# Требования и рекомендации

## Введение

Для выполнения задач вы можете использовать любые инструменты, предоставляемые согласно инфраструктурному листу.

В случае нехватки времени для выполнения всех оставшихся задач вы можете пропускать выполнение некоторых задач в пользу других. Однако ожидается, что вы предоставите максимально завершенную работу в конце каждой сессии, чтобы облегчить оценку вашей работы.

## Правила

Во время чемпионата необходимо соблюдать следующие правила:

* Запрещен доступ в Интернет (кроме одноразового доступа в течение сессии не более 15 минут за всю сессию);
* Запрещено использование любых гаджетов (мобильный телефон, планшет, смарт-часы, шлемы виртуальной реальности ит.д.);
* Запрещено использование ваших собственных устройств хранения данных (USB-накопители, жесткие диски и т.д.);
* Запрещено общение с другими участниками и экспертами чемпионата;
* Запрещено приносить на экзамен книги, заметки и т.д.;
* Разрешено использовать личные устройства ввода информации (клавиатура, мышь, трекбол и т.д.), но эти устройства должны быть проводными, непрограммируемыми и должны работать без дополнительной установки драйверов (эти требования предварительно проверяются техническим экспертом);

Несоблюдение этих правил может привести к удалению с площадки проведения экзамена.

## Название приложения

Используйте соответствующие названия для ваших приложений и файлов.

## Файловая структура

Файловая структура проекта должна отражать логику, заложенную в приложение. Например, все формы содержатся в одной директории, пользовательские визуальные компоненты – в другой, классы сущностей – в третьей.

## Структура проекта

Каждая сущность должна быть представлена в программе как минимум одним отдельным классом. Классы должны быть небольшими, понятными и выполнять одну единственную функцию (Single responsibility principle).

Для работы с разными сущностями используйте разные формы, где это уместно.

## Руководство по стилю

Визуальные компоненты должны соответствовать макету. Обеспечьте соблюдение требований всех компонентов в следующих областях:

• цветовая схема,

• размещение логотипа,

• использование шрифтов,

• установка иконки приложения.

## Макет и технические характеристики

Все компоненты системы должны иметь единый согласованный внешний вид, соответствующий руководству по стилю, а также следующим требованиям:

* разметка и дизайн (предпочтение отдается масштабируемой компоновке; должно присутствовать ограничение на минимальный размер окна; должна присутствовать возможность изменения размеров окна, где это необходимо; увеличение размеров окна должно увеличивать размер контентной части, например, таблицы с данными из БД);
* группировка элементов (в логические категории);
* использование соответствующих элементов управления (например, выпадающих списков для отображения подстановочных значений из базы данных, datepicker для дат и тп);
* расположение и выравнивание элементов (метки, поля для ввода и т.д.);
* последовательный переход фокуса по элементам интерфейса (по нажатию клавиши TAB);
* общая компоновка логична, понятна и проста в использовании;
* последовательный пользовательский интерфейс, позволяющий перемещаться между существующими окнами в приложении (в том числе обратно, например, с помощью кнопки «Назад»);
* соответствующий заголовок на каждом окне приложения (не должно быть значений по умолчанию типа MainWindow, Form1 и тп).

У всех диаграмм и графиков приложения должна быть добавлена легенда.

## Обратная связь с пользователем

Уведомляйте пользователя о совершаемых им ошибках или о запрещенных в рамках задания действиях, запрашивайте подтверждение перед удалением, предупреждайте о неотвратимых операциях, информируйте об отсутствии результатов поиска и т.п. Окна сообщений соответствующих типов (например, ошибка, предупреждение, информация) должны отображаться с соответствующим заголовком и пиктограммой. Текст сообщения должен быть полезным и информативным, содержать полную информацию о совершенных ошибках пользователя и порядок действий для их исправления. Также можно использовать визуальные подсказки для пользователя при вводе данных.

## Обработка ошибок

Не позволяйте пользователю вводить некорректные значения в текстовые поля сущностей. Например, в случае несоответствия типа данных или размера поля введенному значению. Оповестите пользователя о совершенной им ошибке.

