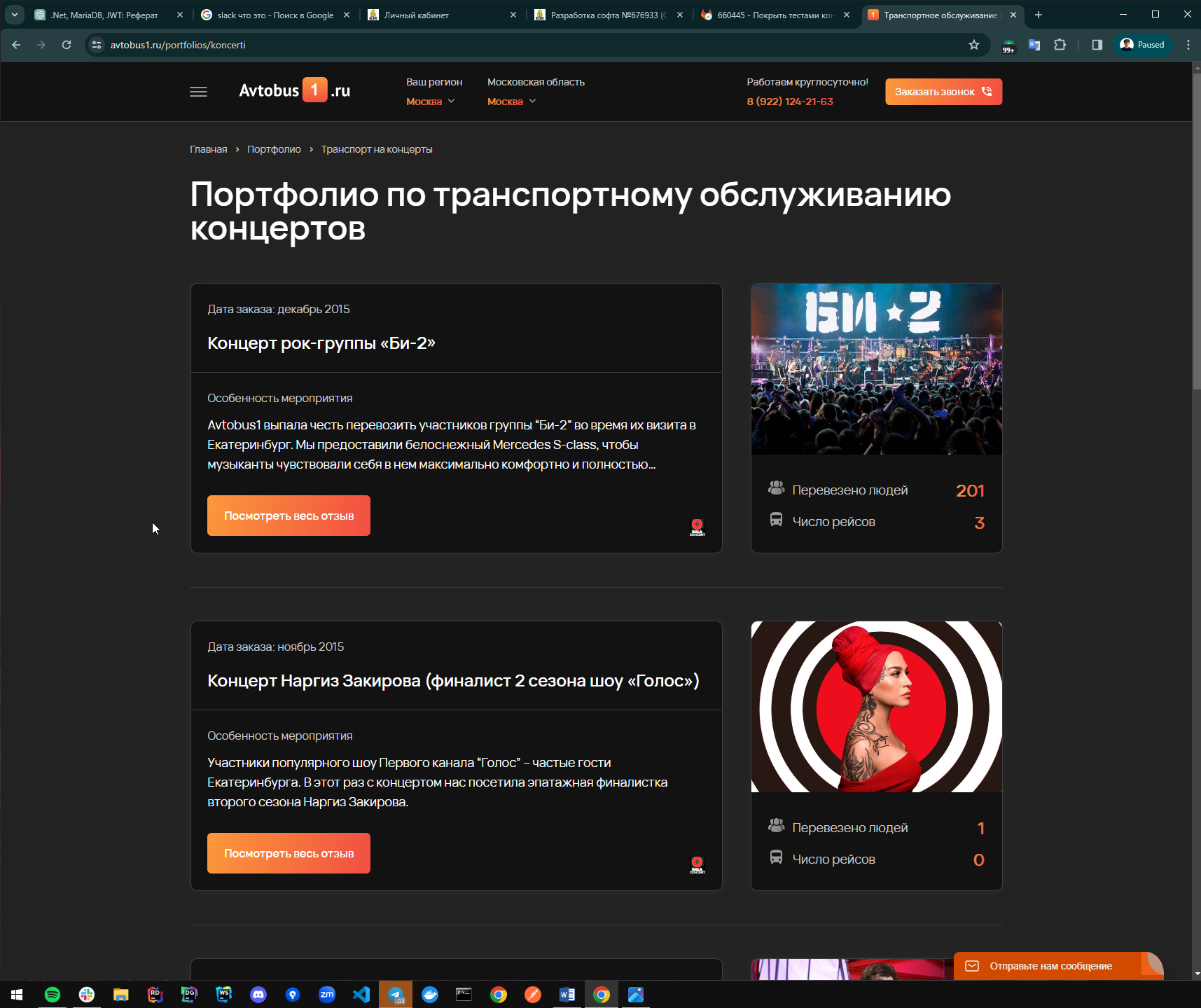
Я работаю .net разработчиком в Российской компании Автобус1. Это крупнейшая транспортная компания в России, до 2019 года компания осуществляла перевозки по всему миру, даже в Австралии было несколько филиалов, но сейчас только в России.

Вот страница с портфолио компании(я сделал тут вывод фото и отображение числа рейсов)



Каждое утро я встаю с четким пониманием, что меня ждет новый день в мире программирования, и с полным стремлением использовать его на полную катушку. Мой рабочий день начинается с жеста, который, хоть и незаметен для многих, но имеет огромное значение для меня: в 9:00 утра я нажимаю на кнопку "Пришел на работу" в виртуальном офисе. Это мое первое подтверждение готовности к новым вызовам и задачам, которые ждут меня сегодня.

После этого, в 9:05 утра, я получаю сообщение от бота в Slack, напоминающее мне о моем ежедневном планировании. Это момент, когда я взаимодействую с моими коллегами, обсуждаю текущие задачи и определяю приоритеты на день. Этот обмен информацией не только помогает мне лучше структурировать свою работу, но и создает общее понимание и синхронизацию в команде.

Хотя мой официальный рабочий день длится с 9:00 до 18:00, я имею гибкость в распределении своего времени. В мире программирования нет жестких рамок, и я могу работать в те моменты, когда мое внимание и продуктивность наивысши. Это позволяет мне достичь оптимальных результатов в своей работе и эффективно управлять своим временем.

Часто я нахожу себя за работой даже в выходные, вглубляясь в сложные задачи или углубляясь в новые технологии. Это не обязательство, а скорее стремление к самосовершенствованию и расширению знаний в области программирования.

Таким образом, моя рабочая рутина - это не просто выполнение задач и обязанностей, это постоянное стремление к развитию и совершенствованию в мире программирования. Это искусство находить баланс между творчеством и техническими навыками, между индивидуальной работой и коллективным сотрудничеством. Это искусство быть .NET разработчиком.

Каждый вторник в 11:00 наступает момент, который я жду с нетерпением - наше еженедельное совещание с командой. Это время не только для обсуждения текущих задач и проектов, но и для поддержания связи и сотрудничества с коллегами.

Когда наступает 11:00, я приготавливаюсь к совещанию, переключаясь на режим "приема информации". Это час, когда я могу временно отложить свои текущие задачи и сфокусироваться на обсуждении важных вопросов с командой. В это время я слушаю внимательно, делаю заметки и, если возникают вопросы или замечания, записываю их для последующего обсуждения.

Когда приходит моя очередь высказаться, я с удовольствием сообщаю, что у меня нет вопросов, зная, что мои коллеги и руководитель оценят мой активный участие в обсуждении. После совещания мой руководитель имеет возможность взглянуть на мои недельные задачи и прогресс через нашу систему управления предприятием (ERP), что помогает ему оставаться в курсе и принимать информированные решения.

Этот час вторника - не просто промежуток времени, когда нужно "ничего не делать", это возможность для обмена идеями, взаимодействия с командой и совместной работы над общими целями. Как .NET разработчик, я понимаю, что успешное сотрудничество с коллегами и прозрачная коммуникация являются ключевыми компонентами успешного выполнения проектов.

