ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОНИКИ И МАТЕМАТИКИ

Отчёт
по курсовой работе бакалавров

студентов группы БИВ215
Абрамова Ильи Александровича, Чеканова Ивана Сергеевича

Москва, 2022

Оглавление

Аннотация

Объектом разработки данной курсовой работы является программный комплекс для отслеживания задач (т.н. таск-трекер). Цель работы — реализация программного решения, состоящего из двух независимых компонентов и позволяющего в удобном для пользователя формате вести работу по повышению личной и корпоративной эффективности. В ходе работы мы познакомились с языками Python и C#, научились создавать веб и десктопные интерфейсы, изучили методологии планирования. Результатом работы стало создание набора из двух программных решений: десктопного и веб-приложения. Результат работы уже может использоваться по прямому назначению. Также заложен большой задел для доработок системы.

Введение

Таск-трекер — это программа для отслеживания статусов задач, планирования работы, хранения идей. В наше время без таких решений не обходится ни одна компания, ведь всем нужен наиболее удобный инструмент управления проектами. Даже в МИЭМе в качестве корпоративного программного обеспечения, помогающего студентам организовать работу в проектах, используется опенсорс-решение «Taiga».

Наш проект — это курсовая работа, в первую очередь ставщая перед собой образовательную задачу. Поэтому, несмотря на то, что существует множество различных реализаций таск-трекеров, мы решили сделать ещё одну. В результате выполнения работы мы познакомились с языком Python, библиотекой для создания веб-сайтов Django, базой данных SQLite, а также языками HTML (фреймворк Bootstrap 5), CSS и JS, с помощью которых создается пользовательский интерфейс программы.

Задача планирования вставала перед людьми с самых ранних времен. Сначала человек мог хранить все идеи в голове. Со временем количество социальных функций и других дел увеличивалось. Люди стали вести дневники. В последнее столетие человечество осуществляет множество мегапроектов, каждый из которых требует грамотного планирования. В этой ситуации не обойтись без стройной и логичной системы менеджмента дел. Так появились различные методологии планирования. В качестве наиболее современных, популярных и широко применимых можно выделить японскую Kanban, зародившуюся в компании Toyota в далёком 1959 году и Getting Things Done, предложенную Дэвидом Алленом в одноимённой книге 2001 года.

Более подробно остановимся на второй системе. Gettings Thing Done за счёт своей простоты и гибкости подходит для работы над широчайшим кругом задач. Согласно автору, задачи должны быть разделены по

принципу, показанному на рис. 1. Основная идея системы гласит, что человек должен освободить свой разум от запоминания текущих задач, перенеся задачи и напоминания о них во внешнюю систему. Таким образом, разум человека, освобождённый от запоминания того, что должно быть сделано, может сконцентрироваться на выполнении самих задач, которые должны быть чётко определены и сформулированы заранее («какое следующее действие нужно совершить?»).

