

Министерство науки и высшего образования Российской Федерации
ФГБОУ ВО «Кубанский государственный технологический университет»
(ФГБОУ ВО КубГТУ)

Институт компьютерных систем и информационной безопасности
Кафедра информационных систем и программирования
Специальность 09.03.04 программная инженерия
Профиль проектирование и разработка программного обеспечения

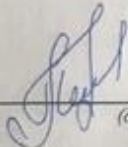
КУРСОВАЯ РАБОТА

по дисциплине технологии разработки программного обеспечения
(наименование дисциплины)

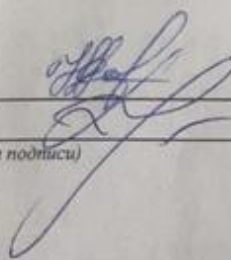
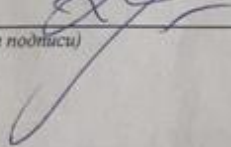
на тему: «Будильник»
(тема курсовой работы)

Выполнила студентка курса 2 группы 19-КБ-ПР1
Привалова Кристина Кирилловна
(фамилия, имя, отчество)

Допущен к защите 23.12.2020
(дата)

Руководитель (нормоконтролер) работы  поц Попова О.Б.
(должность, подпись, дата)

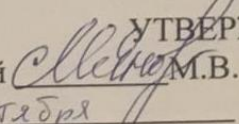
Защищена 24.12.2020 Оценка отм
(дата)

Члены комиссии: _____
(должность, подпись, дата, расшифровка подписи)  Кушнир Н.В.
 Тотухов К.Е.

Краснодар
2020 г.

ФГБОУ ВО «Кубанский государственный технологический университет»
(ФГБОУ ВО КубГТУ)

Институт компьютерных систем и информационной безопасности
Кафедра информационных систем и программирования
Направление подготовки прикладная информатика
Профиль беспрофильный

УТВЕРЖДАЮ
Зав. кафедрой  М.В. Янаева
«10» сентября 2020 г.

ЗАДАНИЕ
на курсовой проект

Студентке Приваловой Кристине Кирилловне группы 19-КБ-ПР1 2 курса
Тема работы: «Будильник.»

План работы:

1. Изучение предметной области
2. Проектирование
3. Описание реализованных диаграмм

Объем работы:

- а) пояснительная записка 32 с.
- б) листинг 3 листа

Рекомендуемая литература:

1. Йордон. «Объектно-ориентированный анализ и проектирование систем»
2. Роберт А. Максимчук. «UML для простых смертных»
3. «Автоматизация проектирования вычислительных систем.» ред. М.Брейер

Срок выполнения:

с «1» 09 по «25» 12 2020г.

Срок защиты:

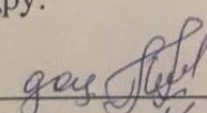
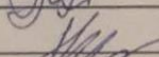
«25» 12 2020г.

Дата выдачи задания:

«10» 09 2020г.

Дата сдачи работы на кафедру:

«21» 12 2020г.

Руководитель работы  доцент Попова О.Б.
Задание приняла студентка  Привалова К.К.

Реферат

Пояснительная записка курсового проекта (работы) 32 с.; 15 рис.; 7 источников; 3 приложения.

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ, ПРОЕКТИРОВАНИЕ, МОДЕЛЬ, КЛАСС, СИСТЕМА, БУДИЛЬНИК, ТАЙМЕР, UML, BPMN, EPC, FURPS+, IDEF0, DFD, ДИАГРАММЫ.

Объектом исследования является программное обеспечение способное работать в качестве будильника имеющем систему взвода и сброса.

Целью данной курсовой работы состоит в разработке программного обеспечения “Будильник”

В процессе создания курсовой работы было создано два приложения WinForm, написанных на языке программирования C# в формате исполняемых файлов (.exe). Создавались данные программы в среде разработки Microsoft Visual Studio 2019.

При выполнении курсовой работы были выполнены все этапы создания программного продукта: от постановки задачи до практической реализации. В ходе выполнения курсовой работы были приобретены навыки работы со специализированной литературой, справочниками, стандартами.

Оглавление

Нормативные ссылки	4
Введение	5
Формулировка задачи	6
1 Временные и сетевые диаграммы	7
2 Функциональное проектирование	10
3 Диаграмма потоков данных (DFD).....	13
4 Объектно-ориентированное проектирование	15
5 ЕРС	21
6 BPMN	23
Заключение	25
Список использованных источников	26
Приложение А	27
Приложение Б.....	28
Приложение В	31

Нормативные ссылки

В настоящей пояснительной записке используются ссылки на следующие нормативные документы:

ГОСТ 19.701-80 – ЕСПД. Схемы алгоритмов, программ, данных и систем.

ГОСТ 7.32-2001 «Отчет о научно-исследовательской работе. Структура и правила оформления».

ГОСТ 7.1-2003 «Библиографическая запись. Библиографическое описание. Общие требования и правила составления».

ГОСТ 7.80-2000. «Библиографическая запись. Заголовок. Общие требования и правила составления».

ГОСТ 7.32-2001 «Отчет о научно-исследовательской работе. Структура и правила оформления».

ГОСТ 19.701-80 – ЕСПД. Схемы алгоритмов, программ, данных и систем.

Введение

Microsoft Visual C# представляет собой весьма эффективный и в то же время простой язык, предназначенный преимущественно для разработчиков, создающих сборки приложений в среде Microsoft .NET Framework. Visual C# унаследовал множество лучших свойств от C++ и Microsoft Visual Basic, но при этом его разработчики постарались избавиться от различных несоответствий и анахронизмов, в результате чего появился более понятный и логичный язык, который удобен в изучении как опытным программистам, работавшим на любом другом языке программирования высокого уровня, так и новичкам в этом деле. Именно поэтому C# позволяет студентам с легкостью начать изучать азы программирования.

Целью исследования, проводимого в рамках настоящей курсовой работы, является разработка и реализация на языках высокого уровня алгоритмов решения задач, представленных в задании курсовой работы.

Объектами исследования настоящей курсовой работы являются методы и технологии разработки программных продуктов.

Предметами исследования настоящей курсовой работы являются методы, алгоритмы и приёмы разработки программ обработки файлов и строк.

Информационной базой исследования является учебная литература по информатике и программированию, техническая документация по языку C# инструментальной среды MS Visual Studio 2019.

Формулировка задачи

Программное обеспечение встроенного микропроцессора для будильника. Будильник постоянно отображает текущее время (часы, минуты, например: 12: 00). Управление будильником осуществляется следующими кнопками: –кнопкой режима установки времени, –кнопкой режима установки времени срабатывания, –двумя отдельными кнопками для установки часов и минут, –кнопкой сброса сигнала «СБРОС». На будильнике имеется переключатель режима работы со следующими положениями: «ВЫКЛ» и «ВКЛ». Для установки текущего времени нужно нажать на кнопку режима установки и, при нажатой кнопке, нажимать на кнопки установки часов и минут. При каждом нажатии на кнопки, устанавливаемое значение увеличивается на одну единицу (один час или одну минуту соответственно). При достижении максимального значения производится сброс. Для установки времени срабатывания будильника нужно нажать на кнопку режима установки времени срабатывания и, держа кнопку нажатой, нажимать на кнопки установки часов и минут. Когда переключатель режима работы находится в положении «ВКЛ», при достижении времени срабатывания происходит подача звукового сигнала в течение одной минуты. Сигнал можно прервать, нажав на кнопку «СБРОС». При этом сигнал должен быть возобновлен через пять минут. При установке переключателя в положение «ВЫКЛ» звуковой сигнал не подается. При нажатии на кнопку режима установки времени, будильник должен отображать время срабатывания.