В рамках моей работы я занимаюсь поддержкой и разработкой для двух основных проектов в нашей компании. Первый из них - наша ERP система, или бекофис, который играет ключевую роль в управлении бизнесом и операционными процессами. Этот проект сопровождает нас уже вот шесть лет и является неотъемлемой частью нашей деятельности. Моя работа над этим проектом включает в себя решение проблем, улучшение функционала и обеспечение его стабильной работы.

Второй проект - это разработка и поддержка веб-сайтов, предоставляющих различные услуги. В этом направлении моей основной задачей является обеспечение непрерывной работы и дальнейшего развития восьми веб-сайтов, включая ключевой проект autobus1. Я активно занимаюсь ремонтом, добавлением нового функционала и обеспечением безопасности этих веб-сайтов

Несмотря на то, что я нахожусь в отделе бекенда, моя работа также распространяется на фронтенд. Я считаю себя фулл-стек разработчиком, потому что у меня есть опыт работы как с бекендом, так и с фронтендом. Это позволяет мне более глубоко понимать всю систему разработки, а также более эффективно взаимодействовать с другими членами команды и принимать участие в различных аспектах проекта.

Моя работа требует от меня не только технических навыков, но и способности анализировать, решать проблемы и эффективно коммуницировать с коллегами. Я горжусь своей ролью в обоих проектах и стремлюсь к их дальнейшему успеху и развитию.

Как корпоративный мессенджер, я использую Slack для обеспечения эффективной коммуникации и сотрудничества внутри команды и с коллегами по всей компании. Slack предоставляет широкий спектр возможностей для организации общения, обмена файлами, планирования задач и даже автоматизации рутиных процессов.

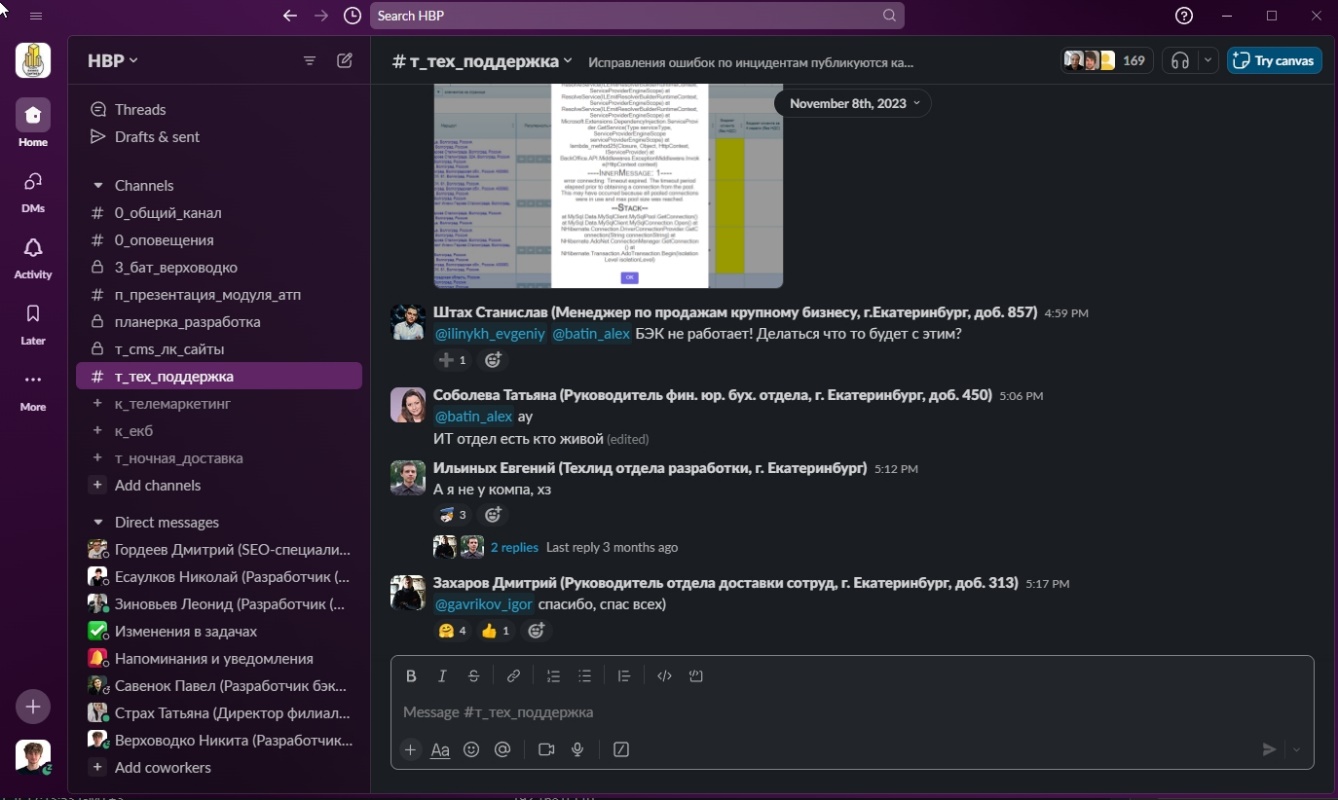
В Slack я создаю различные каналы для обсуждения разных проектов и тем, что позволяет поддерживать структурированную коммуникацию внутри команды. Мы также используем личные сообщения для частных обсуждений и быстрой передачи информации.

Одной из важных возможностей Slack является интеграция с другими сервисами и инструментами, которые мы используем в работе. Мы можем легко получать уведомления о изменениях в наших проектах из систем управления версиями кода или задачами, а также быстро обмениваться файлами из наших хранилищ данных.