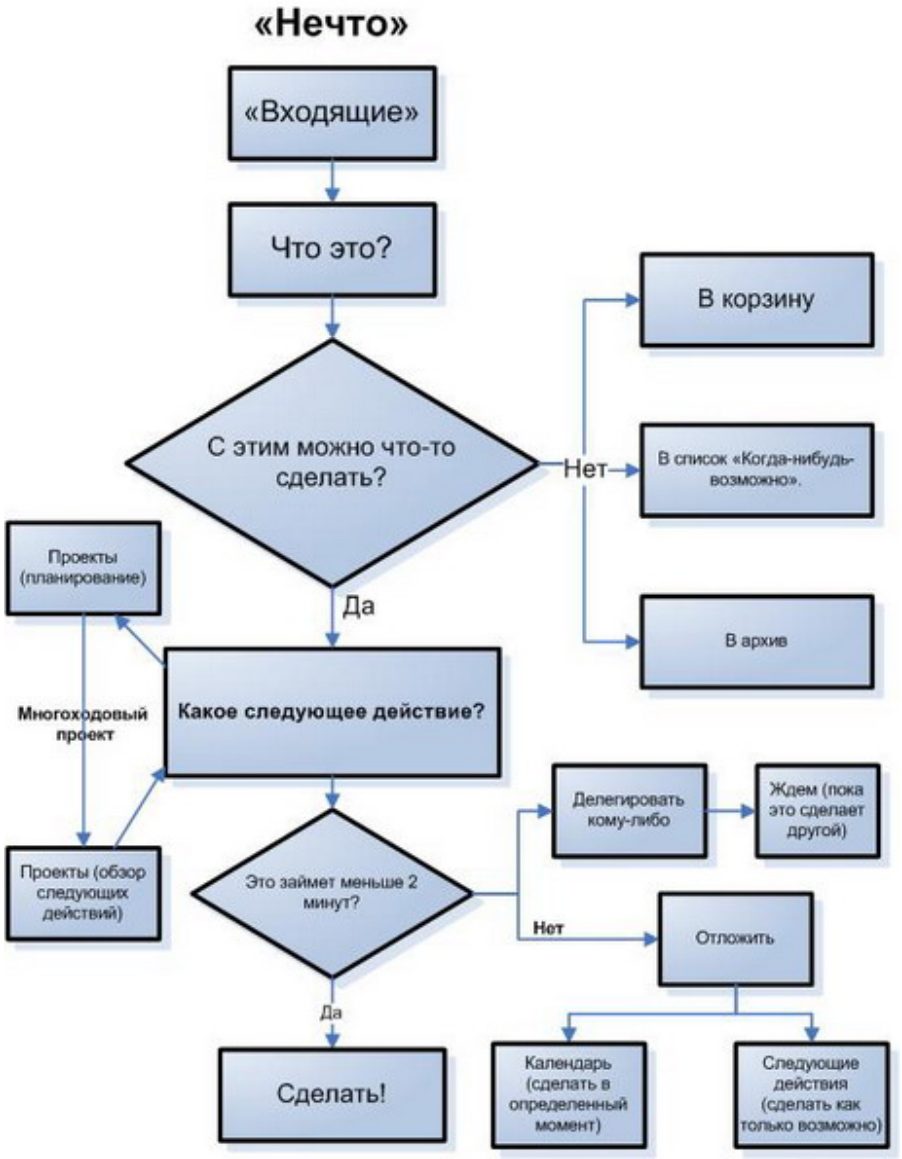

Рис. 1. Представление системы сортировки задач по принципу GTD.

Мы выбрали систему Getting Thing Done в качестве основополагающей для нашего программного обеспечения по двум причинам:

1. Это универсальная система, которую может применять как домохозяйка, так и топ-менеджер. Следовательно, охват такого ПО будет наиболее широким.
2. Система проста для реализации, потому что основана на списках и карточках задач. Другие системы планирования могут подразумевать более сложные для реализации перетаскивания задач, создание временных линий и так далее.

Таким образом, в текущей версии трекера реализованы такие функции: можно создавать задачи, разделять их по спискам, выделять подзадачи из задач (то есть создавать проекты), настраивать описание и срок выполнения. В качестве возможных направлений развития нашего программного продукта можно выделить такие варианты: создание доски Kanban, добавление многопользовательских функций, различные интерфейсные улучшения, разработка мобильной версии.

Разработку мы разбили на несколько циклов, состоящих из типовых этапов. В каждом цикле разработка начинается с бэкенда, затем формируется простой фронтенд и отлаживается вся логика. Следующий шаг – реализация полноценного фронтенда для созданного функционала. Планируется три таких цикла разработки: реализация базового функционала, добавление всех заявленных идей, отладка программы.

Ресурсное обеспечение проекта

Для запуска программы подойдёт любой компьютер, подходящий под следующие системные требования:

- ОС – Windows 10
- Процессор – 2.4 ГГц или больше
- Оперативная память – 4 Гб
- Жесткий диск – 512 Мб
- Графика – Встроенная графика
- Браузер – Chrome, версия > 99
- Python 3, версия > 3.8
- Django Python Library, версия > 4.0.0

В случае, если требуется запуск только основного веб-приложения, список операционных систем расширяется до всех, поддерживающих язык Python и работающих с современными версиями популярных браузеров.

Описание проектного решения

В нашем программном продукте используется следующий стек технологий:

- Объектно-ориентированный язык программирования C#
    - Инструмент создания графических интерфейсов Windows Forms
    - Управляющая базами данных библиотека SQLite для языка C#
- Высокоуровневый язык программирования Python
    - Библиотека для разработки веб-сайтов Django
    - Управляющая базами данных библиотека SQLite для языка Python

Таким образом, проект реализован на двух языках программирования и содержит в себе два различных пользовательских интерфейса и общую базу данных, к которой подключаются оба фронтенда.