Обратите внимание на использование абсолютных и относительных путей к изображениям. Приложение должно корректно работать в том числе и при перемещении папки с исполняемым файлом.

При возникновении непредвиденной ошибки приложение не должно аварийно завершать работу.

## Оформление кода

Идентификаторы переменных, методов и классов должны отражать суть и/или цель их использования, в том числе и наименования элементов управления (например, не должно быть значений по умолчанию типа Form1, button3).

Идентификаторы должны соответствовать соглашению об именовании (Code Convention) и стилю CamelCase (для C#, Java) и snake\_case (для Python).

Допустимо использование не более одной команды в строке.

## Комментарии

Используйте комментарии для пояснения неочевидных фрагментов кода.

Хороший код воспринимается как обычный текст. Не используйте комментарии для пояснения очевидных действий. Комментарии должны присутствовать только в местах, которые требуют дополнительного пояснения.

Используйте тип комментариев, который в дальнейшем позволит сгенерировать XML-документацию, с соответствующими тегами (например, param, return(s), summary и др.)

## Оценка

Каждая задача оценивается путем тестирования реализации требуемого функционала. Так как требования к реализуемой системе очень высоки, возможно, будут использоваться средства для автоматизированного тестирования приложения. В связи с этим, в ходе разработки, может возникнуть необходимость следовать определенным правилам именования и структурирования проекта.

## Предоставление результатов

Все практические результаты должны быть переданы путем загрузки файлов в предоставленную систему контроля версий git. Для каждой сессии могут быть выданы специфические инструкции по сохранению каждого модуля. Практическими результатами являются

• исходный код приложения (в виде коммита текущей версии проекта, но не архивом),

• прочие графические/текстовые файлы.

При создании коммитов используйте содержательные и понятные комментарии.

Для оценки работы будет учитываться только содержимое репозитория. При оценке рассматриваются заметки только в электронном виде (readme.md). Рукописные примечания не будут использоваться для оценки.

Репозиторий обязательно должен содержать описание в формате Markdown (см. шаблон в файле README-Template.md или README-Template\_rus.md). Заполните также дополнительную информацию о проекте и способе запуска приложения в файле readme.md.

Обратите внимание, что дополнительного времени после окончания сессии на сохранение не предусмотрено, поэтому будьте бдительны и загружайте результаты работ своевременно в рамках сессии.

## Взаимодействие через API

Разрабатываемый продукт должен быть реализован с помощью двух разных проектов: клиентская и серверная части. Серверная часть должна быть напрямую связана с базой данных, а также содержать в себе нужные конечные точки (endpoints) для запросов с клиентской части посредством HTTP. Клиентское приложение, в свою очередь, не должно быть связано с базой данных напрямую: получать и отправлять нужные данные следует через запросы в серверной части. Взаимодействие через HTTP должно быть осуществлено следуя архитектуре REST (<https://ru.wikipedia.org/wiki/REST>).

Клиентская часть при работе с API должна учитывать возможные проблемы с интернет-соединением, а также прочие ошибки на стороне сервера. В случае ошибок при запросах к серверу, приложение должно информировать о них пользователя, а также дополнительно указывать техническую информацию об ошибке.

Все методы серверной части должны быть задокументированы с помощью создания коллекции в приложении Postman и содержать в себе описание передаваемых параметров, варианты ответов от сервера. В случае если вы не реализовали нужные методы на стороне сервера, то их необходимо спроектировать и добавить в postman-коллекцию.

## Обновление UI без перезагрузки

В случае если пользователь меняет какие-либо данные в клиентском приложении, а измененные данные отображаются в какой-либо дополнительной части приложения, то необходимо, чтобы интерфейс приложения обновлялся везде. Например, если меняется иконка/название проекта, то после сохранения без перезагрузки приложения должен обновиться и список проектов (в меню).

# Введение

## Информационная система “Проектный офис”

Информационная система “Проектный офис” ориентирована на упрощение и автоматизацию управления проектами, а также улучшение коммуникации и взаимодействия между участниками проектов.

