Assembler, компьютерные алгоритмы, инженерные и математические знания.

Мы рассказали об основных видах разработчиков, которые встречаются в ИТ - вакансиях. Наши рекрутеры знают, как подбирать таких разработчиков и помогут найти классного специалиста. Оставляйте заявку на нашем сайте.

2.2 Backend – разработка.

2.1.1 Что такое backend – разработка?

Backend - разработка, или разработка серверной части программного обеспечения, играет ключевую роль в создании современных веб-приложений. Backend отвечает за обработку запросов от клиентской стороны, взаимодействие с базами данных, бизнес-логику приложения и многие другие аспекты работы системы. Основная задача backend - разработчиков - создать структуру, которая обеспечивает надежную и эффективную работу приложения.

Backend - разработка основывается на использовании различных технологий и языков программирования. Часто для создания серверных приложений используются языки программирования, такие как Java, Python, PHP, Ruby, и другие. Эти языки позволяют разработчикам реализовать функционал серверной части приложения, обеспечивают возможность взаимодействия с базами данных, обработку запросов от клиентов, а также обеспечивают безопасность и масштабируемость системы.

Важным аспектом backend - разработки является обеспечение безопасности приложения. Backend - разработчики должны уделять особое внимание защите данных пользователей, обработке запросов безопасным образом, предотвращению уязвимостей, и многим другим аспектам, связанным с кибербезопасностью.

Еще одной важной составляющей backend - разработки является оптимизация производительности. Разработчики должны создавать эффективные и масштабируемые решения, которые обеспечивают быструю обработку запросов и отзывчивость системы даже при больших нагрузках.

* + 1. Backend - разработка на языке программирования Python.

Backend - разработка на Python представляет собой процесс создания серверной части веб-приложений с использованием языка программирования Python. Python - это мощный и гибкий язык программирования, который широко применяется в веб-разработке благодаря своей простоте и выразительности.

Backend - разработчики, работающие с Python, отвечают за реализацию функционала серверной части приложения. Их задачи включают в себя разработку API, взаимодействие с базами данных, обработку запросов от клиентов, реализацию бизнес - логики приложения и обеспечение безопасности данных.

Python имеет богатое экосистему инструментов и фреймворков, которые облегчают разработку серверной части приложений. Некоторые из наиболее популярных фреймворков для Backend - разработки на Python включают Django, Flask, FastAPI и другие. Эти фреймворки предоставляют разработчикам готовые инструменты для работы с HTTP - запросами, маршрутизацией, авторизацией и многими другими задачами.

Python отличается высокой читаемостью кода, простотой синтаксиса и богатой стандартной библиотекой, что делает его привлекательным языком для создания серверных приложений. Кроме того, Python поддерживает асинхронное программирование, что позволяет создавать высокопроизводительные и отзывчивые веб-приложения.

Backend - разработка на Python представляет собой увлекательный и перспективный направление в сфере веб - разработки. Использование Python в качестве основного инструмента для создания серверных приложений позволяет разработчикам эффективно реализовывать функционал приложения, обеспечивая высокую производительность и безопасность.

Глава 3. Основные понятия web – технологий.

3.1 Основы HTML.

3.1.1 Что такое HTML?

HTML (Hyper Text Markup Language) — стандартизированный язык гипертекстовой разметки документов для просмотра веб-страниц в браузере. Веб-браузеры получают HTML документ от сервера по протоколам HTTP/HTTPS или открывают с локального диска, далее интерпретируют код в интерфейс, который будет отображаться на экране монитора. Понимание основ HTML и его особенностей имеет важное значение для любого web – разработчика.

HTML использует теги, которые представляют собой метки, используемые для разметки и классификации информации, что упрощает для браузеров интерпретацию и форматирование контента на веб-сайте. Проще говоря, теги служат ключевыми словами, которые указывают браузеру, как обрабатывать и отображать содержимое сайта.

3.1.2 Структура элементов разметки HTML

Ниже представлена структура HTML элемента:



Рисунок 1. Структура элементов разметки HTML.