Первая часть проекта реализована на языке C#. Такой выбор обусловлен требованием преподавателя разработать некоторый пользовательский интерфейс в нативном для ОС Windows формате при помощи фреймворка .Net Framework, а именно его части — Windows Forms. В качестве альтернативы мы рассматривали язык IronPython, который является реализацией стандарта Python, написанной на C#, вследствие чего поддерживающей .Net Framework. Однако, столкнувшись со сложностью установки, невозможностью компиляции финальной программы и общей “неестественности” такого решения, мы отказались от него в пользу классического C#.

Программа, разработанная с применением .Net Framework не будет являться кроссплатформенной, зато обеспечит пользователю простоту установки и запуска в ОС Windows. Было принято решение создавать данный интерфейс в формате виджета для рабочего стола. Его функционал

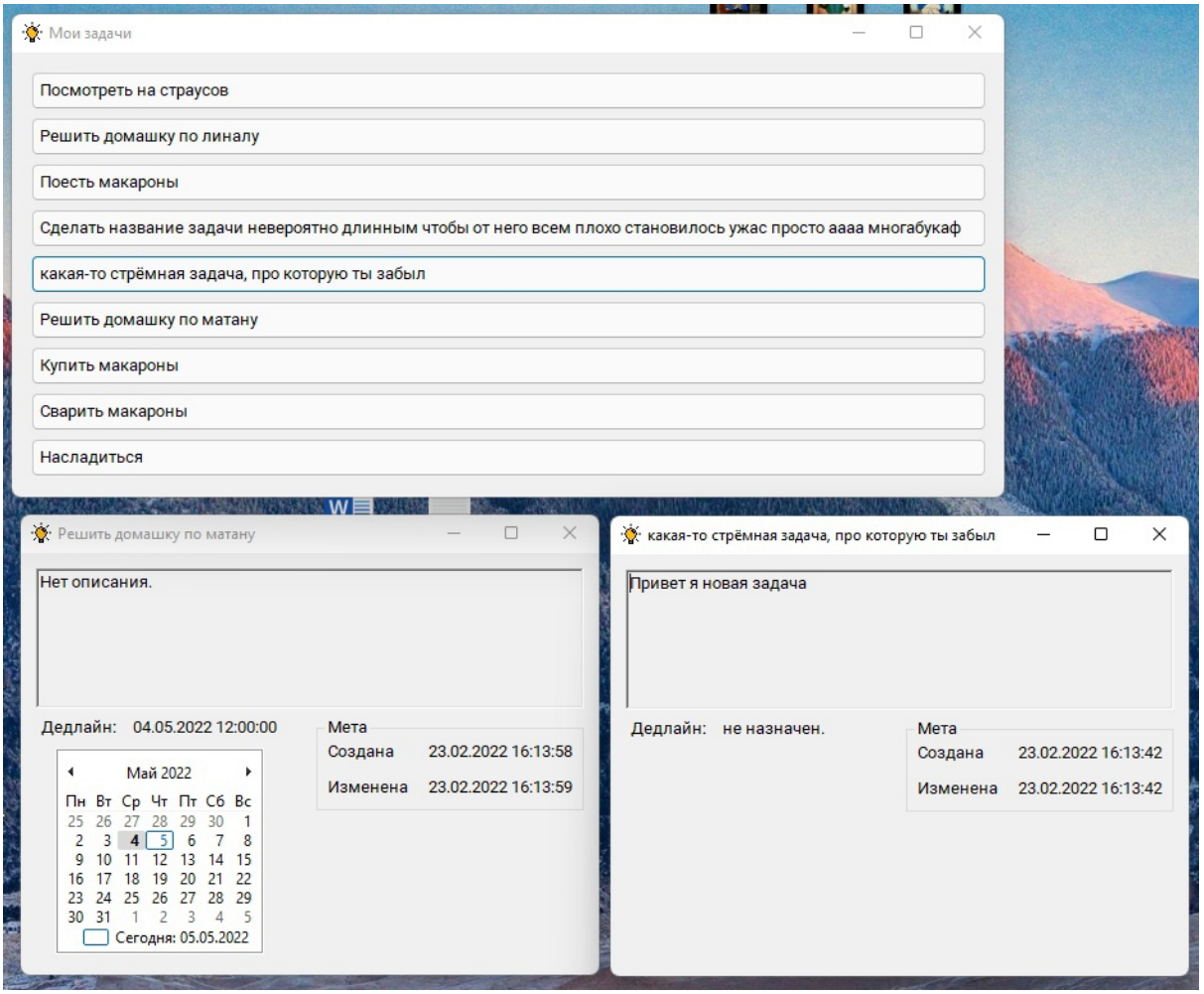

Рис. 2. Внешний вид виджета для рабочего стола.