Система должна обеспечивать:

* создание проектов и их описания;
* определение целей и задач проектов;
* определение сроков выполнения проектов;
* учет и распределение бюджета проектов;
* учет и распределение трудовых ресурсов;
* учет и распределение кадровых ресурсов;
* учет и распределение временных ресурсов;
* документирование проектов;
* отслеживание выполнения задач проектов;
* отслеживание сроков выполнения проектов;
* анализ выполнения задач проектов;
* анализ использования ресурсов проектов;
* анализ сроков выполнения проектов;
* анализ бюджета проектов.

### Требования к архитектуре системы

Система должна быть построена на трехзвенной архитектуре:

* клиентская часть;
* серверная часть (база данных и бизнес-логика);
* интеграционный слой (API для взаимодействия с БД).

### Требования к безопасности

Система должна обеспечивать безопасность данных и доступа к системе. Для этого необходимо:

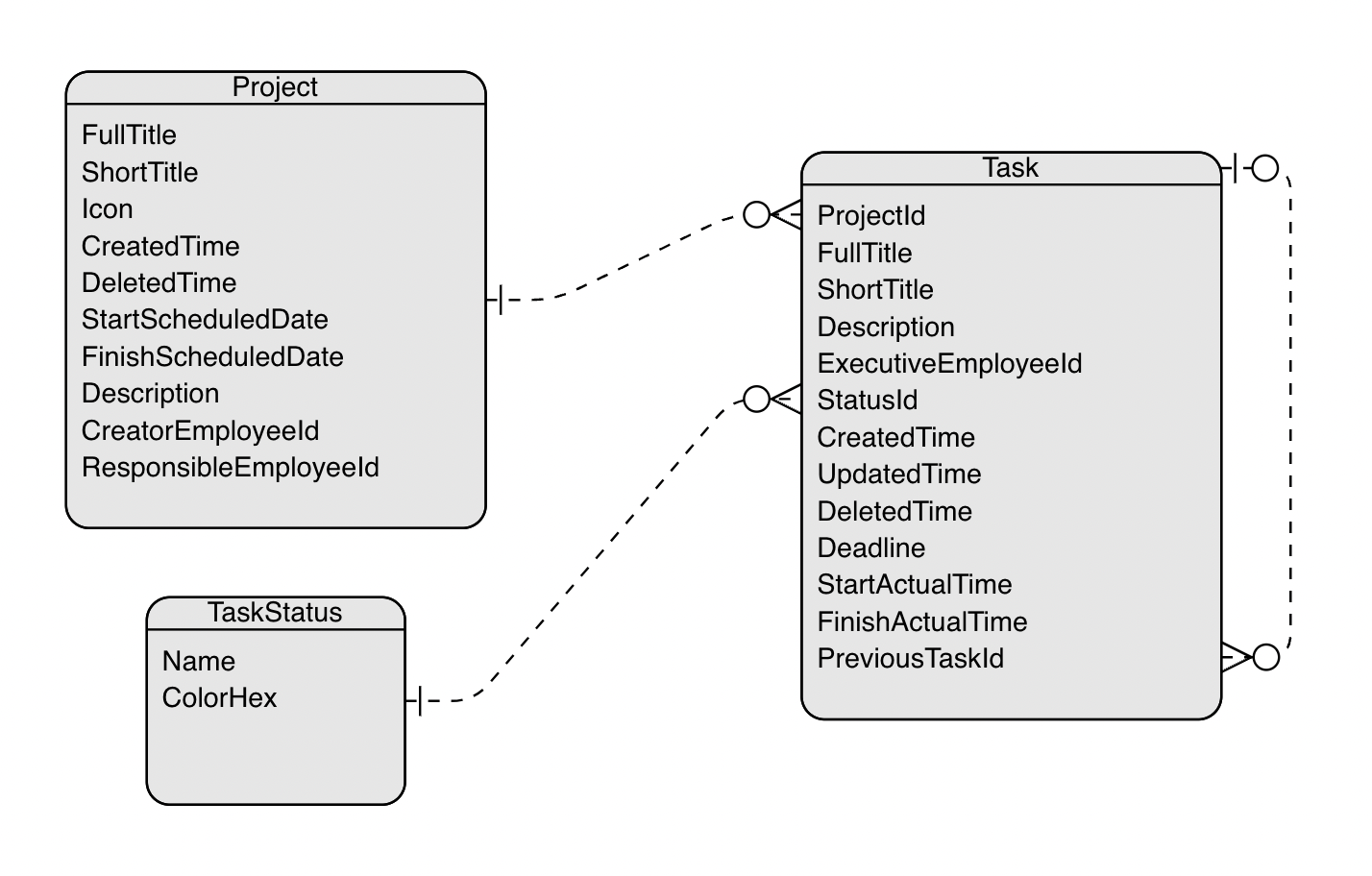
* использовать механизмы аутентификации и авторизации пользователей;
* использовать механизмы шифрования данных в базе данных.