1 Временные и сетевые диаграммы

Временные и сетевые диаграммы полезны для представления графика работ. Временная диаграмма показывает время начала и окончания каждого этапа и его длительность. Сетевая диаграмма отображает зависимости между различными этапами проекта. Эти диаграммы можно создать автоматически с помощью программных средств поддержки управления на основе информации, заложенной в базе данных проекта.

На основе приведенных значений длительности этапов и зависимости между ними строится сетевой график последовательности этапов (рис. 1). На этом графике видно, какие работы могут выполняться параллельно, а какие должны выполняться последовательно друг за другом. Этапы обозначены прямоугольниками. Контрольные отметки и контрольные проектные элементы показаны в виде овалов и обозначены буквой М с соответствующим номером. Даты на данной диаграмме соответствуют началу выполнения этапов. Сетевую диаграмму следует читать слева направо и сверху вниз.

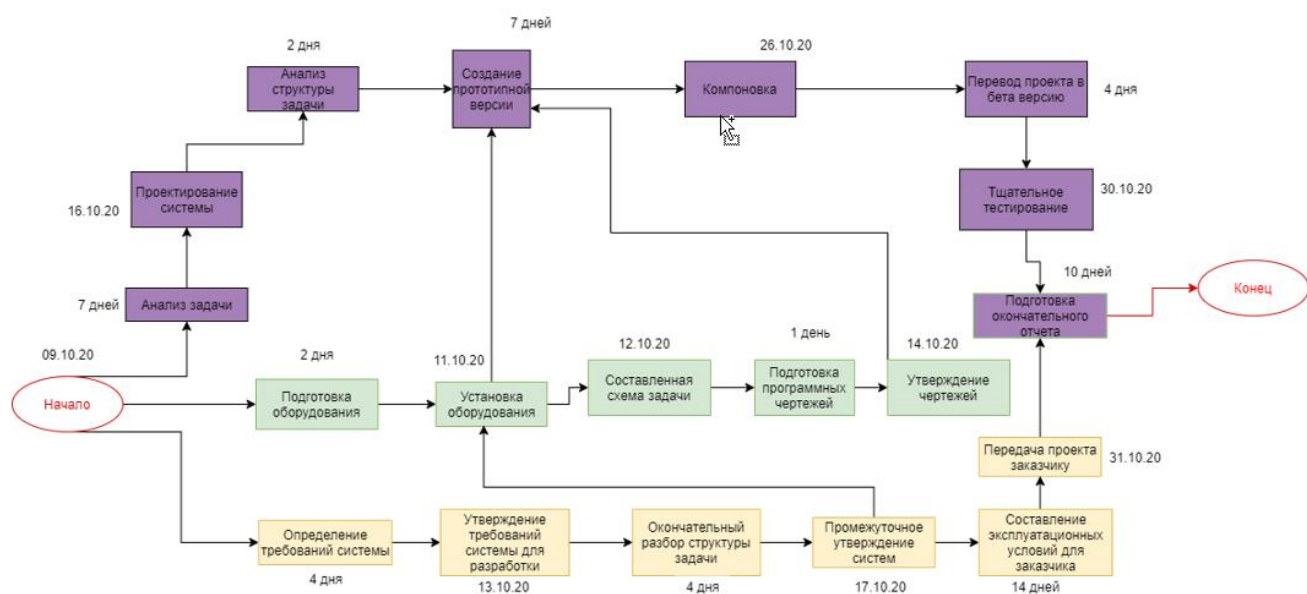


Рисунок 1-Сетевая диаграмма этапов

Минимальное время выполнения всего проекта можно рассчитать, просуммировав в сетевой диаграмме длительности этапов на самом длинном пути (длина пути здесь измеряется не количеством этапов на пути, а суммарной длительностью этих этапов) от начала проекта до его окончания (это так называемый критический путь). В нашем случае продолжительность проекта составляет 16 недель. На рис. 2 критический путь показан более толстыми линиями, чем остальные пути. Таким образом, общая продолжительность реализации проекта зависит от этапов работ,

находящихся на критическом пути. Любая задержка в завершении любого этапа на критическом пути приведет к задержке всего проекта. Задержка в завершении этапов, не входящих в критический путь, не влияет на продолжительность всего проекта до тех пор, пока суммарная длительность этих этапов (с учетом задержек) на каком-нибудь пути не превысит продолжительности работ на критическом пути. Например, задержка этапа Т8 на срок, меньший 20 дней, никак не влияет на общую продолжительность проекта. На рис. 2 представлена временная диаграмма, на которой показаны возможные задержки на каждом этапе.