сильно ограничен и позволяет лишь подключаться к базе данных и просматривать задачи, находящиеся в ней. Такое ограничение преследует сразу две цели:

1. Упрощение пользовательского интерфейса. Приложение-виджет должно быть одновременно быстро запускающимся и не занимающем много пространства на рабочем столе. Мы приняли решение отказаться от функций редактирования в угоду простоте и минималистичности.
2. Купирование возможности возникновения ошибок, связанных с одновременным редактированием базы данных из двух приложений. Зачастую базы данных реализуют клиент-серверную архитектуру, то есть существует некоторый сервер, управляющий файлами базы и

обрабатывающий запросы клиентов. В таком случае к базе данных может осуществляться несколько параллельных подключений, которые не будут мешать друг другу. В свою очередь, архитектура базы SQLite является бессерверной, то есть управляющий такой БД программный код содержится внутри клиентского приложения и только файл, непосредственно хранящий данные базы, “лежит” отдельно. Такая архитектура сильно упрощает процесс разработки, не уступает в скорости классической клиент-серверной архитектуре, но может вызывать ошибки, связанные с одновременным доступом к данным из разных приложений.

Таким образом, интерфейс. разработанный на языке C# имеет ограниченную функциональность и может служить только для цели ознакомления с содержимым базы данных.

Второй интерфейс представляет из себя веб-приложение, созданное с использованием языка Python и библиотеки Django. Идея создать кроссплатформенный интерфейс была основополагающей при выборе темы курсовой работы. Самым универсальным решением для создания пользовательских интерфейсов на данный момент являются браузерные приложения и языки HTML и CSS. Соответственно, мы выбирали технологию лишь для создания бэкенда будущего сайта. Кандидатами снова выступили языки C# и классический Python в реализации C. В связи с тем, что часть работы уже планировалось разрабатывать на C# мы приняли решение выбрать другой язык, то есть Python. Стандартным решением при разработке сайтов на этом языке является библиотека Django, включающая в себя интеграцию с базой данных SQLite. Таким образом, формирование стека было завершено.

В веб-интерфейс была заложена гораздо более широкая функциональность, чем в виджет для рабочего стола. Он позволяет полностью взаимодействовать с базой данных:

- создавать списки;
- создавать задачи;
- менять все параметры задач;
- отмечать задачи как выполненные;
- удалять задачи;
- удалять списки.

Также, в нем реализованы функции сортировки задач, смены имени пользователя, отображение текущей версии приложения.

Перед началом разработки мы создали концепт интерфейса нашего приложения (рис. 3) при помощи инструмента Figma (который является полностью web-based решением, что подтвердило нашу уверенность в правильности выбора стека технологий для разработки программы).

Формат веб-приложения выбран для наибольшей совместимости. Без преувеличения, браузер сейчас установлен на каждом устройстве, поэтому нет сомнения в том, что у пользователя не возникнет проблем с доступом к нашему программному обеспечению. В качестве доказательства могут служить такие веб-сервисы как Figma и Google Документы, которые совсем не уступают полноценным программам для ПК. Поэтому такой выбор точно не станет ограничением для развития проекта. Также решение создавать веб-приложение упрощает разработку, ведь нужно создавать лишь одну версию программного обеспечения для всех доступных платформ. Единственной нерешенной сложностью на данный момент является разработка мобильной версии.Однако, у такого подхода есть и свои значительные недостатки. Так, если программа запускается только для одного пользователя, а не работает на выделенном сервере, то для запуска недостаточно просто скачать набор файлов, ведь для корректной работы нужен интерпретатор языка Python с некоторыми зависимостями. С другой стороны, если разработчик самостоятельно будет предоставлять клиентам уже готовый настроенный сервис, то на его плечи ложится

забота по поддержанию работоспособности оборудования и его замены в случае поломок. Такой подход увеличивает конечную стоимость продукта и заставляет искать источники финансирования для поддержания работоспособности всей системы.

Итогом работы стал интерфейс, изображенный на рис. 4. Он реализует весь заявленный функционал и удобен для повседневного использования. Использование языка Python в связке с библиотекой Django позволяет вносить почти неограниченное количество улучшений в уже работающий продукт.