# Сессия 1 (Проекты и задачи)

|  |  |
| --- | --- |
| **Наименование репозитория/ветки** | **Раздел задания** |
| Designing/DB | Проектирование БД |
| Desktop/Project\_Management | Desktop-разработка |
| Designing/API | Документирование API |

## Проектирование и реализация БД

Базовые сущности были спроектированы ранее: предоставленные наработки необходимо перенести в СУБД без изменений и определить подходящие типы данных. Допускается наращивать структуру БД дополнительными полями, таблицами, связями и другими объектами при необходимости. Итоговый вариант схемы данных загрузите в соответствующий репозиторий в формате .pdf/.jpg/.png.



Проект

* Иконка
* Наименование
* Короткое наименование (чаще всего это аббревиатура, легко и компактно идентифицирующая проект, например, PRO)
* Плановые даты начала/окончания
* Дата создания/удаления (дата создания должна фиксироваться на уровне БД)
* Описание (длинный текст)
* Ответственный сотрудник
* Создатель

Задача

* Наименование
* Короткое наименование (автоматически формируется как “Короткое наименование проекта” + порядковый номер задачи внутри проекта, например, “PRO-13”)
* Описание (длинный текст)
* Исполнитель
* Статус
* Дата создания/удаления/обновления (дата создания и обновления должна фиксироваться на уровне БД)
* Крайний исполнения задачи (Deadline)
* Предшествующая задача (когда задачу нельзя начинать до выполнения предыдущей)
* Фактическое время начала/завершения выполнения

Статус (относительно статичный набор данных)

* Наименование
* Цвет (в формате #ffffff)

Необходимо дополнить структуру данных для хранения следующей информации:

* Список наблюдателей за продвижением работы над задачей
* Комментарии/обсуждения сотрудников по конкретной задаче
* Набор вложений и/или ссылок к конкретной задаче

Также требуется спроектировать и реализовать историю изменения статусов задач (на какой статус и когда было изменение). История должна генерироваться на стороне базы данных (проверка будет осуществлена без клиентского приложения) с помощью триггеров.

## Импорт

Выполните импорт данных из предоставленных файлов в созданные таблицы БД без потери данных.

## Desktop-разработка

### Общие требования к приложению

К приложению необходимо добавить иконку из ресурсов.

Необходимо использовать единую цветовую схему во всём приложении в соответствии с макетами.

### Левое меню

Необходимо реализовать общее меню для навигации по функционалу приложения. В меню должны быть добавлены следующие пункты:

* Дашборд
* Задачи
* Гант

Для элементов меню необходимо подготовить пустые страницы-заготовки для каждого раздела. Приложение должно сохранять последний запущенный раздел в приложении, чтобы при следующем запуске приложения открывать его.

Под элементами меню должны быть отображены проекты в виде двух первых букв из названия проекта. Если название проекта состоит из двух или более слов, то для иконки проекта необходимо использовать первые буквы первых двух слов. При наведении на элементы проектов необходимо показывать пользователю подсказку с полным названием проекта. При нажатии на конкретный проект он должен быть выбран текущим и вся информация в последующих разделах приложения должна быть получена по выбранному проекту. По умолчанию должен быть выбран первый проект.