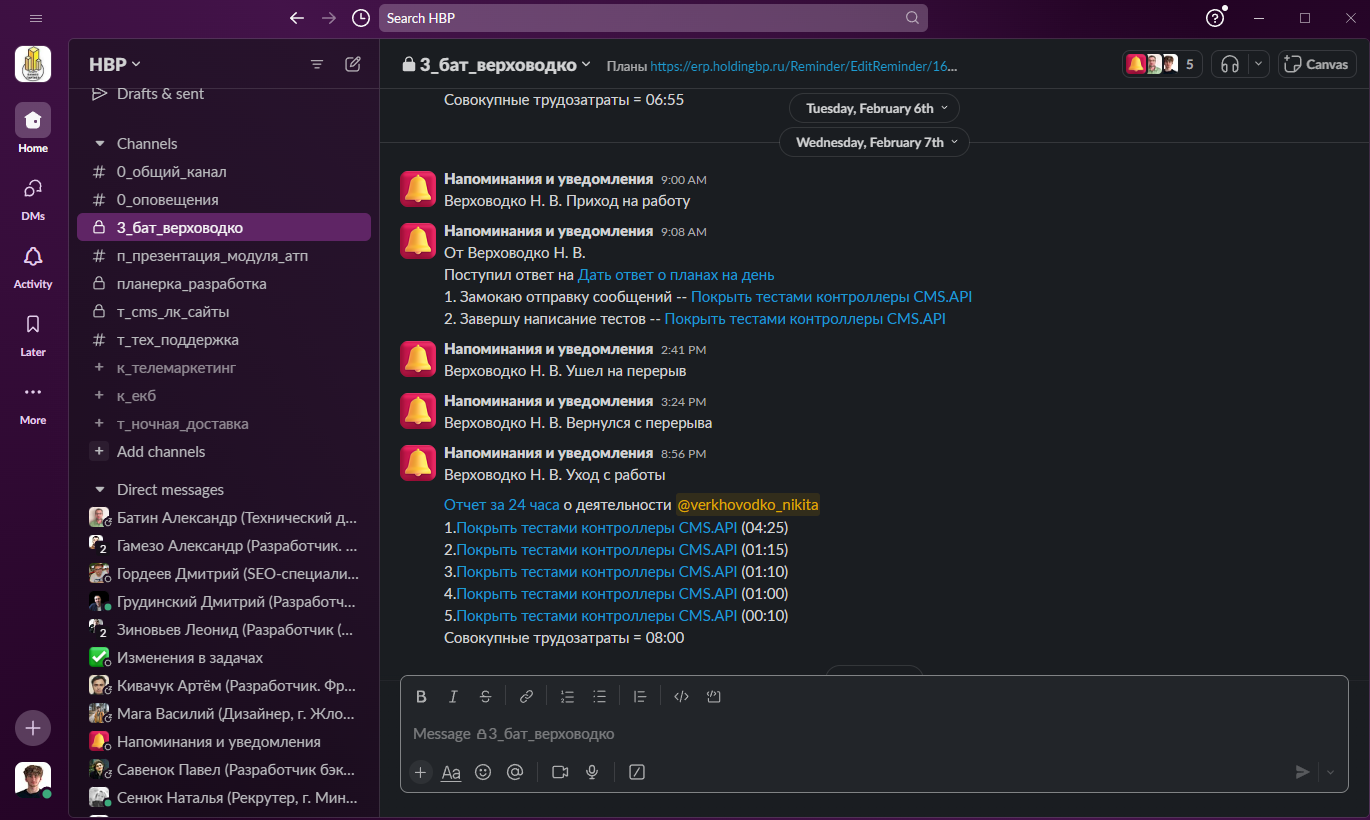
Кроме того, Slack поддерживает использование ботов и приложений, что позволяет автоматизировать многие рутиные задачи и упрощает нашу работу. Например, мы можем использовать ботов для управления задачами, планирования встреч или даже для отправки уведомлений о статусе наших систем.

В целом, использование Slack в качестве корпоративного мессенджера позволяет нам эффективно сотрудничать, обмениваться информацией и оставаться в курсе всех происходящих событий в нашей компании.

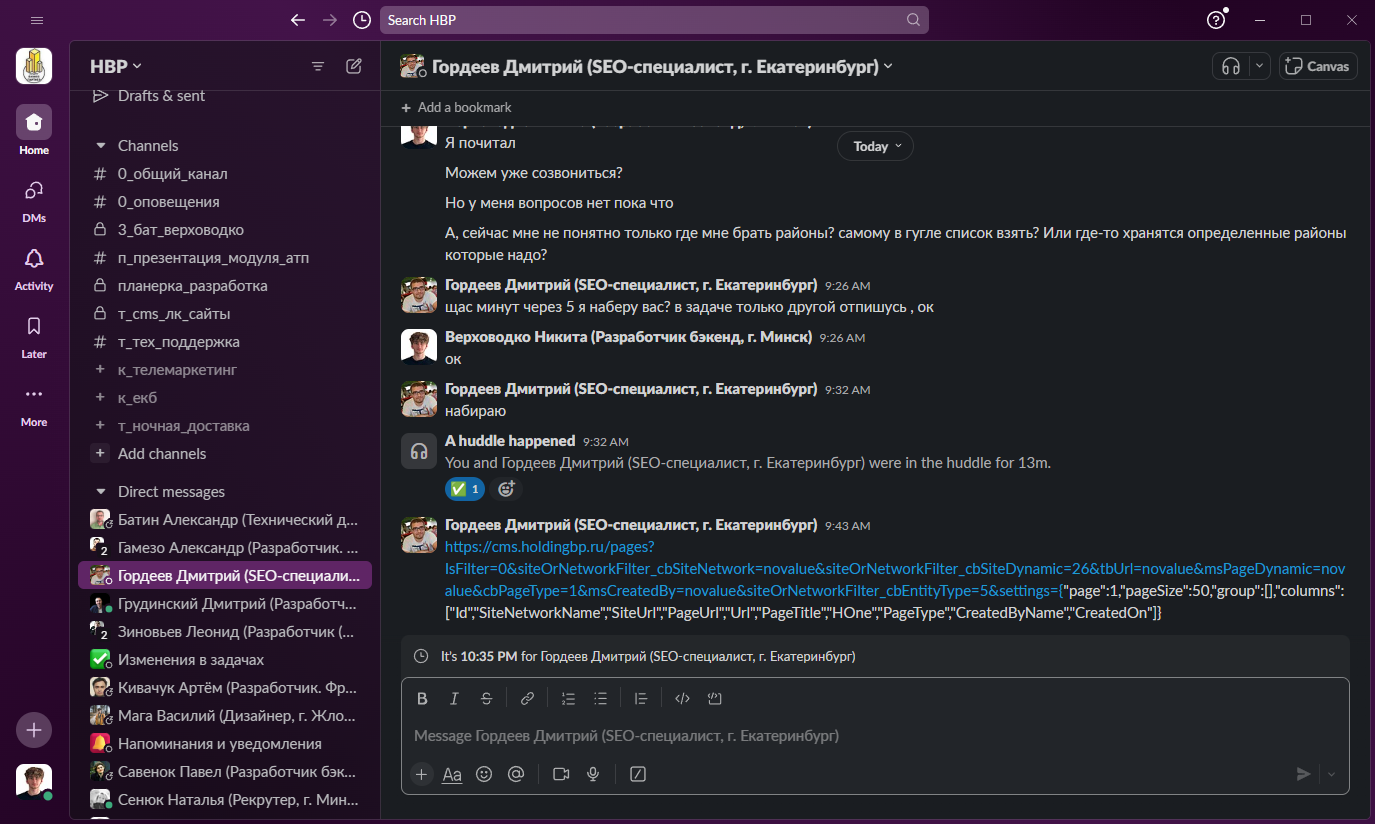
У нас есть общий канал где ведутся общие обсуждения, но в основном туда летит спам для ИТ отдела о том что система не работает, мы конечно же молниеносно реагируем.



Для общения с руководителем я использую наш канал, где есть: гениральный директор, техлид и технический директор, туда приходят все уведомления о моей деятельности



Для решения задач с постановщиками мы используем личные сообщения



Бекенд у нас на .NET - это платформа для разработки и запуска приложений, созданная компанией Microsoft. Она включает в себя языки программирования, библиотеки, инструменты и среду выполнения, которые позволяют разработчикам создавать разнообразные типы приложений - от веб-сайтов и мобильных приложений до корпоративных решений и игр.

Основные компоненты .NET включают в себя:

На сегодняшний момент язык программирования C# один из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. В настоящий момент на нем пишутся самые различные приложения: от небольших десктопных программок до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей.