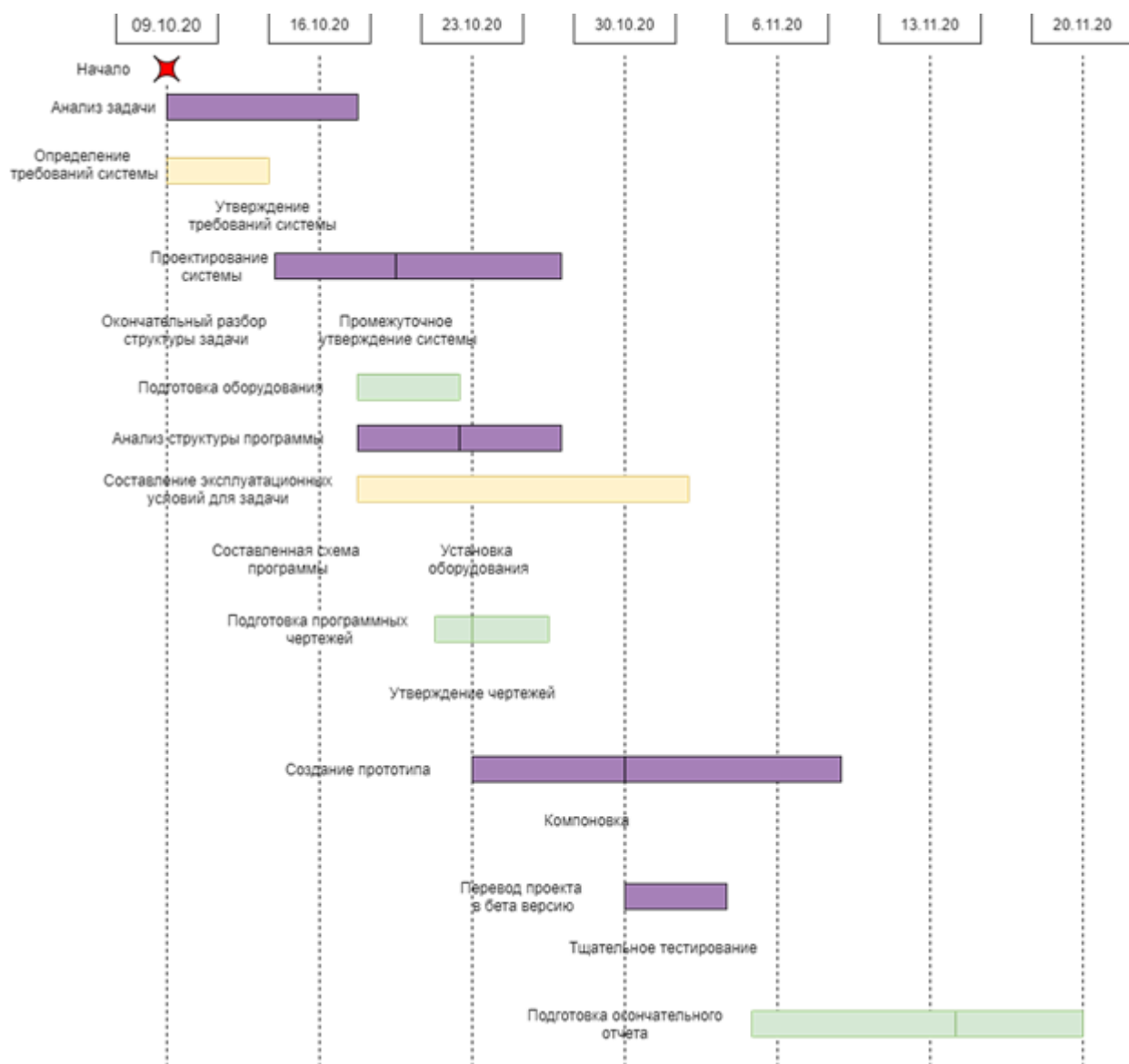


Рисунок 2 – Временная диаграмма длительности этапов

Приведенная таблица может быть использована программными средствами поддержки процесса управления для построения временной диаграммы занятости сотрудников на определенных этапах работ (рис. 3). Персонал не занят в работе над проектом все время его реализации. В течение периода незанятости сотрудники могут быть в отпуске, работать над другими проектами, проходить обучение и т.д.

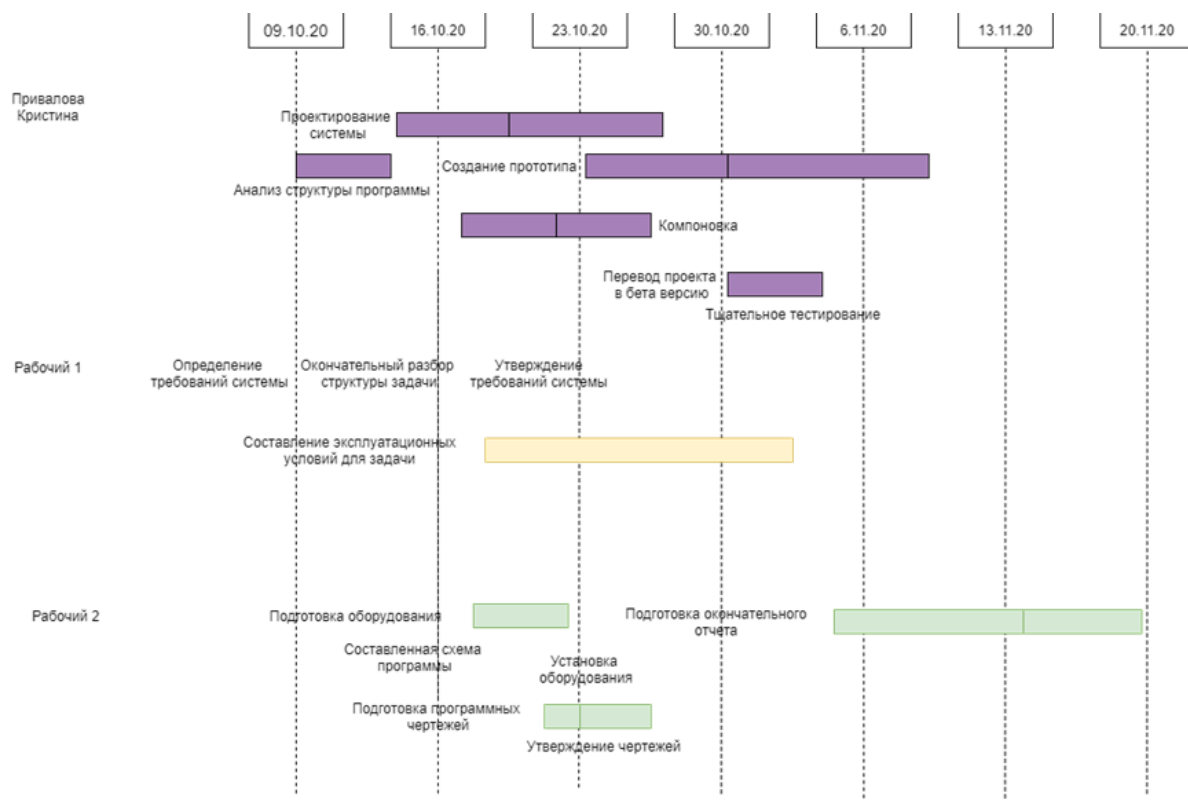


Рисунок 3-Диаграмма распределения участников группы по этапам

В больших организациях обычно работает много специалистов, которые задействуются в проекте по мере необходимости. Конечно, такой подход может создать определенные проблемы для менеджеров проектов. Например, если специалист занят в проекте, который задерживается, это может создать прямые сложности для других проектов, где он также должен участвовать.

Первоначальный график работ неизбежно содержит какие-нибудь ошибки или недоработки. По мере реализации проекта рассчитанные оценки длительности выполнения этапов работ должны сравниваться с реальными сроками выполнения этих этапов. Результаты сравнения должны использоваться в качестве основы для пересмотра графика работ еще не реализованных этапов проекта, в частности для того, чтобы попытаться уменьшить длительность этапов критического пути.

2 Функциональное проектирование

IDEF0 (Integrated Definition Function Modeling) - методология функционального моделирования. В основе IDEF0 методологии лежит понятие блока, который отображает некоторую бизнес-функцию. Четыре стороны блока имеют разную роль: левая сторона имеет значение "входа", правая - "выхода", верхняя - "управления", нижняя - "механизма" (рис. 4).

Взаимодействие между функциями в IDEF0 представляется в виде дуги, которая отображает поток данных или материалов, поступающий с выхода одной функции на вход другой. В зависимости от того, с какой стороной блока связан поток, его называют соответственно "входным", "выходным", "управляющим".

В IDEF0 реализованы три базовых принципа моделирования процессов:

- принцип функциональной декомпозиции;
- принцип ограничения сложности;
- принцип контекста.

При определении главной бизнес-функции необходимо всегда иметь ввиду цель моделирования и точку зрения на модель. Одно и то же предприятие может быть описано по-разному, в зависимости от того, с какой точки зрения его рассматривают: директор предприятия и налоговой инспектор видят организацию совершенно по-разному.