В нижней части компонента “меню” необходимо отобразить версию приложения, а также номер сборки в формате 1.0.456 (где 1.0 - версия, а 456 - номер сборки приложения. При каждой сборке приложения через средство разработки номер должен увеличиваться и сохраняться в параметрах проекта.

### Список задач

Список задач должен содержать задачи только выбранного проекта. По умолчанию пользователю отображаются только задачи, у которых статус не “закрыта”. Задачи должны быть отсортированы в следующем порядке:

1. Задачи со статусом “в работе”, но у которых дедлайн уже прошел
2. Задачи со статусом “открыта”, но у которых дедлайн уже прошел
3. Задачи со статусом “в работе”
4. Задачи со статусом “открыта”
5. Остальные задачи

Каждая задача в списке должна отображать следующую информацию:

* номер (с коротким названием проекта),
* наименование,
* исполнитель,
* дедлайн.

В зависимости от статуса и дедлайна у задачи должна быть определенная заливка.

В верхней части списка задач должна быть реализована возможность поиска задач. Поиск должен работать в реальном времени, а также позволять находить задачи у которых наименование или описание задачи содержат текст из поисковой строки. Результаты поиска необходимо сортировать по совпадению сначала в заголовке задачи, а после по описанию.

### Подробный просмотр задачи

При нажатии на определенную задачу список задач сжимается в два раза, а в правой части окна отображается компонент “подробный просмотр задачи”. Выбранная задача в списке, в свою очередь, должна быть выделена цветом.

В компоненте по просмотру/добавлению информации по задаче необходимо предусмотреть вывод и редактирование всех возможных данных задачи с последующим сохранением информации в БД.

### Создание задачи

В правой нижней части окна списка задач должна быть отображена кнопка для добавления новой задачи, которая открывает компонент “подробный просмотр задачи” для заполнения данных новой задачи.

Реализуйте функционал по созданию задачи с сохранением данных в БД. Необходимо учесть не только поля у объекта “Задача”, но и ряд зависимых наборов данных, в частности:

* список файлов-приложений;
* наблюдателей.

### Удаление задачи

Задачу, которую еще фактически не завершили, допускается пометить как удаленную. Удаленные задачи не должны отображаться в списке. При удалении задачи необходимо учесть, что она могла быть задействована в качестве предшествующей, запросить у пользователя подтверждение действия и только после этого удалить ссылки на задачу и саму задачу. При удалении задачи из цепочки задач, необходимо выполнить перепривязку.

## Документирование API

На основе предоставленных макетов по всем четырем сессиям продумайте и подготовьте документацию по API, которая будет вам полезна для реализации взаимодействия с базой данных.

Убедитесь, что ваша документация API содержит следующую информацию:

* путь для вызова каждой конечной точки;
* какие методы HTTP можно использовать с каждой конечной точкой;
* описание и назначение каждого метода;
* поля данных запроса и куда они идут, например, путь, строка запроса или тело;
* объяснение того, какие данные запроса требуются, а какие являются необязательными;
* список потенциальных ошибок;
* описание типа возвращаемого объекта при успешном выполнении запроса.

Результат необходимо предоставить в формате .xlsx.

Сессия 2 (Gantt Chart)

|  |  |
| --- | --- |
| **Наименование репозитория/ветки** | **Раздел задания** |
| Desktop/Project\_Management | Desktop-разработка |

## Desktop-разработка

### Диаграмма Ганта по проекту

При нажатии на элемент меню “Гант” необходимо открывать пользователю окно с отображением задач в виде диаграммы Ганта на весь экран для выбранного в меню проекта.   
 В верхней части окна необходимо отобразить и реализовать следующие элементы:

* Бегунок для изменения масштаба отображения диаграммы, при перемещении которого необходимо расширять или сжимать горизонтальные интервалы у дней (по оси X)
* Выпадающий список с вариантами ограничений по отображению задач по определенным временным интервалам (1 неделя, 2 недели, 1 месяц, 1 год)
* Элемент для изменения отображаемого промежутка дат
* Кнопка импорта задач к текущему проекту.
* Кнопка “закрыть”, при нажатии которой окно с диаграммой закрывается и открывается раздел “дашборд”

Если всё содержимое диаграммы не вмещается в пользовательское окно, то необходимо предусмотреть горизонтальное пролистывание содержимого диаграммы.