C# уже не молодой язык и как и вся платформа .NET уже прошел большой путь. Первая версия языка вышла вместе с релизом Microsoft Visual Studio .NET в феврале 2002 года. Текущей версией языка является версия C# 12, которая вышла 14 ноября 2023 года вместе с релизом .NET 8.

C# является языком с Си-подобным синтаксисом и близок в этом отношении к C++ и Java. Поэтому, если вы знакомы с одним из этих языков, то овладеть C# будет легче.

C# является объектно-ориентированным и в этом плане много перенял у Java и С++. Например, C# поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений. И C# продолжает активно развиваться, и с каждой новой версией появляется все больше интересных функциональностей.

Роль платформы .NET

Когда говорят C#, нередко имеют в виду технологии платформы .NET (Windows Forms, WPF, ASP.NET, .NET MAUI). И, наоборот, когда говорят .NET, нередко имеют в виду C#. Однако, хотя эти понятия связаны, отождествлять их неверно. Язык C# был создан специально для работы с фреймворком .NET, однако само понятие .NET несколько шире.

Как-то Билл Гейтс сказал, что платформа .NET - это лучшее, что создала компания Microsoft. Возможно, он был прав. Фреймворк .NET представляет мощную платформу для создания приложений. Можно выделить следующие ее основные черты:

Поддержка нескольких языков. Основой платформы является общеязыковая среда исполнения Common Language Runtime (CLR), благодаря чему .NET поддерживает несколько языков: наряду с C# это также VB.NET, C++, F#, а также различные диалекты других языков, привязанные к .NET, например, Delphi.NET. При компиляции код на любом из этих языков компилируется в сборку на общем языке CIL (Common Intermediate Language) - своего рода ассемблер платформы .NET. Поэтому при определенных условиях мы можем сделать отдельные модули одного приложения на отдельных языках.

Кроссплатформенность. .NET является переносимой платформой (с некоторыми ограничениями). Например, последняя версия платформы на данный момент - .NET 8 поддерживается на большинстве современных ОС Windows, MacOS, Linux. Используя различные технологии на платформе .NET, можно разрабатывать приложения на языке C# для самых разных платформ - Windows, MacOS, Linux, Android, iOS, Tizen.

Мощная библиотека классов. .NET представляет единую для всех поддерживаемых языков библиотеку классов. И какое бы приложение мы не собирались писать на C# - текстовый редактор, чат или сложный веб-сайт - так или иначе мы задействуем библиотеку классов .NET.

Разнообразие технологий. Общеязыковая среда исполнения CLR и базовая библиотека классов являются основой для целого стека технологий, которые разработчики могут задействовать при построении тех или иных приложений. Например, для работы с базами данных в этом стеке технологий предназначена технология ADO.NET и Entity Framework Core. Для построения графических приложений с богатым насыщенным интерфейсом - технология WPF и WinUI, для создания более простых графических приложений - Windows Forms. Для разработки кроссплатформенных мобильных и десктопных приложений - Xamarin/MAUI. Для создания веб-сайтов и веб-приложений - ASP.NET и т.д.

К этому стоит добавить активной развивающийся и набирающий популяность Blazor - фреймворк, который работает поверх .NET и который позволяет создавать веб-приложения как на стороне сервера, так и на стороне клиента. А в будущем будет поддерживать создание мобильных приложений и, возможно, десктоп-приложений.

Производительность. Согласно ряду тестов веб-приложения на .NET в ряде категорий сильно опережают веб-приложения, построенные с помощью других технологий. Приложения на .NET в принципе отличаются высокой производительностью.

Также еще следует отметить такую особенность языка C# и фреймворка .NET, как автоматическая сборка мусора. А это значит, что нам в большинстве случаев не придется, в отличие от С++, заботиться об освобождении памяти. Вышеупомянутая общеязыковая среда CLR сама вызовет сборщик мусора и очистит память.

.NET Framework и .NET 8

Стоит отметить, что .NET долгое время развивался премущественно как платформа для Windows под названием .NET Framework. В 2019 вышла последняя версия этой платформы - .NET Framework 4.8. Она больше не развивается

С 2014 Microsoft стал развивать альтернативную платформу - .NET Core, которая уже предназначалась для разных платформ и должна была вобрать в себя все возможности устаревшего .NET Framework и добавить новую функциональность. Затем Microsoft последовательно выпустил ряд версий этой платформы: .NET Core 1, .NET Core 2, .NET Core 3, .NET 5. И текущей версией является расматриваемая в этом руководстве платформа .NET 8. Поэтому следует различать .NET Framework, который предназначен преимущественно для Windows, и кросплатформенный .NET 8. В данном руководстве речь будет идти о C# 12 в связке с .NET 8, поскольку это актуальная платформа.

Управляемый и неуправляемый код

Нередко приложение, созданное на C#, называют управляемым кодом (managed code). Что это значит? А это значит, что данное приложение создано на основе платформы .NET и поэтому управляется общеязыковой средой CLR, которая загружает приложение и при необходимости очищает память. Но есть также приложения, например, созданные на языке С++, которые компилируются не в общий язык CIL, как C#, VB.NET или F#, а в обычный машинный код. В этом случае .NET не управляет приложением.

В то же время платформа .NET предоставляет возможности для взаимодействия с неуправляемым кодом..

JIT-компиляция

Как выше писалось, код на C# компилируется в приложения или сборки с расширениями exe или dll на языке CIL. Далее при запуске на выполнение подобного приложения происходит JIT-компиляция (Just-In-Time) в машинный код, который затем выполняется. При этом, поскольку наше приложение может быть большим и содержать кучу инструкций, в текущий момент времени будет компилироваться лишь та часть приложения, к которой непосредственно идет обращение. Если мы обратимся к другой части кода, то она будет скомпилирована из CIL в машинный код. При том уже скомпилированная часть приложения сохраняется до завершения работы программы. В итоге это повышает производительность.

По сути это все, что вкратце надо знать о платформе .NET и языке C#. А теперь создадим первое приложение.

Фронтенд на "чистом" JavaScript (JS) подразумевает создание пользовательского интерфейса веб-приложения без использования сторонних библиотек или фреймворков, таких как React, Angular или Vue.js. В этом случае разработчики полностью основываются на нативных возможностях JavaScript для создания интерактивных и отзывчивых веб-приложений.

Вот несколько ключевых аспектов и особенностей разработки фронтенда на "чистом" JavaScript:

DOM (Document Object Model): JavaScript использует DOM для взаимодействия с элементами веб-страницы. Разработчики манипулируют DOM-деревом, изменяя его структуру, стили и содержимое для создания интерактивного пользовательского интерфейса.

Событийная модель: JavaScript использует событийную модель для обработки пользовательских действий, таких как клики мышью, нажатия клавиш, отправка форм и другие. Разработчики могут назначать обработчики событий для элементов DOM и реагировать на действия пользователя

AJAX (Asynchronous JavaScript and XML): AJAX позволяет взаимодействовать с сервером асинхронно, без перезагрузки всей страницы. Это позволяет создавать более динамичные и отзывчивые веб-приложения, загружая и обновляя данные на странице без задержек.

