

Software Development Plan

Таймер Помидора

Выполнил студент группы РП-9:

Сучков Алексей Сергеевич

Содержание

Оглавление

Описание продукта	3
Описание команды	4
Требуемые знания	4
Члены команды и их роли	4
Описание модели процесса разработки	6
Модель разработки.....	6
Суть продукта.....	7
Система контроля версий	9
Используемые инструменты.....	11
Приложения	12
Приложение А: Макеты экранных форм	12
Приложение Б: UML диаграмма	14

Описание продукта

Платформа: РС

Целевая аудитория: Пользователи которые хотят сосредоточиться на плодотворной работе.

Описание: Таймер Помидора - это приложение, для управления временем. Пользователи могут устанавливать таймер на 25 минут работы, после чего следует 5 минут перерыва. Один такой цикл 30 минут называется «помодоро». После каждого четвёртого цикла, делается длинный перерыв 15-30 минут.

Описание команды

Требуемые знания

Этап разработки	Требование
Анализ и сбор требований	<ul style="list-style-type: none">● Знание моделей и их этапов разработки ПО● Умение работать с технической документацией● Умение анализировать и структурировать информацию● Опыт в разработке подобных продуктов● Понимание целевой аудитории
Дизайн программной системы	<ul style="list-style-type: none">● Знание принципов построения UML-диаграмм● Знание ООП и ООА● Опыт в проектировании программных систем
Реализация	<ul style="list-style-type: none">● Базовое знание языка Python● Базовое знание ООП● Базовое знание API библиотеки tkinter● Понимание в разработке GUI приложений
Тестирование	<ul style="list-style-type: none">● Базовое понимание принципов тестирования● Знание составления тест-планов● Опыт написания модульных тестов
Развертывание	<ul style="list-style-type: none">● Умение работать с PyInstaller, setuptools

Члены команды и их роли

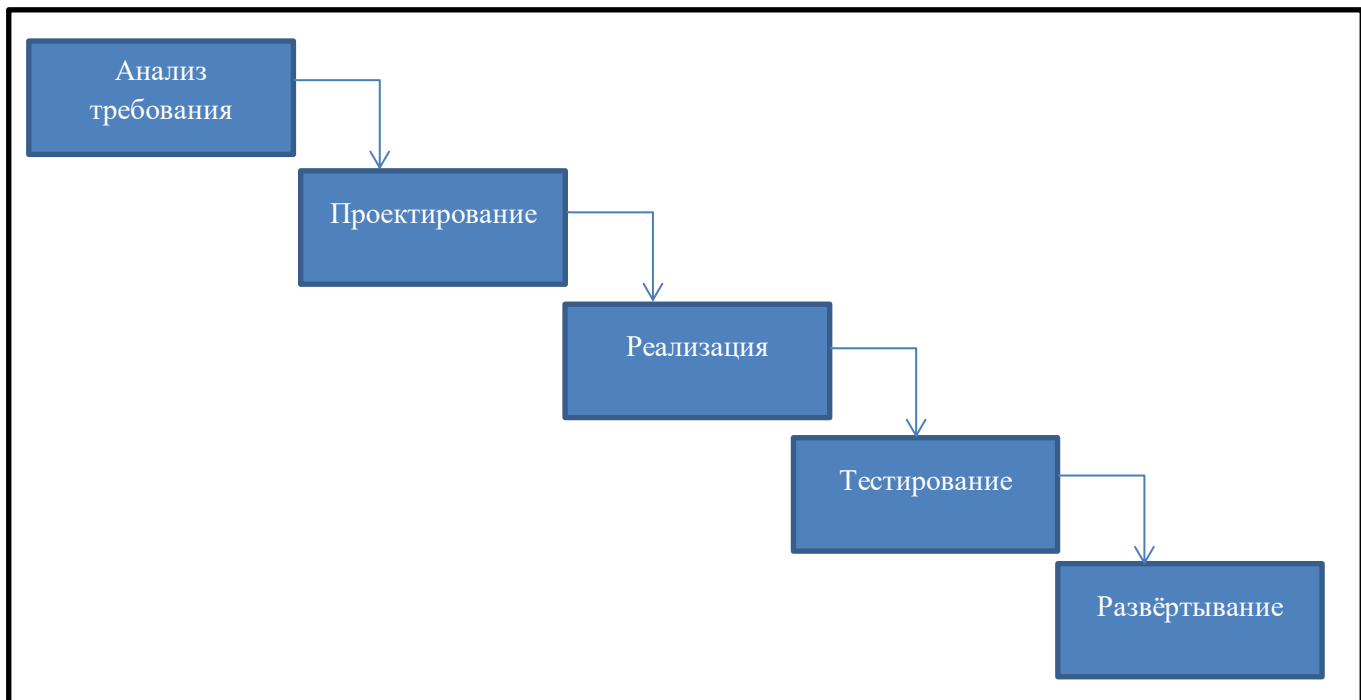
Член команды	Сучков А.С.
Роли	Проектирование и анализ ПО. Реализация всего функционала.

