Служебная записка от 31.11.2023

Инженер-программист Нижегородов Олег Эдуардович

Учитывая должностную инструкцию инженера-программиста отдела программирования я занимаюсь исследовательской работой и изучением современных тенденций и методов в программной инженерии.

Со своей стороны я уже в этом году изучил например асинхронные методы работы, позволяющие работать мне как программисту на достаточно высоком уровне. Ранее мной была написана программа для обогащения БД номерами ОМС, но там использовался метод используя BackgroundWorker класс . Сейчас используется современный метод async/await (шаблон работы с задачами TPL), позволяющий работать в Windows десктопной программе работать как в веббраузере. Причем в программе нет никакой адресной строки – и это безопасность. В разработке асинхронные методы работы ПО актуальны и современны, поскольку операции по сети происходят в основном не моментально, а требуют некоторого времени. Задержка для пользователя не должна быть исключительной ситуацией и должна обрабатываться не нарушая в целом работу программы. Тем более я сейчас повышаю квалификацию (Skillfactory – C# Developer. Также современная платформа NetCore (dotNet) от Microsoft позволяет делать кроссплатформенные высокоэффективные приложения. Начиная с разработки БД ,используя методы EF CORE, программист разрабатывает БД с помощью модели, идентифицирующей ПО. И это достаточно удобно, потому как здесь также поддерживается модификация схемы БД. Например, есть начальные сниппеты по созданию БД на MSSQL/Postgres в NetCore 2.1, 3.1. И это требует дальнейшего и углубленного изучения для эффективности в разработке.

В информационно технологии наиважнейшую роль играет развитие. Старт может быть начат с маленького кусочка кода и, учитывая множество точек приложения вылиться в достаточно эффективное средство при грамотном руководстве процессом развития разработки.

Платформа NetCore позволяет делать API –и это бекэнд. Роль серверных процессов – отдавать информацию в том или ином виде, например формат JSON.

Платформа NetCore имеет развитую и беплатную IDE (VisualStudio). Эта IDE имеющую на своем борту уже изначально много шаблонов для старта разработки (в том числе API). Среди этих шаблонов присутствуют шаблоны для старта фронтенда на JavaScript. Дело в том что серверные процессы работают не одни сами посебе, а вместе с интерфейсными процессами – такие процессы работают обычно используя движок JavaScript, который встроен в вебброузер. Уже это сказанное дает право сказать об экономической целесообразности использования этой IDE. И это не говоря о технологической целесообразности использования платформы dotNet – ибо она вобрала в себя весь опыт использования предшественника – NetFramework. И это большая экосистема, делающая эффективным и современным процесс разработки OpenSource проектов. Кроме всего прочего в образовательном процессе эта IDE может быть достаточно весома ибо в ней наглядно можно изучить например работу с GIT (система версионного контроля).

Таким образом серверные процессы это бэкенд и может быть построены на C#, на том же JavaScript (Node.js) или на специально предназначенном PHP. Интерфейсные процессы вынужденно должны писаться на JavaScript. Кстати сказать если для серверного процесса используется PHP, то кириллица в юникоде в броузере идет не читаемая, а если используется JavaScript или dotNet – то кириллица читаемая. Исходя из этого можно уже в броузере тестировать API (методыGET), не привлекая специализированное ПО при условии что бекэнд написан не на PHP (для PHP обязательно нужно использовать JavaScript для обработки). По моим последним разработкам получается такая картина . Грязные, необработанные данные могут использованы для тестирования. По идее вообще данные по сети редаваются как поток байтов (после сериализации объекта) и чтоб принять данные пришедшую строку (данные) нужно десериализировать. Например в моей программе по работе с направлениями ЕЦП это и используется.