Манипуляция данными: JavaScript используется для обработки и манипуляции данными на стороне клиента. Это включает в себя валидацию форм, отправку данных на сервер, обновление информации на странице и другие операции.

Модули и ES6: С появлением стандарта ECMAScript 6 (ES6) в JavaScript появились новые возможности, такие как стрелочные функции, классы, модули и другие. Разработчики могут использовать эти функции для улучшения читаемости, поддерживаемости и масштабируемости своего кода.

Отладка и тестирование: Для разработки на "чистом" JavaScript разработчики могут использовать встроенные в браузер инструменты для отладки кода и тестирования приложений. Это включает в себя консоль разработчика, инструменты для анализа производительности и другие.

Хотя фронтенд на "чистом" JavaScript может быть более сложным и требовательным к коду, чем использование фреймворков, это также предоставляет разработчикам большую гибкость и контроль над их приложениями.

MariaDB - это мощная реляционная система управления базами данных (СУБД), которая является форком MySQL и предоставляет высокую производительность, надежность и масштабируемость для различных приложений. Мы используем MariaDB в нашей компании как основную базу данных для хранения и управления данными.

Преимущества MariaDB:

Производительность: MariaDB обеспечивает высокую производительность благодаря оптимизации запросов, использованию индексов и поддержке параллельной обработки запросов.

Надежность: MariaDB предлагает высокую степень надежности и устойчивости благодаря механизмам репликации данных, резервного копирования и восстановления после сбоев.

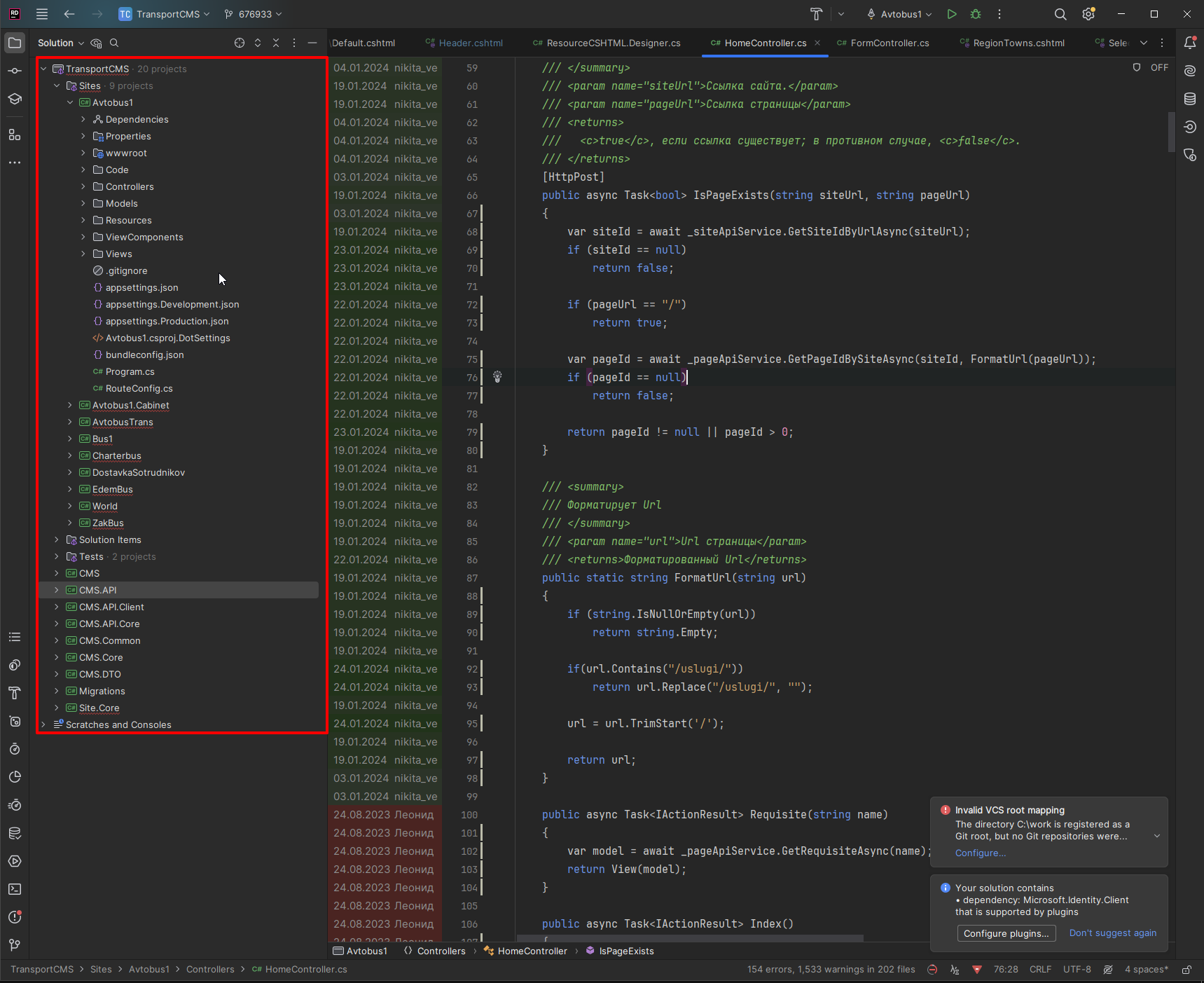
Масштабируемость: MariaDB позволяет горизонтальное и вертикальное масштабирование, что позволяет эффективно управлять ростом данных и нагрузкой на базу данных.

Открытый исходный код: MariaDB является проектом с открытым исходным кодом, что обеспечивает гибкость в настройке и расширении функционала в соответствии с потребностями нашей компании.

Совместимость с MySQL: MariaDB совместима с MySQL и предоставляет поддержку большинства команд и функций, используемых в MySQL, что облегчает миграцию существующих приложений на MariaDB