	Тестирование Развертывание ПО.
Навыки	Базовое знание: Python, Git, ООП и ООА
Знание Python	Internal

Описание модели процесса разработки

Модель разработки

Модель разработки: каскадная. Её суть заключается в том, что процесс разработки разбивается на несколько этапов, каждый из которых следует строго один за другим, без возврата на предыдущие стадии.



Время разработки - 30 дней с момента получения технического задания
(14.06.24 - 10.08.24)

Этапы разработки:

1. Анализ и сбор требований: – 14.06.24 - 22.06.24 (8 дней)
2. Дизайн программной системы - 22.06.24 - 06.07.24 (2 недели)
3. Реализация - 06.07.24 - 27.07.24 (3 недели)
4. Тестирование - 27.07.24 - 10.08.24 (2 недели)

Суть продукта

Таймер помодоро - инструмент, помогающий в организации своего рабочего времени.

Рабочее время: промежуток времени (стандартное значение = 25 минут), который обозначает время работы.

Короткий перерыв: промежуток времени (стандартное значение = 5 минут), обозначающее время для отдыха. Начинается после окончания рабочего времени. Также знаменует окончание цикла.

Длинный перерыв: промежуток времени (стандартное значение = 15 минут), обозначающее время для более длинного отдыха. Начинается после того, как прошло 4 цикла.

Время для каждого элемента и цвет заднего фона может быть изменено пользователем в настройках.

Пользователь может запускать таймеры и останавливать их. Так же отслуживать сколько циклов уже пройдено.

План создание продукта



Чтобы продукт считался завершенным, должно быть сделано следующее:

1. Software Development Plan
2. UML-диаграммы
3. Основной функциональные требование и графический интерфейс
4. Дополнительный функционал(необязательно)
5. Модульное тестирование

Система контроля версий

1. Правила ведения репозитория

В репозитории имеется две основные ветки и одна вспомогательная:

- a) `main` – с ней взаимодействует только ведущий разработчик. В ней хранятся только готовый продукт и его обновленные версии.
- b) `dev` – с ней может взаимодействовать каждый разработчик. В нее вливаются готовые структурные функциональные части продукта, не имеющие каких либо багов или ошибок.
- c) `fix` – создается когда в ветке `main` обнаруживается баг, а изменения в `dev` недостаточно стабильны. Предназначена для быстрого исправления ошибок

Для разработки каждый программист имеет свою локальную ветку, в которую он может сохранять любые изменения. Сохранять изменения имеющие ошибки не стоит. При начале работы он должен объединить все изменения из ветки `dev` в свою. После окончания работы разработчик должен влить изменения в ветку `dev`, при этом устранить все конфликты.

2. Оформление коммитов

Коммиты пишутся следующим образом:

Тип коммита(область применения): краткое описание. Длинное описание[если требуется]

- a) Типы коммитов
 - `feat`: добавление нового функционала
 - `fix`: исправление ошибок
 - `refactor`: Правки кода без исправления ошибок или добавления новых функций
 - `delete`: удаление ненужного/лишнего функционала
 - `comment`: комментирование кода
- b) Области применения коммитов