Контекстная диаграмма играет еще одну роль в функциональной модели. Она "фиксирует" границы моделируемой бизнес-системы, определяя то, как моделируемая система взаимодействует со своим окружением. Это достигается за счет описания дуг, соединенных с блоком, представляющим главную бизнес-функцию.

На рис. 4 и рис. 5 представлен пример построения функциональной диаграммы, описывающей изготовление изделия.

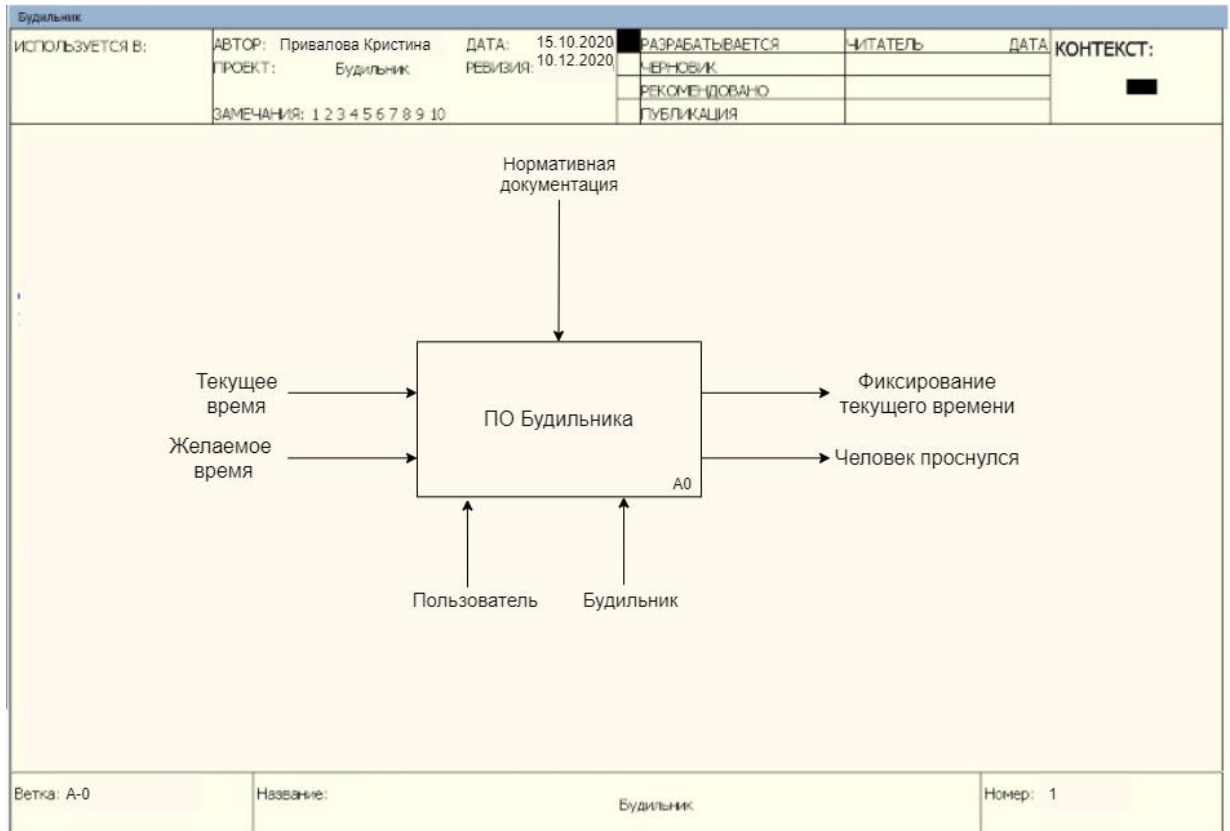


Рисунок 4- Контекстная диаграмма

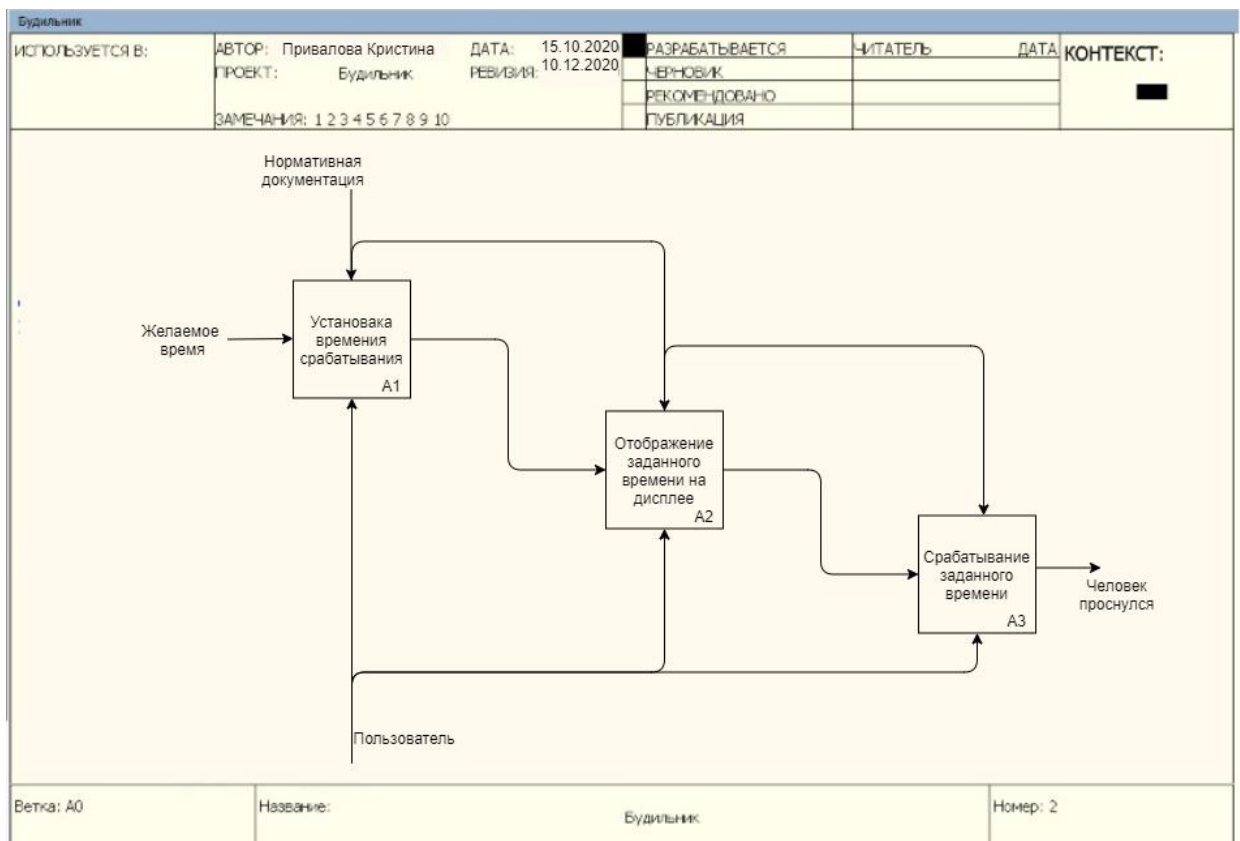


Рисунок 5 – Первый уровень декомпозиции

Для описания логики взаимодействия информационных потоков наиболее подходит IDEF3, называемая также workflow diagramming - методологией моделирования, использующая графическое описание информационных потоков, взаимоотношений между процессами обработки информации и объектов, являющихся частью этих процессов.