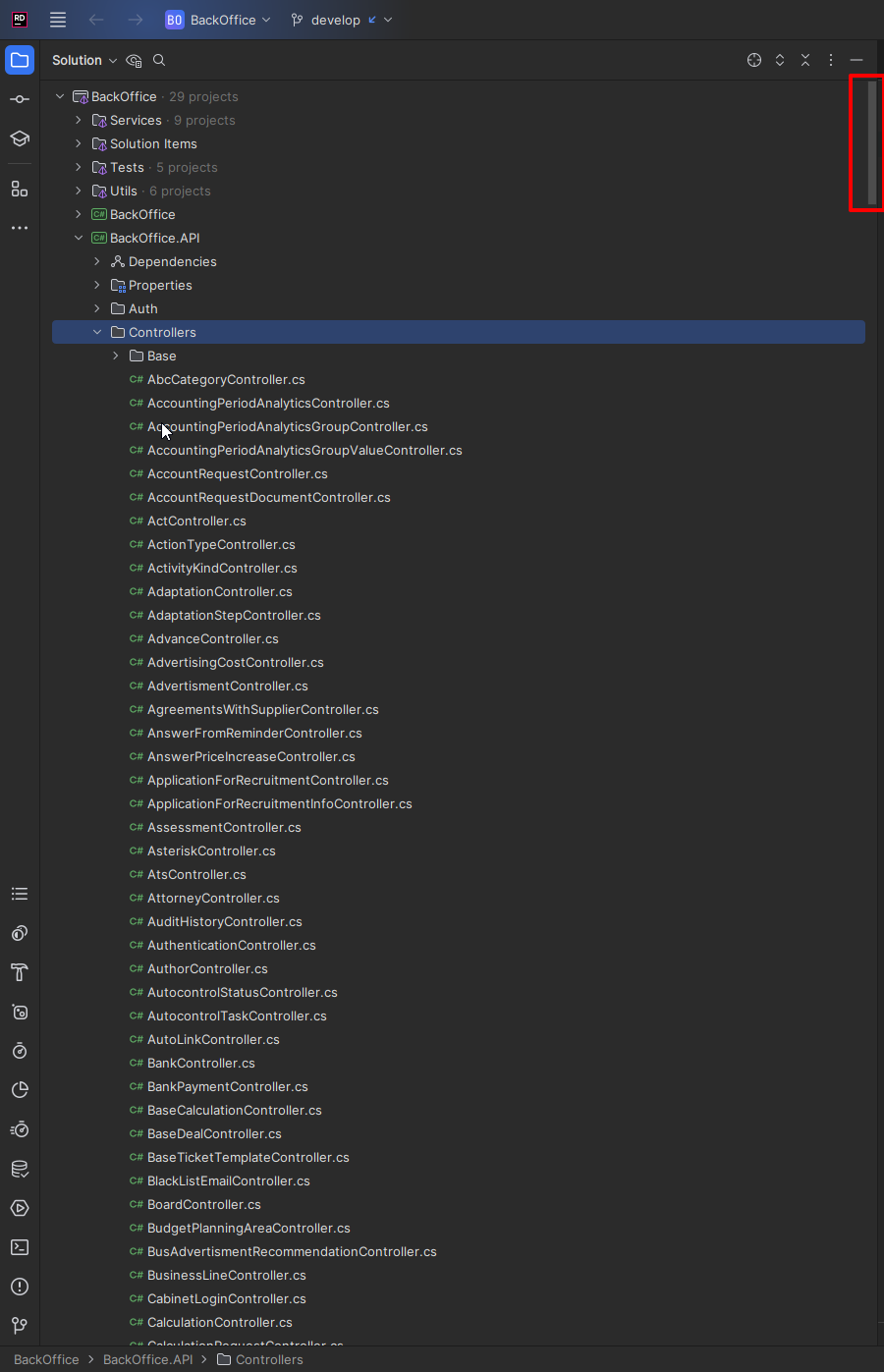
Активное сообщество: MariaDB имеет активное сообщество разработчиков и пользователей, которые постоянно работают над улучшением и развитием системы.

Использование MariaDB в нашей компании позволяет нам эффективно управлять данными, обеспечивая высокую производительность, надежность и масштабируемость наших приложений и сервисов.

Архитектура проекта MVC   


Для запуска сразу надо запустить CMS.API, после чего через 2-3 минуты запустить любой другой проект, который нужен.

Для понимая масштабности проекта BackOffice:



Это только часть контроллеров(А-С) буквы алфавита.

Для работы с git мы используем GitLab

Как использовать Git для работы с командой

Узнайте, как эффективно использовать мощный инструмент Git для работы в команде и управления кодом с примерами и советами от эксперта.

Git является мощным инструментом для работы с командой и управления кодом. В этой статье мы разберем основные принципы и практики, которые помогут вам эффективно использовать Git при работе в команде.

Основы Git

Прежде всего, необходимо установить и настроить Git на вашем компьютере. Затем создайте репозиторий для вашего проекта и добавьте его на сервер (например, на GitHub или GitLab).

Следующим шагом будет клонирование репозитория на локальный компьютер каждого участника команды. Это позволит им вносить изменения в код и отправлять их на сервер.

Ветки в Git

Одной из ключевых особенностей Git является работа с ветками. Ветки позволяют разрабатывать новые функции и исправлять ошибки параллельно с основной линией разработки.

Создавайте новые ветки для каждой задачи. Например, если вы работаете над функцией «авторизация», создайте ветку feature/auth.

Чтобы создать новую ветку, выполните следующую команду:

git checkout -b feature/auth

После завершения работы над задачей, необходимо объединить ветку с основной линией разработки (обычно это ветка master или main).

Коммиты

Коммиты — это «снимки» ваших изменений. При работе над задачей, делайте коммиты часто и описывайте их содержание. Это поможет вам и вашей команде легче разбираться в истории изменений.

Помните, что хороший коммит должен содержать только одно логическое изменение.

Чтобы сделать коммит, выполните следующие команды:

git add .

git commit -m "Описание изменений"

Слияние и конфликты

При слиянии двух веток могут возникнуть конфликты. Конфликты возникают, когда два разработчика изменяют одну и ту же строку кода.

Как правило, Git автоматически решает конфликты, сливая изменения. Однако иногда вам придется вручную разрешить конфликт, указав, какие изменения должны быть сохранены.