- project: изменения в коде проекта
- test: изменения в тестовой части проекта

3. Примеры коммитов

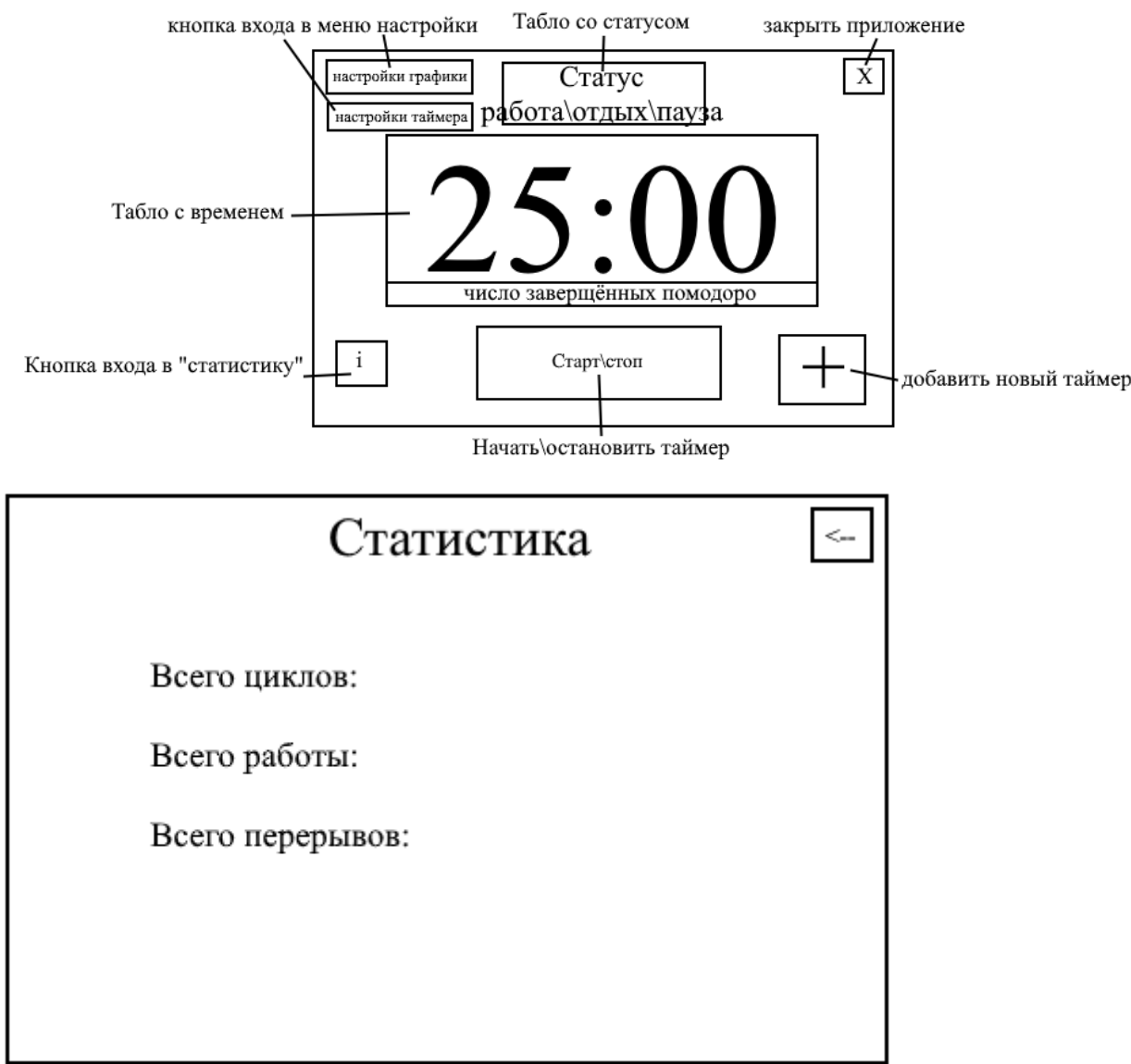
- feat(project): add saving data.
- fix(project): fixed bug that caused color to not change
- feat(test): add test to switching timer

Используемые инструменты

Инструмент	Описание
PyCharm	IDE для написания кода
draw.io	Создание UML, use case диаграмм
Git	Система контроля версий
GitHub	Репозиторий
Microsoft Office	Написание, редактирование документов

Приложения

Приложение А: Макеты экранных форм



Графические настройки



Выберите цвет фона



Выберите цвет кнопок



Настройки таймера



Задайте время работы

Задайте время перерыва

Приложение Б: UML диаграмма