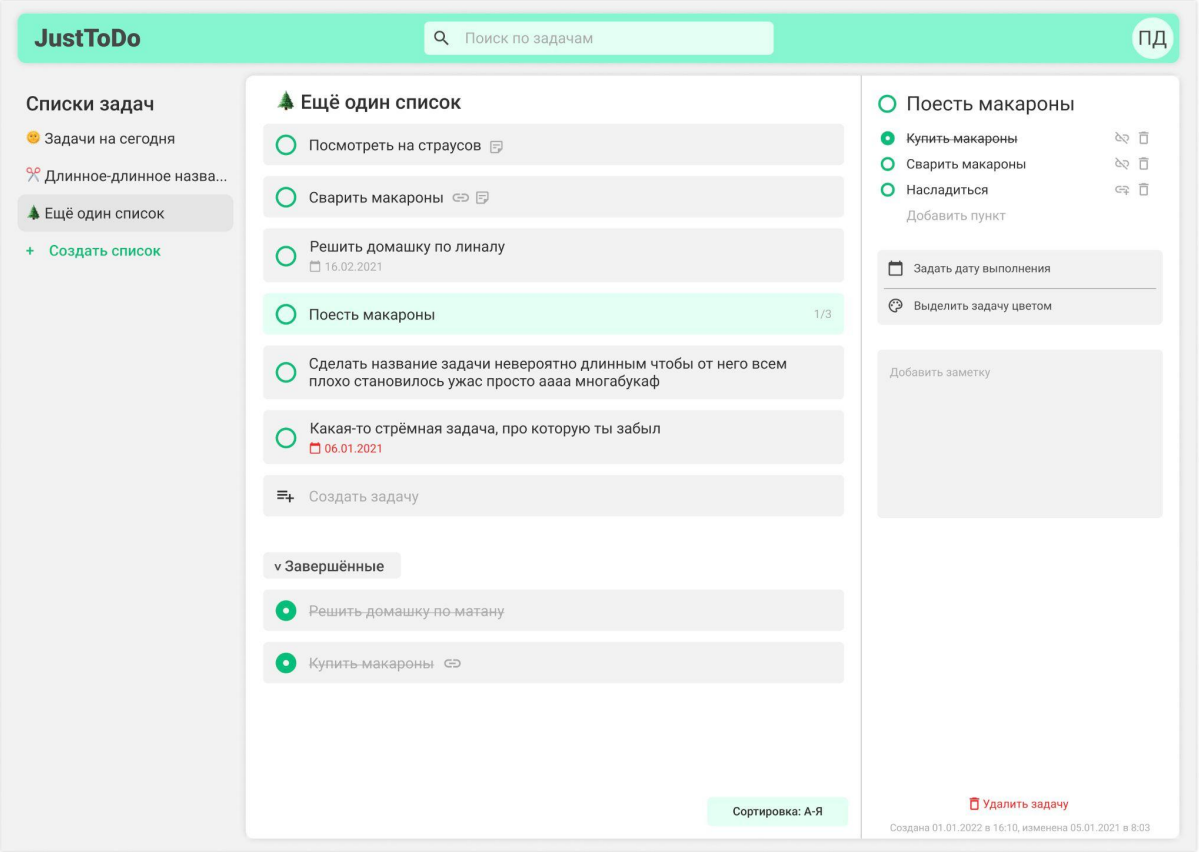

Рис. 3. Первоначальный концепт главного окна таск-трекера.