Также в VisualStudio имеются начальные сниппеты для старта на JavaScript, позволяющие делать интерфейсную часть приложения для серверных процессов. JavaScript программы пишутся в основном в другом редакторе, тоже бесплатном и также от Microsoft – Visual Studio Code. Этот редактор позволяет также тестировать-отлаживать интерфейсную часть в вебброузере, поскольку для интерфейса нужен серверный процесс – и этот редактор представляет встроенный сервер. Редактор имеет дружественный интерфейс c удобным кодированием. Этот редактор также может быть установлен в Linux.

Мне была поручена исследовательская работа по выяснению идентификационных элементов в программном коде dotNet и JavaScript.

Параллельно с этой работой я провел работу по выяснению возможностей такого стека в Linux системах, учитывая старые интересы и согласуясь с новыми технологическими тенденциями в современном программной инженерии ,в том числе для отечественного сегмента (Astra Linux, Alt Linux, Green Linux).

Выяснилось, что :

1. на Astra Linux нельзя в принципе установить Node.js выше версии 16 (проблема в зависимостях Си библиотек), тогда как по текущему моменту актуальна версия Node.js 20 (Windows 10 это обеспечивает). Хотя заявлено что Astra Linux – это замена Windows. Хочу отметить что Astra Linux в целом это хороший кандидат на замену Windows (удобный и быстрый рабочий стол, офис, БД, работа с сетью, серверные процессы, экономия памяти и т.п). Это касается и Alt Linux. Но вот по таким параметрам как разработка на JavaScript здесь имеется пробел. Причем например на Alt Linux 8 нельзя в принципе установить Node.js выше версии 8. Хотя бекэнд здесь и возможен (на Node.js, Express), но для разработки фронтенда этого явно недостаточно. Для успешной разработки фронтенда требуется Node.js выше 16 версии например для Vue3. Здесь уже приходится выбирать версию Alt Linux выше – 9 или 10.
2. Astra Linux построена на Debian Linux и по пакетам ПО совместима с Debian. Многие вещи можно установить из репозитория Debian. Успешно устанавливается dotNet. По умолчанию здесь устанавливается Node.js 8 причем без менеджера пакетов npm (а как без сторонних библиотек ? изобретать велосипед ?). Я успешно установил Node.js 16 (c npm) из пакета ПО скачанного с оригинального сайта.
3. Alt Linux использует пакеты ПО тоже известные но не Debian, но установщик как в Debian. В этом смысле Alt Linux это сильно индивидуальная система. В этом заключается сложность.

Объективности ради таким образом замена Windows не 100%.

Хочу заметить, такие исследования должны продолжаться. На это тратится достаточно много времени и сил. Это также достаточно глубоквая и знаниеемкая работа, помогающая отечественным разработчикам повышать качество разрабатываемых систем

Что касается Green Linux – это разработка от Яндекс. Благополучно устанавливается Node.js 20 версии без проблем. Достаточно современная система, правда занимает памяти в два раза больше Астры или Альт. Совместима с Убунту ибо на ней основана.

Мной собрана система программирования для JavaScript разработки на Убунту 20 linux. Причем я начал собирать ее с сервера и пришлось добавлять графическую систему отдельно – Гном 3, чтобы установить IDE VisualStudioCode. Рабочий стол достаточно быстрый. dotNet устанавливается без проблем, причем Windows 10 использует ту же Убунту как дополнение для работы с Linux. Виртуальная машина у меня занимает памяти немногим меньше по сравнению с отечественными Астра и Альт, и она хорошо поддерживается мировым сообществом.

На сегодняшний момент можно сделать вывод, что Убунту может быть использована для современной разработки на JavaScript. Кроме всего прочего для установки ПО можно использовать репозитории на Яндексе. Объективности ради, на Убунте нельзя установить VisualStudio c ее многочисленными плюшками. Таким образом это тоже не 100% замена Windows.

Опять же, объективности ради, думаю что для разработчика Windows и Linux могут прекрасно сосуществовать и дополнять друг друга. Доля диктуется исходя из условий и окружения, в котором работает задача разработки.