Вся диаграмма должна быть поделена вертикально на дни с указанием даты и дня недели в нижней части. Столбец дня должен быть закрашен:

* цветом для текущего дня,
* цветом для выходного дня (без учета праздничных дней).

Диаграмма должна отображать текущее время с помощью горизонтальной линии, которая в режиме реального времени должна передвигаться по диаграмме. В левой части линии должно отображаться время в формате hh:mm.

Элемент изменения отображаемого промежутка дат должен показывать текущие крайние даты в диаграмме, а также давать возможность с помощью стрелок переключать промежуток по выбранному временному интервалу как в прошлое, так и в будущее.

Например, если выбран временной интервал “1 месяц”, то должны показываться даты начала и окончания периода (1 марта - 31 марта), переключение промежутка в прошлое - февраль, в будущее - апрель.

Задачи в диаграмме должны отображаться в соответствующих позициях по дням и времени старта выполнения (или создания, если время выполнения не определено). Если задача занимает несколько дней, то она должна быть отображена по ширине нужного количества дней. В случае если у задачи есть предшествующая задача, то необходимо нарисовать между этими задачами стрелку, указывающую направление от предшествующей задачи к следующей.

Пользователь должен иметь возможность менять дату/время начала исполнения задачи путем перемещения задачи по диаграмме методом drag&drop. После того как пользователь “отпустил” задачу в новом временном промежутке, то система должна автоматически сохранять новые данные в БД.

При нажатии на задачу необходимо отображать всплывающее окно с информацией о задаче.

Если задача берёт начало или заканчивается за текущим отображаемым промежутком, то задачу необходимо отображать, но с дополнительной пометкой в левой или правой части задачи соответственно.

Параметры отображаемого промежутка дат и масштаба должны сохраняться даже после перезапуска приложения.

При наведении курсора мыши на задачу, необходимо у остальных задач уменьшать заливку на 30%, но в случае если пользователь навёл курсор на задачу которая находится в цепочке задач, то у задач из цепочки заливку изменять не нужно.

### Всплывающее окно с информацией о задаче

Во всплывающем окне необходимо отображать количество времени в удобном формате:

* для завершенных задач - фактически потраченное время,
* для текущих задач - время до дедлайна,
* для будущих задач - планируемое время на выполнение.

Удобный формат подразумевает следующую логику:

* числовое значение с единицей измерения не должно превышать переход в следующий “разряд”,
* не должно быть нулевых значений с более крупными единицами измерения времени,
* не указывать слишком мелкие единицы измерения для значения (не более двух единиц измерения),
* округлять по математическим правилам (учитывая размерность разряда)

Например:

* 63 минуты 15 секунд - это 1 час и 3 мин ~~15 сек~~,
* 15 дней 45 минут - это 15 дней ~~00 час 45 мин 00 сек~~,
* 12 часов 50 минут 32 секунды - это 12 час 51 мин.

### Импорт

Для удобства проверки и отладки диаграммы необходимо предусмотреть механизм импорта файла с задачами для текущего проекта. Перед добавлением новых задач из файла, необходимо автоматически удалить старые задачи по проекту. Структура файла для импорта с требуемым расширением присутствует в ресурсах к заданию. Другие форматы и расширения файлов недопустимы, о чем пользователя необходимо оповещать в клиентском приложении, как и об успешном факте импорта с количеством записей.