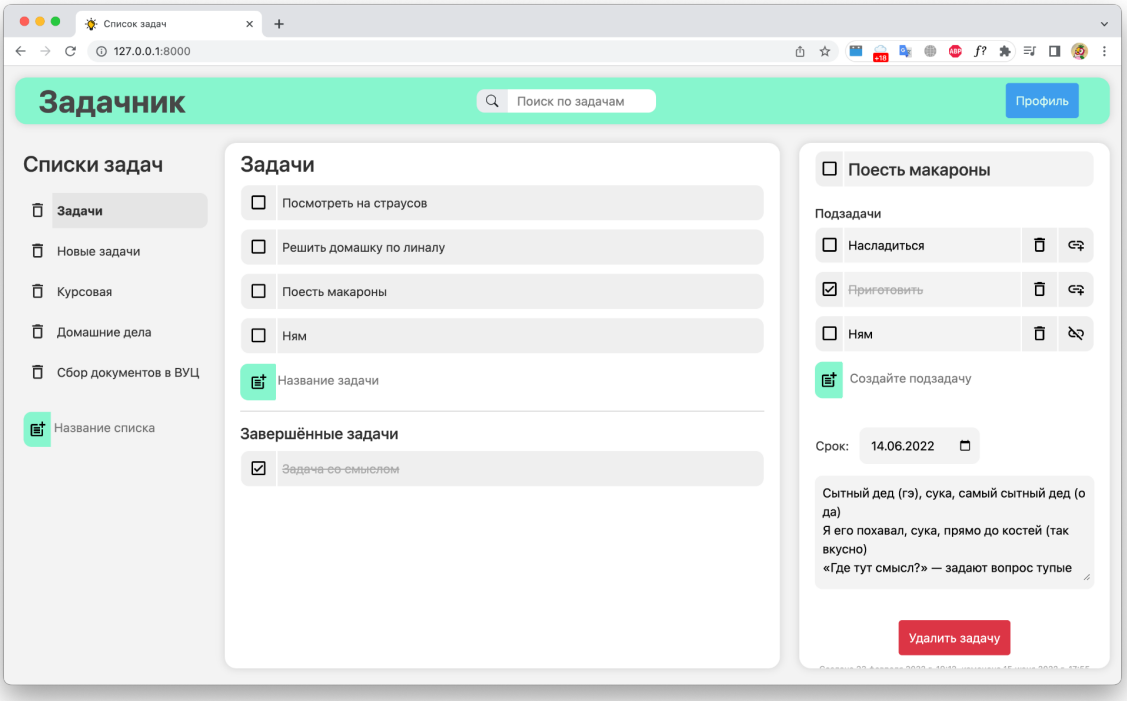

Рис. 4. Финальный внешний вид интерфейса.

План развития проекта

Проект находится в процессе развития: планируется осуществить ряд работ, направленных на его усовершенствование. В настоящей версии нашего трекера реализуется методология Getting Things Done: задачи можно разделять по спискам, выделять подзадачи из задач (то есть создавать проекты), настраивать описание и срок выполнения. В качестве возможных направлений развития нашего программного продукта можно выделить такие варианты:

1. **Внедрение доски kanban**

   Создание доски с карточками, работающей согласно методологии Kanban, стоящей в одном ряду с Getting Things Done по популярности в мировом бизнес-сообществе. Для ведения реальных проектов (а не просто учёта дел) часто удобно пользоваться методологией kanban. Она заключается в том, что выделяется три колонки: “бэклог”, “в работе”, “готово”. Все задачи помещаются в первую колонку, а затем постепенно переходят во вторую и третью. Таким образом очень удобно отслеживать прогресс выполнения проекта, а интересные идеи не забудутся, так как будут помещены в колонку “бэклог”.

2. **Создание мобильной версии веб-приложения**

   В наше время некоторые сервисы работают по парадигме mobile-only, то есть вообще не создают приложений или сайтов для персональных компьютеров. Действительно, нельзя не отметить массовую миграцию пользователей на мобильные устройства и последующий отказ от больших экранов. Поэтому любой сервис, который хочет наиболее активно развиваться должен учитывать этот факт и иметь версию как для больших экранов, так и для экранов смартфона.

3. **Разработка системы авторизации**

   Первоначальная концепция проекта не предполагала реализации многопользовательского интерфейса, однако для вывода приложения на рынок жизненно необходимо создавать аккаунты пользователей и сохранять в тайне конфиденциальную информацию (такую как задачи). Поэтому модуль авторизации очень важен для запуска приложения “в продакшн”.

4. **Имплементация многопользовательских функций**

   Для ведения работы в команде жизненно необходим функционал делегирования задач, создания общих списков и рабочих пространств. Эти функции присутствуют у многих конкурирующих решений, поэтому важно не отставать от общих трендов в отрасли.

5. **Внедрение синхронизации**

   Третьим, но очень важным пунктом по созданию онлайн-инфраструктуры должна стать система синхронизации пользовательских заметок между двумя устройствами. Без неё пользователь, работающий, к примеру, на ноутбуке и телефоне может попадать в ситуацию рассинхронизации данных на разных устройствах. Поэтому очень важна система, которая будет решать конфликты такого рода и принудительно обновлять данные в программе не только при запуске, но и во время работы.