Главными частями нашего элемента являются:

1. Открывающий тег. Состоит из имени элемента (в данном случае, "h1"), заключённого в открывающие и закрывающие угловые скобки. Открывающий тег указывает, где элемент начинается или начинает действовать, в данном случае — где начинается заголовок первого уровня.
2. Закрывающий тег - это то же самое, что и открывающий тег, за исключением того, что он включает в себя косую черту перед именем элемента (в данном случае, "/h1"). Закрывающий элемент указывает, где элемент заканчивается, в данном случае — где заканчивается заголовок первого уровня. Отсутствие закрывающего тега является одной из наиболее распространённых ошибок начинающих.
3. Контент - то контент элемента, который в данном случае является просто текстом.
4. Элемент - Открывающий тег, закрывающий тег и контент вместе составляют элемент.

HTML элементы могут иметь атрибуты. Атрибуты предназначены для добавления дополнительной информации об элементе и/или для изменения его стандартного поведения. Атрибуты всегда указываются внутри открывающего тега. В большинстве случаев атрибуты являются необязательными и указываются только при необходимости. Один элемент никогда не должен содержать в себе два и более атрибутов, имеющих одинаковое имя. Синтаксис элемента с атрибутом:



Рисунок 2. Атрибут элемента.

Атрибуты состоят из пары: имя\_атрибута="значение". Имена атрибутов не чувствительны к регистру символов и могут быть записаны с любым сочетанием строчных и заглавных букв. Значение атрибута может состоять из текста, цифр и других символов (единственным исключением является символ амперсанда (&), его использование запрещено). Дополнительные ограничения на использование различных символов в значении атрибута зависят от способа записи значения. При указании нескольких атрибутов в одном элементе, они должны быть отделены друг от друга символом пробела. Порядок следования атрибутов не имеет значения.

Атрибуты часто используются для поиска элементов на странице, поэтому необходимо обратить на них внимание при изучении HTML.

3.2 Основы CSS

3.2.1 Что такое CSS?

CSS (Cascading Style Sheets) — это код, используемый для настройки внешнего вида вашей веб-страницы. Каскадные таблицы стилей предоставляют дополнительные возможности для разметки html-документа и свободу по созданию уникального дизайна для веб-страниц.

Когда браузер обрабатывает HTML-код, он использует встроенный по умолчанию стиль представления HTML-элементов на веб-странице. Чтобы понять, что такое "стиль по умолчанию" рассмотрим в качестве примера элементы <h1> - <h6>: заголовки являются блочными элементами, занимают всю доступную ширину в родительском элементе, имеют разрыв строки до и после элемента, текст заголовка отображается жирным начертанием и имеет определённый размер, в зависимости от уровня заголовка, всё это вместе является встроенным стилем для заголовков.

С помощью CSS можно переопределить установленный для элементов стиль по умолчанию на свой собственный, создав тем самым уникальный стиль оформления для элементов веб-страницы, например изменить цвет текста заголовка и размер шрифта, выделить изображение красной рамкой и т.д.

Чем полезны таблицы стилей помещенные в отдельный документ? Ответ очень прост, можно собрать все стили, которые используются на сайте, в один внешний файл с расширением .css и связать его со всеми страницами сайта. После этого, когда вы будете редактировать стиль, изменения моментально затрагивают все элементы на страницах сайта, где есть ссылка на данный внешний файл со стилями. Таким образом, вы можете полностью изменить внешний вид путем редактирования единственного файла таблицы стилей, что существенно упрощает работу, нежели редактировать стили на каждой странице в отдельности.

3.2.2 Синтаксис CSS

CSS представляет из себя правило или набор правил, описывающих форматирование (изменение внешнего вида) отдельных элементов на веб-странице. Правило, состоит из двух частей: селектора и следующим за ним блоком объявлений. На изображении ниже показана структура (синтаксис) правила:

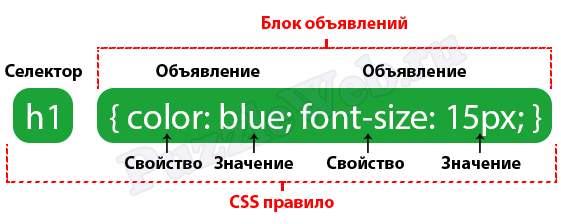


Рисунок 3. Синтаксис CSS.