# Сессия 3 (Dashboard)

|  |  |
| --- | --- |
| **Наименование репозитория/ветки** | **Раздел задания** |
| Desktop/Project\_Management | Desktop-разработка |
| Library/Employee\_Time | Алгоритмы. Разработка библиотеки |
| Designing/UI | Проектирование UI |
| Designing/UML | Документирование |
| Test/Case | Тест-кейсы |

## Desktop-разработка

1. Dashboard по проекту

Планируется, что данный раздел будет ключевым для руководства и постоянно открыт у определенных типов пользователей. Информация в блоках должна обновляться каждые 30 секунд.

Необходимо реализовать раздел с выгрузкой по данным в приложении. Данный раздел должен быть собран из отдельных компонентов, которые должны быть размещены в интерфейсе в виде сетки для удобного отображения при разных размерах окна. Например,

* если пользователь просматривает окно на всю ширину - должны помещаться 4 элемента в одну линию,
* при уменьшении размера окна - 3 или 2 элемента в одну линию, а остальные должны переместиться ниже.

Таким образом пользователь должен всегда видеть информационные блоки в удобном режиме.

Блоки (не отступы между ними) должны растягиваться и сжиматься при изменении размеров окна (быть “резиновыми”) и только в случае критически маленького размера блоков они должны быть перемещены.

Дашборд должен формироваться по выбранному проекту и содержать описанные ниже блоки.

1. Блок с выводом количества незавершенных задач;
2. Блок с выводом количества просроченных задач. В случае если просроченных задач больше двух, необходимо выделять блок красным цветом;
3. Блок с выводом количества активных задач в текущий момент времени. Если на текущий момент на компьютере пользователя рабочий день (с 09:00 до 18:00), то при отсутствии активных задач необходимо выделять блок красным цветом;
4. Блок с выводом количества незавершенных задач на текущую неделю;
5. Таблица ТОП5 сотрудников по количеству закрытых задач в текущем месяце;
6. Таблица ТОП5 сотрудников по количеству просроченных задач в текущем месяце.

## Алгоритмы. Разработка библиотеки

Разработайте библиотеку, которая позволит выполнить расчеты по времени работы сотрудника для его оптимизации.

Необходимо обязательно следовать правилам именования библиотек, классов и методов в них. В случае ошибок в рамках именования ваша работа не может быть проверена и ваш результат не будет зачтен. Классы и методы должны содержать модификатор public (если это реализуемо в рамках платформы), чтобы внешние приложения могли получить к ним доступ.

Описание входных/выходных параметров, методику и примеры расчета, а также тестовые данные для тестирования библиотеки можно найти в ресурсах к заданию.

## Проектирование UI

Wireframes — это организация полного функционала конечного продукта, в виде структуры с представлением элементов интерфейса и навигации, их взаимодействия друг с другом. Как правило, в wireframes не используется типографика, цвет и любые другие графические элементы оформления, так как основное внимание уделяется функциональности, поведению и содержанию.

Разработайте wireframe для мобильного клиента. Мобильное приложение предназначено для клиентов. При разработке wireframe вам необходимо соблюдать принципы UX для дизайна мобильного программного обеспечения.

Ниже приведены некоторые требования:

* все необходимые поля должны быть функциональными (не должно быть «пустых» областей;
* кнопки и другие элементы управления должны быть четко обозначены;
* оптимальное использование полей ввода, списков, меню и т.д.

Разработайте  wireframe для мобильного приложения пользователя системы. Мобильное приложение ориентировано на исполнителя задач. В макете обязательно должны быть учтены экраны:

1. Список задач
2. Информация по задаче
3. Профиль пользователя
4. Чат с сотрудниками по проекту

## Тест-кейсы

Для выполнения процедуры тестирования библиотеки по расчету времени работы сотрудника . Вам нужно описать пять сценариев. Прецедент может быть выполнено успешно, а может быть отклонен согласно требованиям предметной области. Необходимо, чтобы варианты тестирования демонстрировали различные исходы работы алгоритма. Для описания тестовых сценариев в ресурсах предоставлен шаблон.