6. **Улучшение пользовательского опыта**

   Нельзя забывать об удобстве пользователей, поэтому необходимо постепенно реализовать ряд вещей, которые повысят визуальную привлекательность и удобство использования программы. К таким вещам можно отнести выделение задач цветами, задание приоритета задачам, возможность добавления фона-картинки для каждого списка. Такие улучшения улучшат пользовательский опыт и создадут конкурентные преимущества перед другими сервисами.

7. **Интеграция со сторонними сервисами**

   Для повышения вовлеченности пользователей можно создавать интеграции с календарями, программами для создания заметок, сервисами видеозвонков, офисными пакетами. Такие интеграции качественно улучшают опыт использования программы, а также действуют как якорь для того, чтобы пользователь возвращался в программу снова и снова.

8. **Улучшение интерфейса**

   Работа над различными интерфейсными улучшениями: хороший интерфейс не обращает на себя слишком много внимания, поэтому важно создавать различные цветовые акценты, а значит позволять пользователям осуществлять выбор фонов для списков (как цвета, так и фотографи), выделение задач цветом, прикрепление файлов к задачам.

9. **Создание обобщённых списков**

   Для улучшения пользовательского опыта и более широких возможностей по обзору запланированных задач важно реализовать функционал так называемых “смарт-списков”, которые позволят организовывать задачи на мета-уровне:

   a. список задач “на завтра”;
   b. список задач, отмеченных как избранные;
   c. список всех задач с установленными дедлайнами и группировкой по времени до дедлайна;

10. **Разработка нативного приложения для различных ОС**

    В качестве одного из второстепенных шагов можно рассматривать разработку отдельных приложений для каждой из популярных операционных систем. Такое решение усложнит развитие проекта, ведь придётся поддерживать большое количество программного кода, имеющего сходный функционал, но работающего на разных

платформах. Однако, оно позволит создать более быстрые программы с адаптированным под конкретную платформу интерфейсом.

11. **Интеграция с умным домом**

Больше маркетинговым, чем действительно полезным шагом может стать интеграция нашего таск-трекера с такими популярными голосовыми ассистентами, как Алиса от Яндекса или Маруся от VK. Такое дополнение может стать драйвером продаж и отличным дополнением для процедурно-генерируемых утренних “подкастов”, которые создаёт Алиса для своих пользователей.

Анализ полученных результатов

Созданный нами программный продукт позволяет людям лучше организовывать свое время, выполнять комплексные задачи с использованием парадигмы Divide and Conquer. В настоящее время приложение основывается на методологии Gettings Things Done, планируется kanban-доска. Дополнительный виджет со списком задач напоминает пользователю о неотложных делах. Применение нашего программного обеспечения должно положительно сказываться на продуктивности как отдельного человека, так и целой компании.

Выводы

Мы изучили методологии ведения проектов и оптимизации времязатрат на выполнение поставленных задач. В ходе работы были реализованы:

- front-end составляющая на языках HTML и CSS, дающая доступ к основному функционалу приложения;
- Python и Django back-end составляющая, осуществляющая контроль, сортировку, хранение, обновление и удаление задач, а также осуществляющую общую инфраструктурную поддержку;
- виджет на языке C# и Windows Forms, осуществляющий вывод актуальных задач на рабочий стол пользователя.

По результатам работы был составлен план дальнейшего развития, состоящий из 10 пунктов, который позволит продукту стать более конкурентоспособным и выйти на массовый рынок.

Список литературы

1. David Allen, Getting Things Done: The Art of Stress-Free Productivity / David Allen. – London : Penguin Books, 2015. – 352 с.
2. William S. Vincent, Django for Beginners: Build Websites with Python and Django (Welcome to Django) / William S. Vincent. – US : WelcomeToCode, 2020. – 317 с.
3. Python 3.9.12 documentation [Электронный ресурс]. – Режим доступа: https://docs.python.org/3.9/. – Дата доступа: 01.04.2022.
4. Bootstrap v5.1.3 documentation [Электронный ресурс]. – Режим доступа: https://getbootstrap.com/docs/5.1/getting-started/introduction/. – Дата доступа: 01.04.2022.
5. Django v4.0 documentation [Электронный ресурс]. – Режим доступа: https://docs.djangoproject.com/en/4.0/. – Дата доступа: 11.04.2022.
6. SQLite v3.38.2 documentation [Электронный ресурс]. – Режим доступа: https://www.sqlite.org/docs.html. – Дата доступа: 06.04.2022.