Диаграммы Workflow могут быть использованы в моделировании бизнес-процессов для анализа завершенности процедур обработки информации. С их помощью можно описывать сценарии действий сотрудников организации, например последовательность обработки заказа или события, которые необходимо обработать за конечное время. Каждый сценарий сопровождается описанием процесса и может быть использован для документирования каждой функции.

IDEF3 - это метод, имеющий основной целью дать возможность аналитикам описать ситуацию, когда процессы выполняются в определенной последовательности, а также описать объекты, участвующие совместно в одном процессе.

Техника описания набора данных IDEF3 является частью структурного анализа. В отличие от некоторых методик описаний процессов IDEF3 не ограничивает аналитика чрезмерно жесткими рамками синтаксиса, что может привести к созданию неполных или противоречивых моделей. В IDEF3 декомпозиция используется для детализации работ.

Методология IDEF3 позволяет декомпонировать работу многократно, т.е. работа может иметь множество дочерних работ. Это позволяет в одной модели описать альтернативные потоки. Возможность множественной декомпозиции предъявляет дополнительные требования к нумерации работ. Так, номер работы состоит из номера родительской работы, версии декомпозиции и собственного номера работы на текущей диаграмме.

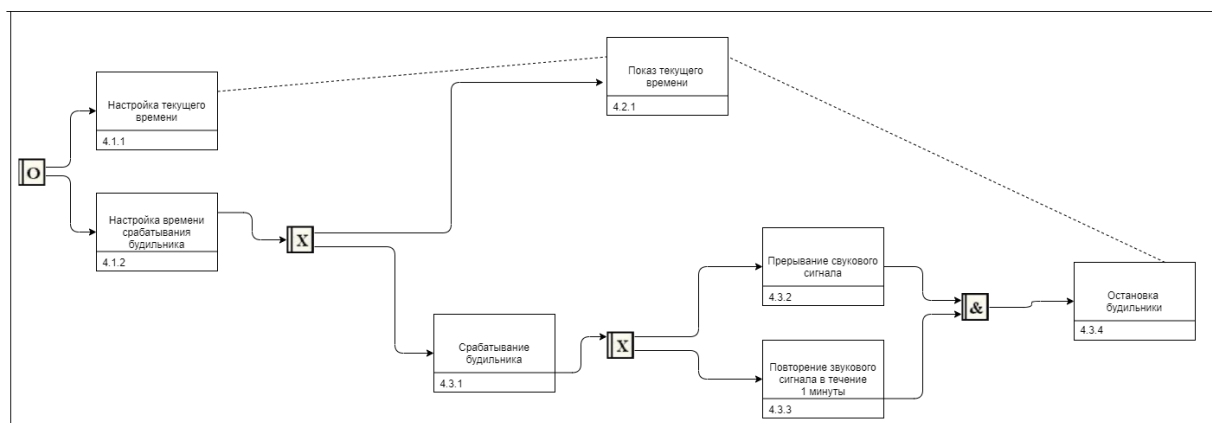


Рисунок 6 – Декомпозиция функционального блока «Управление пользователями» (IDEF3)

3 Диаграмма потоков данных (DFD)

Диаграмма потоков данных DFD (DataFlowDiagrams) - "это методология графического структурного анализа, описывающая внешние по отношению к системе ресурсы и адресаты данные, логические функции, потоки данных и хранилища данных, которая реализуется в Access. Диаграмма DFD является одним из основных инструментов структурного анализа и проектирования информационных систем, существовавших до расширения UML.» В результате декомпрессии системы "Будильник" была получена следующая диаграмма DFD (рис.).

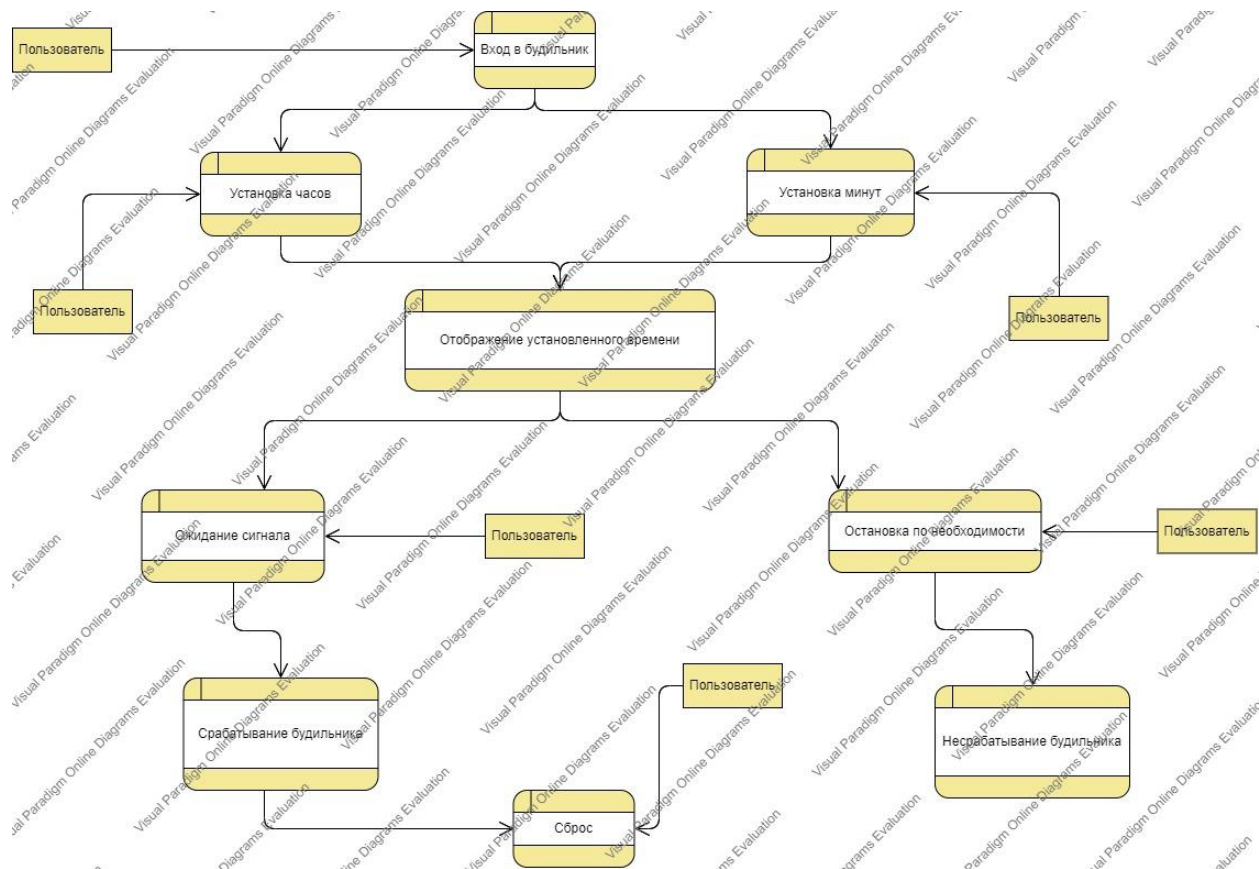


Рисунок 7 – Диаграмма DFD-системы «Будильник»