* Первым всегда указывается селектор, он сообщает браузеру, к какому элементу или элементам веб-страницы будет применен стиль.
* Далее следует блок объявлений, который начинается с открывающей фигурной скобки {и заканчивается закрывающей}, между фигурными скобками указываются форматирующие команды (объявления), которые используются браузером для стилизации выбранного селектором элемента. В блоке может быть указано сколько угодно объявлений. Сам блок объявлений так же называют просто стилем.
* Каждое объявление состоит из двух частей: свойства и его значения. Объявление всегда должно заканчиваться точкой с запятой (;). Опустить указание ";" можно только в конце последнего объявления перед закрывающей фигурной скобкой.
* Свойство - это команда форматирования, определяющая конкретный стилевой эффект для элемента. Само свойство представляет из себя слово или несколько написанных через дефис слов. Каждое свойство имеет свой предопределенный набор значений. После имени свойства указывается двоеточие, которое отделяет название свойства от допустимого значения.

3.3 Основы JavaScript

JavaScript – это интерпретируемый язык программирования, разработанный для взаимодействия с веб-страницами. JavaScript представляет собой реализацию ECMAScript. ECMAScript – это просто описание языка, который определён в стандарте ECMA-262.

Возможности:

Итак, небольшой список того, что позволяет JavaScript:

* Добавлять различные эффекты анимации
* Реагировать на события - обрабатывать перемещения указателя мыши, нажатие клавиш с клавиатуры
* Осуществлять проверку ввода данных в поля формы до отправки на сервер, что в свою очередь снимает дополнительную нагрузку с сервера
* Создавать и считывать cookie, извлекать данные о компьютере посетителя
* Определять браузер
* Изменять содержимое HTML-элементов, добавлять новые теги, изменять стили
* Этим конечно же список не ограничивается, так как помимо перечисленного JavaScript позволяет делать и многое другое.

Ограничения:

Существуют так же и некоторые ограничения, распространяемые на данный язык:

* JavaScript не может закрывать окна и вкладки, которые не были открыты с его помощью

Не может защитить исходный код страницы и запретить копирование текста или изображений со страницы

* Не может осуществлять кросс доменные запросы, получать доступ к веб-страницам, расположенным на другом домене. Даже когда страницы из разных доменов отображаются в одно и тоже время в разных вкладках браузера, то код JavaScript принадлежащий одному домену не будет иметь доступа к информации о веб-странице из другого домена. Это гарантирует безопасность частной информации, которая может быть известна владельцу домена, страница которого открыта в соседней вкладке
* Не имеет доступа к файлам, расположенным на компьютере пользователя, и доступа за пределы самой веб-страницы, единственным исключением являются файлы cookie, это небольшие текстовые файлы, которые JavaScript может записывать и считывать
* В целом, можно сказать, что он разработан таким образом, чтобы затруднить выполнение вредоносного кода.

Глава 4. Git. Знакомство с контролем версий

* 1. Основы систем контроля версий

Система контроля версий — это система, записывающая изменения в файл или набор файлов в течение времени и позволяющая вернуться позже к определённой версии. Для контроля версий файлов в этой книге в качестве примера будет использоваться исходный код программного обеспечения, хотя на самом деле вы можете использовать контроль версий практически для любых типов файлов.

Система контроля версий Git является распределенной, она используется для отслеживания изменений в файлах и директориях проекта. Позволяет разработчикам контролировать и отслеживать изменения в проекте.

Программное обеспечение контроля версий отслеживает все вносимые в код изменения в специальной базе данных. При обнаружении ошибки разработчики могут вернуться назад и выполнить сравнение с более ранними версиями кода для исправления ошибок, сводя к минимуму проблемы для всех участников команды.