## Документирование

### Postman-коллекция

Необходимо сформировать postman-коллекцию для созданного вами проекта серверной части с описанием логики его работы, общей информации о проекте и отображением информации по каждому созданному запросу и примеры успешных и неуспешных выполнений. Наименование для коллекции: api\_doc\_{номер вашего рабочего места}.postman\_collection.json

### Руководство пользователя

Вам необходимо разработать руководство пользователя для вашего настольного приложения, которое описывает последовательность действий для выполнения всех функций вашей системы.

При подготовке документации старайтесь использовать живые примеры и скриншоты вашей системы для более наглядного пояснения шагов работы с различным функционалом.

Обратите внимание на оформление документа: оформите титульный лист, используйте автоматическую нумерацию страниц, разделите руководство на подразделы и сформируйте оглавление, используйте ссылки на рисунки, нумерованные и маркированные списки для описания шагов и т. д.

Сохраните итоговый документ с руководством пользователя в формате Word , используя в качестве названия следующий шаблон: Руководство пользователя XX , где XX - номер вашего рабочего места.

### Диаграмма деятельности

Разработайте диаграмму деятельности для прецедента «Создание задачи». Сохраните файл с диаграммой в форматах .vsdx и .pdf.

### Диаграмма последовательности

Разработайте диаграмму последовательности UML для описания процесса работы с диаграммой Ганта: изменение сроков выполнения задачи с помощью drag&drop. Сохраните файл с диаграммой в форматах .vsdx и .pdf.

# Сессия 4 (Reports & Diagrams)

|  |  |
| --- | --- |
| **Наименование репозитория/ветки** | **Раздел задания** |
| Desktop/Project\_Management | Desktop-разработка |
| Library/Employee\_Salary | Разработка библиотеки |
| Test/Unit | Модульные тесты |

## Desktop-разработка

Требуется доработать Dashboard блоками с функционалом для генерации и экспорта отчетных/статистических данных.

### Экспорт в .csv

Раздел с возможностью выгрузить следующие данные в .csv-файл:

* 1. Список сотрудников
  2. Список задач
  3. Закрытые задачи за текущий календарный месяц
  4. Открытые задачи на следующий календарный месяц
  5. Список задач за последние 14 дней

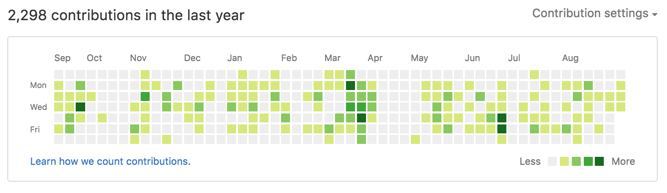
### График

График в виде бублика с указанием процентного соотношения задач и их статусов.

### Диаграмма

Диаграмма с “тепловым” отображением количества завершенных задач в тот или иной день. Данные должны быть отображены в виде квадратов с необходимой заливкой. На экране должна быть отображена информация на 10 недель в прошлое.

Пример диаграммы:



После подсчета числовых значений необходимо определить максимальную загруженность среди всех интервалов времени и реализовать цветовое выделение всех ячеек в соответствии с таблицей ниже.

|  |  |
| --- | --- |
| Значение от максимального | Цвет |
| [0-20) % | #b6bdff |
| [20-40) % | #919cff |
| [40-60) % | #6d7cff |
| [60-80) % | #485bff |
| [80-100] % | #243aff |

### Вывод истории изменения статусов задач

В компоненте подробного описания задачи необходимо добавить вывод истории смены статусов задачи с выводом следующей информации:

* начальный статус (который был изменен),
* новый статус (на который изменили),
* дата и время смены статуса.

Историю смены статусов необходимо отображать от самых свежих записей к старым.

## Разработка библиотеки

Необходимо разработать библиотеку для расчета заработной платы сотрудников.

## Модульное тестирование

Реализуйте 5 unit-тестов на основе технологии TDD для библиотеки, функционал которой описан в предыдущем модуле. Важно, чтобы тестовые данные предусматривали различные ситуации (например, штрафы за просроченные задачи).