Внешняя сущность этой системы-пользователь. Система содержит процессы: ввод в систему бухгалтерского учета, составление заказов для поставщиков, проверка наличия товара, выполнение / невыполнение заказов, редактирование информации, добавление товара, добавление информации о

поставщиках, формирование справки, получение обзора. В системе также есть хранилище данных: время звпуска будильника

4 Объектно-ориентированное проектирование

UML (англ. Unified Modeling Language — "унифицированный язык моделирования") язык графического описания объектов моделирования в области разработки программного обеспечения для моделирования бизнес-процессов, проектирования систем и отображения организационной структуры.

UML-это язык широкого профиля, это открытый стандарт, который использует графические обозначения для создания абстрактной модели системы, называемой моделью UML. UML был создан для идентификации, визуализации, проектирования и документирования в первую очередь программных систем. UML-это не язык программирования, но на основе UML-моделей можно генерировать код.

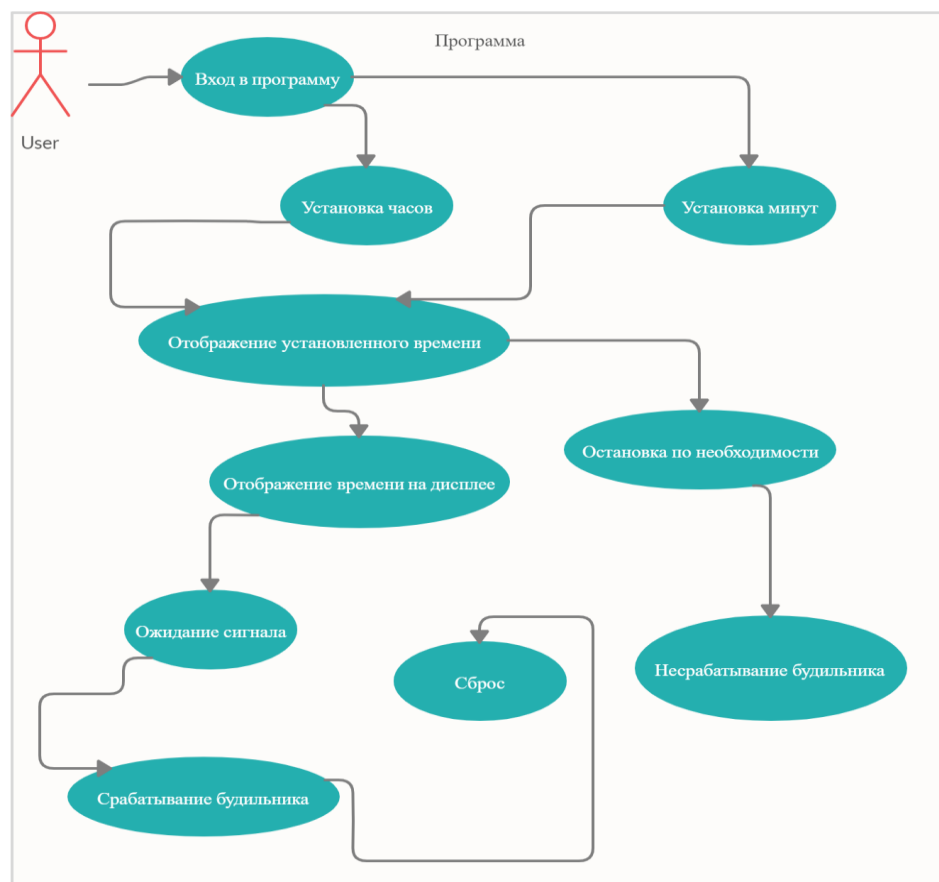


Рисунок 8 – UML-диаграмма системы «Будильник»

Диаграмма последовательности отражает поток событий, происходящих в рамках варианта использования. Все действующие лица показаны в верхней части диаграммы. Стрелки соответствуют сообщениям, передаваемым между действующим лицом и объектом или между объектами для выполнения требуемых функций. На диаграмме последовательности объект изображается в виде прямоугольника, от которого вниз проведена пунктирная вертикальная линия. Эта линия называется линией жизни (lifeline) объекта. Она представляет собой фрагмент жизненного цикла объекта в процессе взаимодействия. Каждое сообщение представляется в виде стрелки между линиями жизни двух объектов. Сообщения появляются в том порядке, как они показаны на странице сверху вниз. Каждое сообщение помечается как минимум именем сообщения. При желании можно добавить также аргументы и некоторую управляющую информацию. Можно показать самоделегирование (self-delegation) – сообщение, которое объект посылает самому себе, при этом стрелка сообщения указывает на ту же самую линию жизни.