Практически во всех программных проектах исходный код является сокровищем: это ценный ресурс, который необходимо беречь. Для большинства команд разработчиков программного обеспечения исходный код — это репозиторий бесценных знаний и понимания проблемной области, которые они скрупулезно собирали и совершенствовали. Контроль версий защищает исходный код от катастрофических сбоев, от случайных ухудшений, вызванных человеческим фактором, а также от непредвиденных последствий.

Группы разработчиков программного обеспечения, не использующие какую-либо форму управления версиями, часто сталкиваются с такими проблемами, как незнание об изменениях, выполненных для пользователей, или создание в двух несвязанных частях работы изменений, которые оказываются несовместимыми и которые затем приходится скрупулезно распутывать и перерабатывать. Если вы как разработчик ранее никогда не применяли управление версиями, возможно, вы указывали версии своих файлов, добавляя суффиксы типа «финальный» или «последний», а позже появлялась новая финальная версия. Возможно, вы использовали комментирование блоков кода, когда хотели отключить определенные возможности, не удаляя их, так как опасались, что этот код может понадобиться позже. Решением всех подобных проблем является управление версиями.

4.2 Преимущества систем контроля версий

Программное обеспечение контроля версий рекомендуется для продуктивных команд разработчиков и команд DevOps. Управление версиями помогает отдельным разработчикам работать быстрее, а командам по разработке ПО — сохранять эффективность и гибкость по мере увеличения числа разработчиков.

За последние несколько десятилетий системы контроля версий (Version Control Systems, VCS) стали гораздо более совершенными, причем некоторым это удалось лучше других. Системы VCS иногда называют инструментами SCM (управления исходным кодом) или RCS (системой управления редакциями). Один из наиболее популярных на сегодняшний день инструментов VCS называется Git. Git относится к категории распределенных систем контроля версий, известных как DVCS (эта тема будет рассмотрена подробнее чуть позже). Git, как и многие другие популярные и доступные на сегодняшний день системы VCS, распространяется бесплатно и имеет открытый исходный код. Независимо от того, какую систему контроля версий вы используете и как она называется, основные ее преимущества заключаются в следующем.

1. Полная история изменений каждого файла за длительный период. Это касается всех изменений, внесенных огромным количеством людей за долгие годы. Изменением считается создание и удаление файлов, а также редактирование их содержимого. Различные инструменты VCS отличаются тем, насколько хорошо они обрабатывают операции переименования и перемещения файлов. В историю также должны входить сведения об авторе, дата и комментарий с описанием цели каждого изменения. Наличие полной истории позволяет возвращаться к предыдущим версиям, чтобы проводить анализ основных причин возникновения ошибок и устранять проблемы в старых версиях программного обеспечения. Если над программным обеспечением ведется активная работа, то «старой версией» можно считать почти весь код этого ПО.

2. Ветвление и слияние. Эти возможности полезны не только при одновременной работе участников команды: отдельные сотрудники также могут пользоваться ими, занимаясь несколькими независимыми направлениями. Создание «веток» в инструментах VCS позволяет иметь несколько независимых друг от друга направлений разработки, а также выполнять их слияние, чтобы инженеры могли проверить, что изменения, внесенные в каждую из веток, не конфликтуют друг с другом. Многие команды разработчиков ПО создают отдельные ветки для каждой функциональной возможности, для каждого релиза либо и для того, и для другого. Имея множество различных рабочих процессов, команды могут выбирать подходящий для них способ ветвления и слияния в VCS.

3. Отслеживаемость. Возможность отслеживать каждое изменение, внесенное в программное обеспечение, и связывать его с ПО для управления проектами и отслеживания багов, например Jira, а также оставлять к каждому изменению комментарий с описанием цели и назначения изменения может помочь не только при анализе основных причин возникновения ошибок, но и при других операциях по исследованию. История с комментариями во время чтения кода помогает понять, для чего этот код нужен и почему он структурирован именно так. Благодаря этому разработчики могут вносить корректные и совместимые изменения в соответствии с долгосрочным планом разработки системы. Это особенно важно для эффективной работы с унаследованным кодом, поскольку дает специалистам возможность точнее оценить объем дальнейших задач.