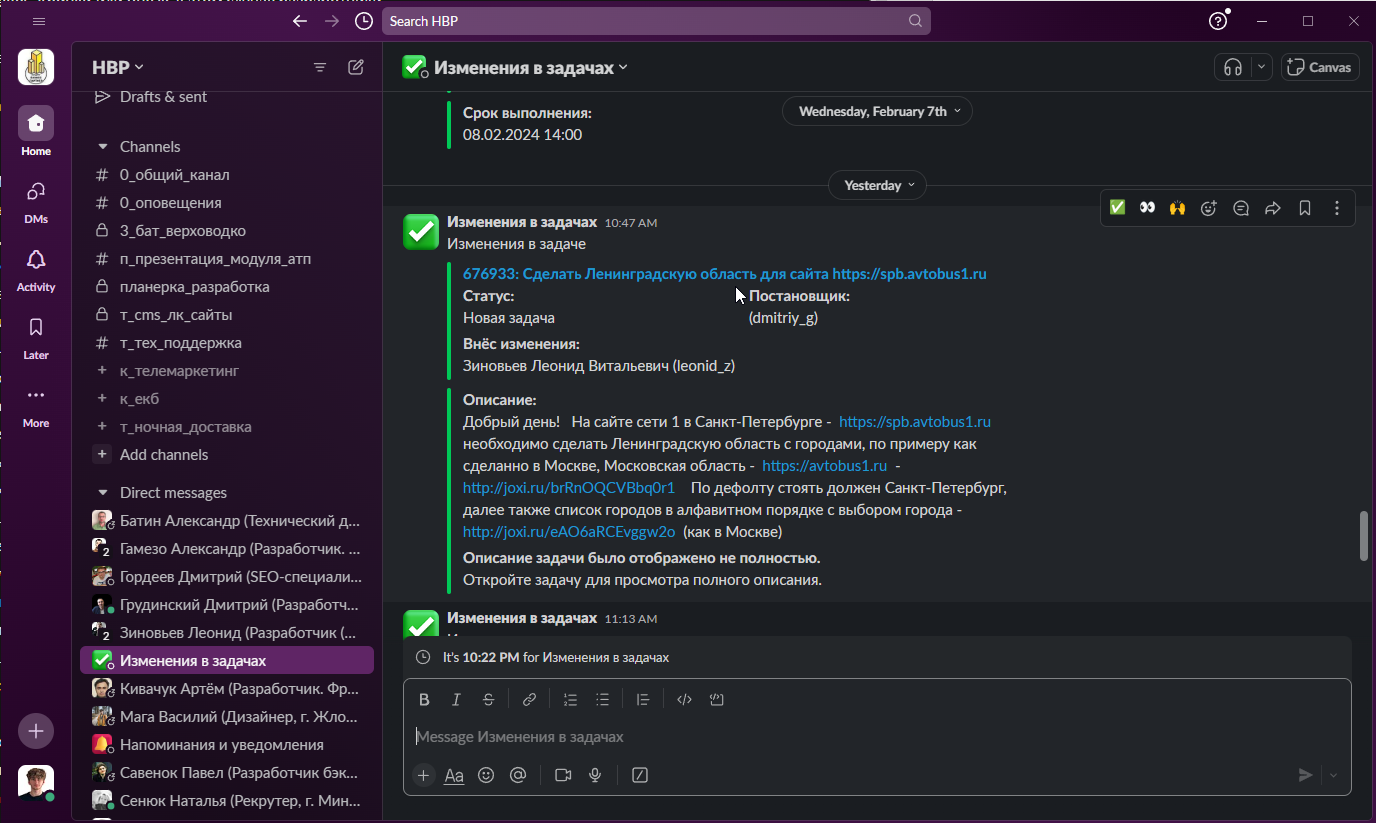
Pull requests

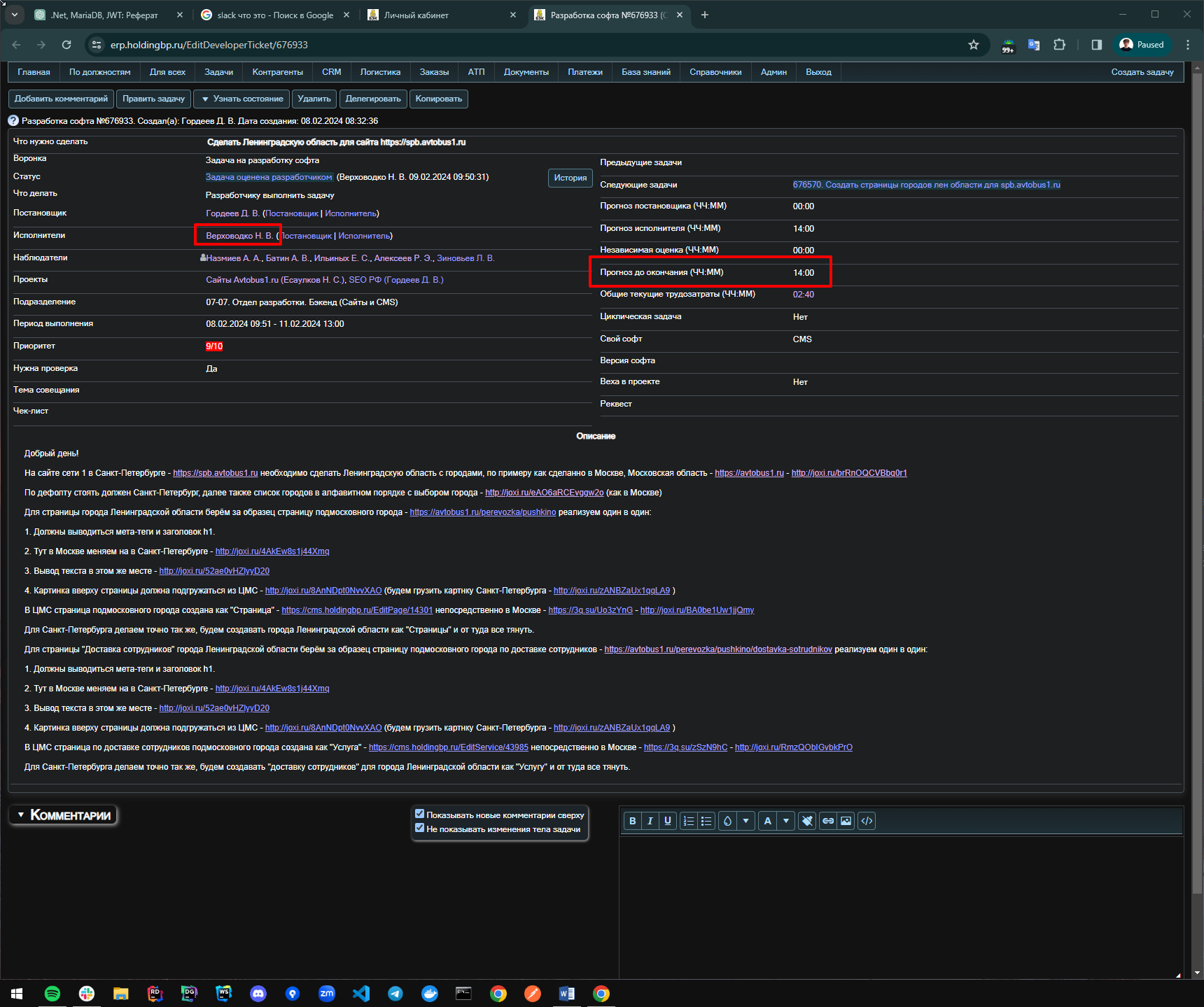
Pull request (PR) — это запрос на слияние вашей ветки с основной линией разработки. PR позволяет вашей команде проверить изменения перед их интеграцией.