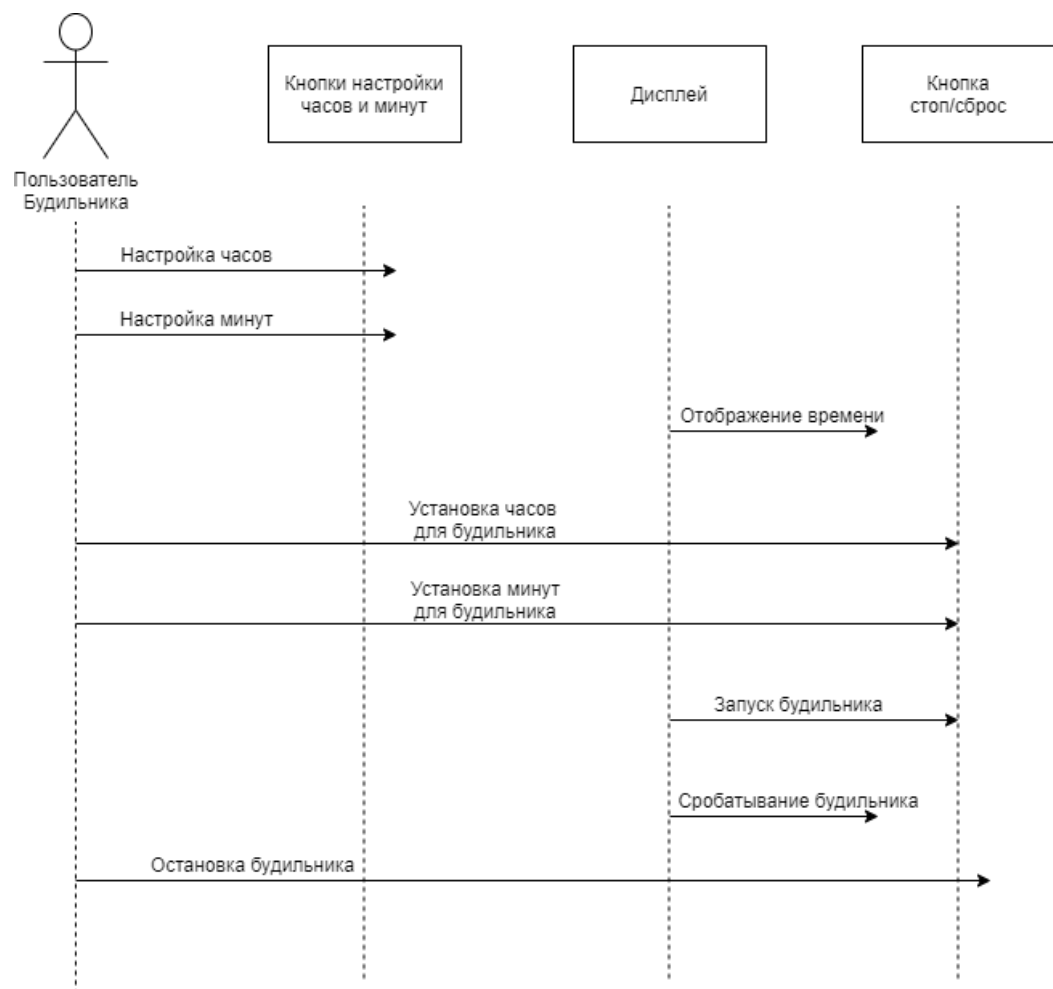


Рисунок 9 – Диаграмма последовательности

Диаграммы кооперации отображают поток событий через конкретный сценарий варианта использования, упорядочены по времени, а

кооперативные диаграммы больше внимания заостряют на связях между объектами. На диаграмме кооперации представлена вся та информация, которая есть и на диаграмме последовательности, но кооперативная диаграмма по-другому описывает поток событий. Из нее легче понять связи между объектами, однако, труднее уяснить последовательность событий. На кооперативной диаграмме так же, как и на диаграмме последовательности, стрелки обозначают сообщения, обмен которыми осуществляется в рамках данного варианта использования. Их временная последовательность указывается путем нумерации сообщений.

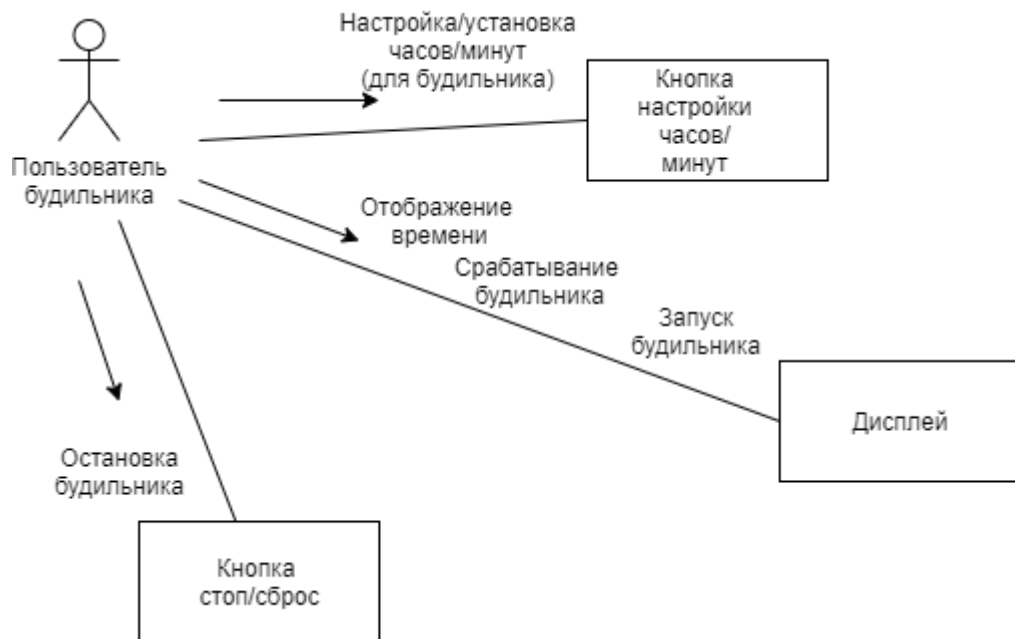


Рисунок 10 – Диаграмма кооперации

Диаграмма размещения (deployment diagram) отражает физические взаимосвязи между программными и аппаратными компонентами системы. Она является хорошим средством для того, чтобы показать маршруты перемещения объектов и компонентов в распределенной системе. Каждый узел на диаграмме размещения представляет собой некоторый тип вычислительного устройства – в большинстве случаев, часть аппаратуры. Эта аппаратура может быть простым устройством или датчиком, а может быть и мейнфреймом. Диаграмма размещения показывает физическое расположение сети и местонахождение в ней различных компонентов. Диаграмма размещения используется менеджером проекта, пользователями, архитектором системы и эксплуатационным персоналом, чтобы понять физическое размещение системы и расположение ее отдельных подсистем.

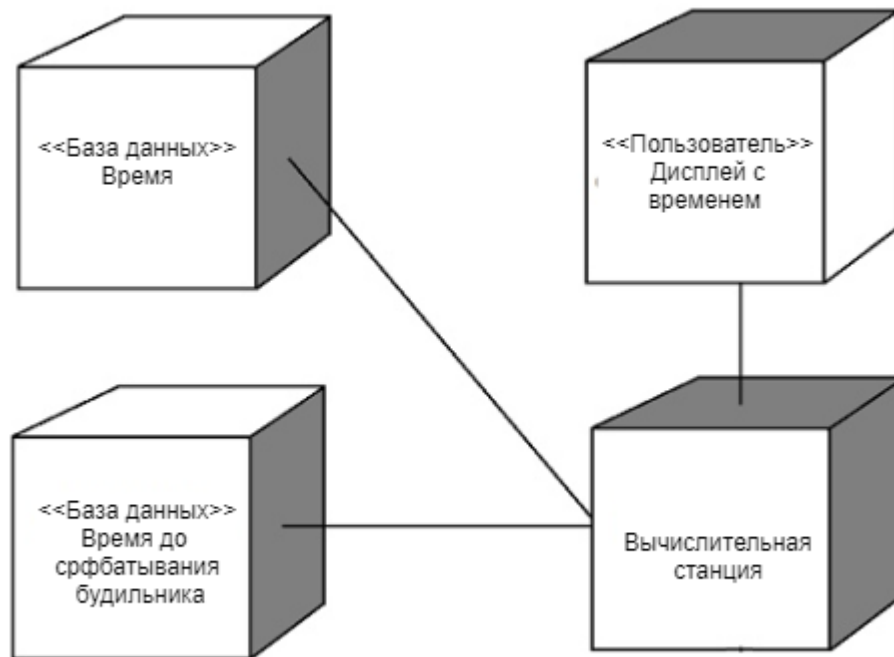


Рисунок 11 – Диаграмма размещений

Диаграммы компонентов показывают, как выглядит модель на физическом уровне. На них изображены компоненты программного обеспечения и связи между ними. При этом на такой диаграмме выделяют два типа компонентов: исполняемые компоненты и библиотеки. Каждый класс модели (или подсистема) преобразуется в компонент исходного кода. После создания они сразу добавляются к диаграмме компонентов. Между отдельными компонентами изображают зависимости, соответствующие зависимостям на этапе компиляции или выполнения программы. Диаграммы компонентов применяются теми участниками проекта, кто отвечает за компиляцию системы. Из нее видно, в каком порядке надо компилировать компоненты, а также какие исполняемые компоненты будут созданы системой. На такой диаграмме показано соответствие классов реализованным компонентам. Она нужна там, где начинается генерация кода.

