**Software Development Plan**

**Таймер Помидора**

**Выполнил студент группы РП-9:**

**Сучков Алексей Сергеевич**

**Содержание**

[**1. Описание продукта 3**](#_2gu40rkakg38)

[**2. Описание команды 4**](#_62mml09bf857)

[2.1. Требуемые знания 4](#_jrq51hzbk2s7)

[2.2. Члены команды и их роли 4](#_l317djvp0qxe)

[**3. Описание модели процесса разработки 5**](#_4ybgkcef3zbb)

[3.1. Модель разработки 5](#_yzvm5xbqzx6p)

[3.2. Этапы разработки 5](#_wx0yh3d0rztd)

[**4. Суть продукта 6**](#_8qh4c3cqyc4m)

[**5. План создание продукта 7**](#_fbd1yatgerpq)

[**6. Система контроля версий 8**](#_8vd2i8m9hipo)

[1.1. Правила ведения репозитория 8](#_6ll3bqnyouc0)

[1.2. Оформление коммитов 8](#_6mljbpm6f9qp)

[1.3. Примеры коммитов 9](#_gqnxbl1q4eso)

[**7. Используемые инструменты 10**](#_cwq852j1ybwe)

[**8. Приложения 11**](#_qwo8spaxbkpd)

[8.1. Приложение А: Макеты экранных форм 11](#_58h8xohq6zff)

# **Описание продукта**

**Платформа:** PC

**Целевая аудитория:** все кто хочет сосредоточится 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 | * Знание моделей и из этапов разработки ПО * Умение работать с технической документацией * Опыт в разработке подобных продуктов |
| Дизайн программной системы | * Знание принципов построения UML-диаграмм * Знание Объектно Ориентированного Проектирования/Анализа * Опыт в проектировании программных систем |
| Реализация | * Базовое знание языка Python * Базовое знание ООП * Базовое знание API библиотеки tkinter * Понимание в разработке GUI приложений |
| Тестирование | * Базовое понимание принципов тестирования * Знание составление тест-планов * Опыт написания модульных тестов |
| Развертывание | * Умение работать с PyInstaller, setuptools |

## Члены команды и их роли

|  |  |
| --- | --- |
| **Член команды** | Сучков А.С. |
| **Роли** | Проектирование и анализ ПО.  Тестирование и реализация всего функционала. Развертывание ПО. |
| **Навыки** | Базовое знание: Python, Git, ООП и ООА |
| **Знание Python** | Internal |

# **Описание модели процесса разработки**

## Модель разработки

**Модель разработки:** Waterfall(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.

## Этапы разработки

1. Анализ и сбор требований**:** 1 неделя
2. **Дизайн программной системы:** 1 неделя
3. **Реализация:** 1 неделя
4. **Тестирование:** 1 неделя

# **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.

**План создание продукта**



Чтобы продукт считался завершенным, должно быть сделано следующее:

1. SDP
2. UML-диаграммы
3. Основный функциональные требование и графический интерфейс
4. Дополнительный функционал(необязательно)
5. Модульное тестирование
6. Система контроля версий

### Правила ведения репозитория

В репозитории имеется две основные ветки:

1. main - с ней взаимодействует только ведущий разработчик. В ней хранятся только готовый продукт и его обновленные версии.
2. dev - с ней может взаимодействовать каждый разработчик. В нее вливаются готовые структурные функциональные части продукта, не имеющие каких либо багов или ошибок.
3. 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.

### Оформление коммитов

**Коммиты пишутся следующим образом:**

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

1. Типы коммитов

* **feat:** добавление нового функционала
* **fix:** исправление ошибок
* **rafector:** Правки кода без исправления ошибок или добавления новых функций
* **delete:** удаление ненужного/лишнего функционала
* **comment:** комментирование кода

1. Области применения коммитов

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

### Примеры коммитов

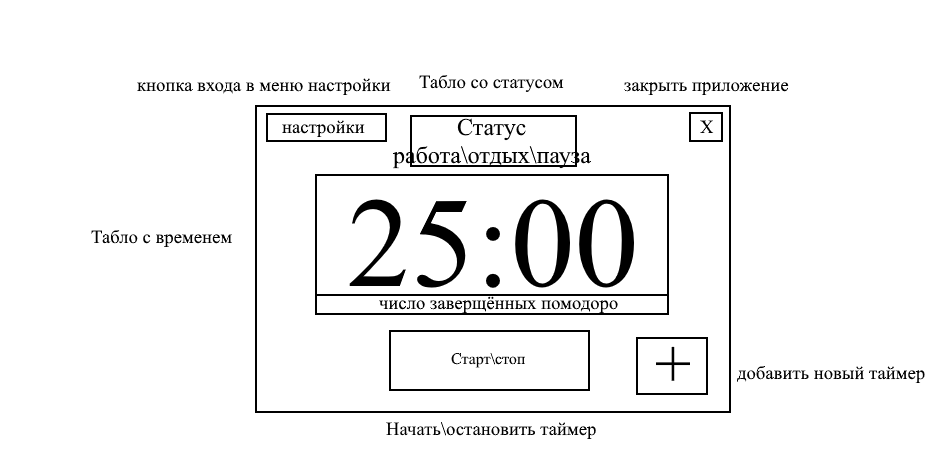
* feat(project): add saving data.
* fix(project): fixed bug that caused color to not change
* feat(test): add test to switching timer

# Используемые инструменты

|  |  |
| --- | --- |
| **Инструмент** | **Описание** |
| PyCharm | IDE для написания кода |
| draw.io | Создание UML-диаграмм |
| Git | Система контроля версий |
| Google Docs | Написание, редактирование документов |

# **Приложения**

## **Приложение А**: Макеты экранных форм



5.2 Приложение Б: UML диаграмма