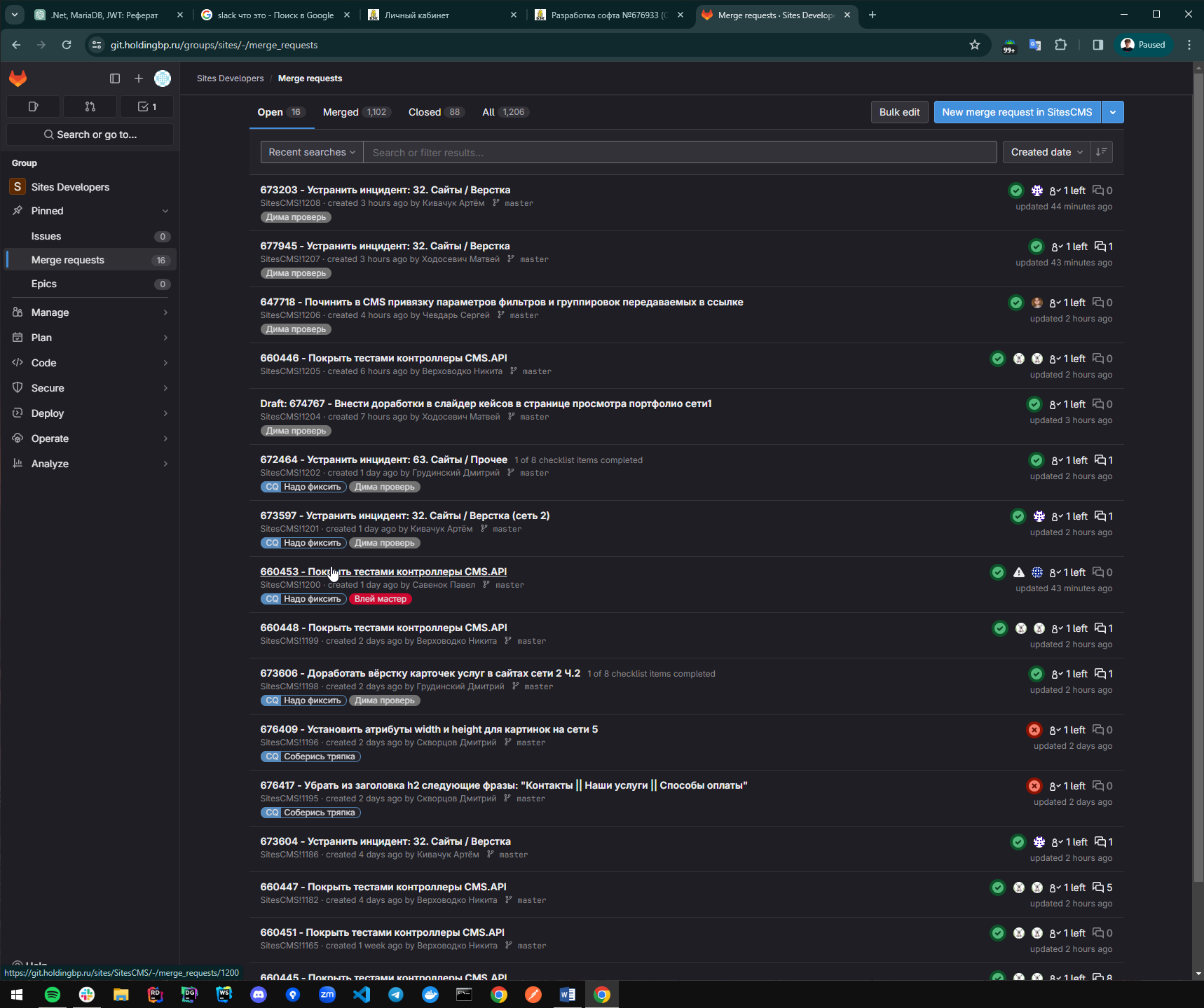
Чтобы создать PR, отправьте вашу ветку на сервер и создайте запрос на слияние через интерфейс Git-сервера (например, GitHub или GitLab).

Давайте разберем как происходит моя работа с одной задачей.

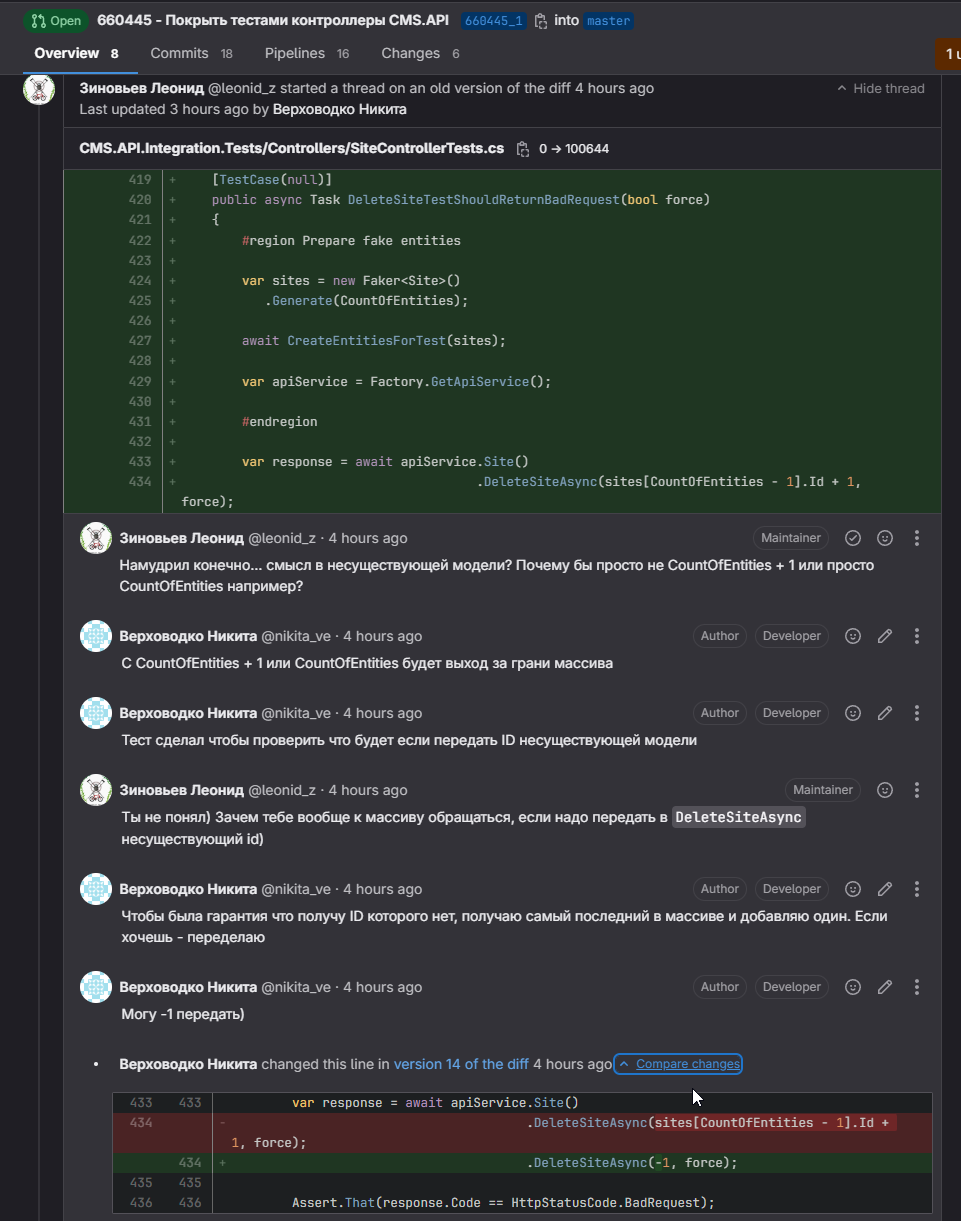
1. Я получаю уведомление в Slack о том что на меня поставили задачу.



1. Когда у меня будет время и желание – захожу и оцениваю ее, выставляю прогноз до окончания  
   
2. И по ходу выполнения задач я пишу комментарии о проделанной работе и указываю трудозатраты
3. Когда все готово делаю Merge Request



После чего получаю комментарии от ведущего разработчика, с предложениями об улучшении кода



После того как я сделал реквест бот переводит задачу в статус Pull-реквест создан исполнителем.

1. Код проходит тесты, у нас настроен GitLab CI/CD. Первая часть тестов это сброрка каждого проекта в решении, вторая – прохождение тестов, которые я и мой коллега писали пол месяца.