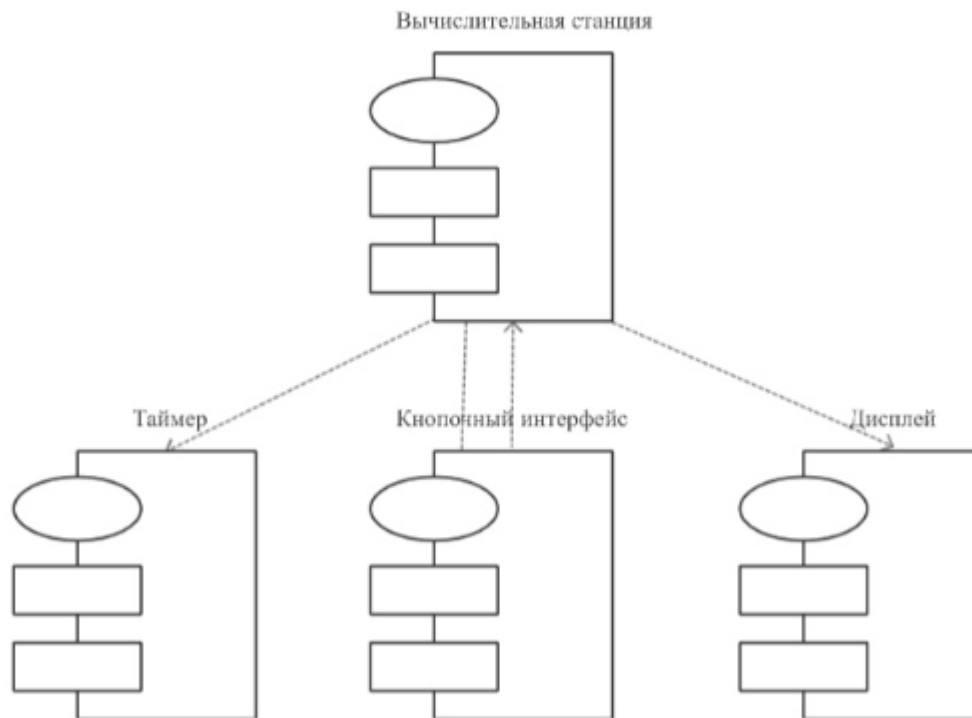
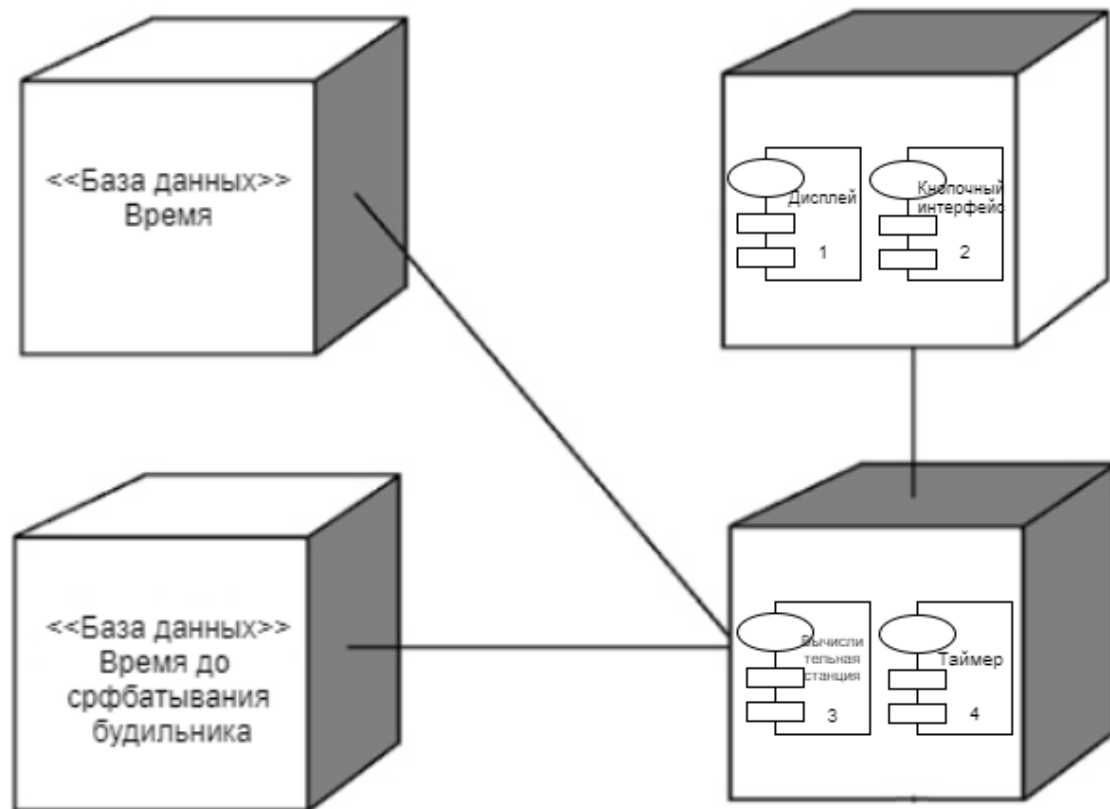


Рисунок 12 – Диаграмма компонентов

В некоторых случаях допускается размещать диаграмму компонентов на диаграмме развертывания. Это позволяет показать какие компоненты выполняются и на каких

узлах.



1.Дисплей 2. Кнопочный интерфейс 3. Вычислительная станция 4. Таймер

Рисунок 13 – Объединение диаграмм компонентов и развертывания

5 EPC

Разработка технологической цепочки (EPC-диаграмма, англ. event-driven process chain) - это тип блок-схемы, используемый для бизнес-моделирования. EPC может использоваться для настройки системы планирования ресурсов предприятия (ERP), а также для улучшения бизнес-процессов.

Организации используют диаграммы EPC для планирования рабочих процессов бизнес-процессов. Существует целый ряд инструментов для создания диаграмм EPC, некоторые из этих инструментов поддерживают формат обмена данными EPC независимо от языка брендинга инструмента EPMML. Диаграммы EPC используют символы нескольких видов для отображения структуры потока управления (последовательности решений, функций, событий и других элементов) бизнес-процесса.

Метод ЭПК был разработан Аугусто-Вильгельмом Шеером в рамках работ по созданию АРИС в начале девяностых годов. Многие организации используются для моделирования, анализа и реорганизации бизнес-процессов.

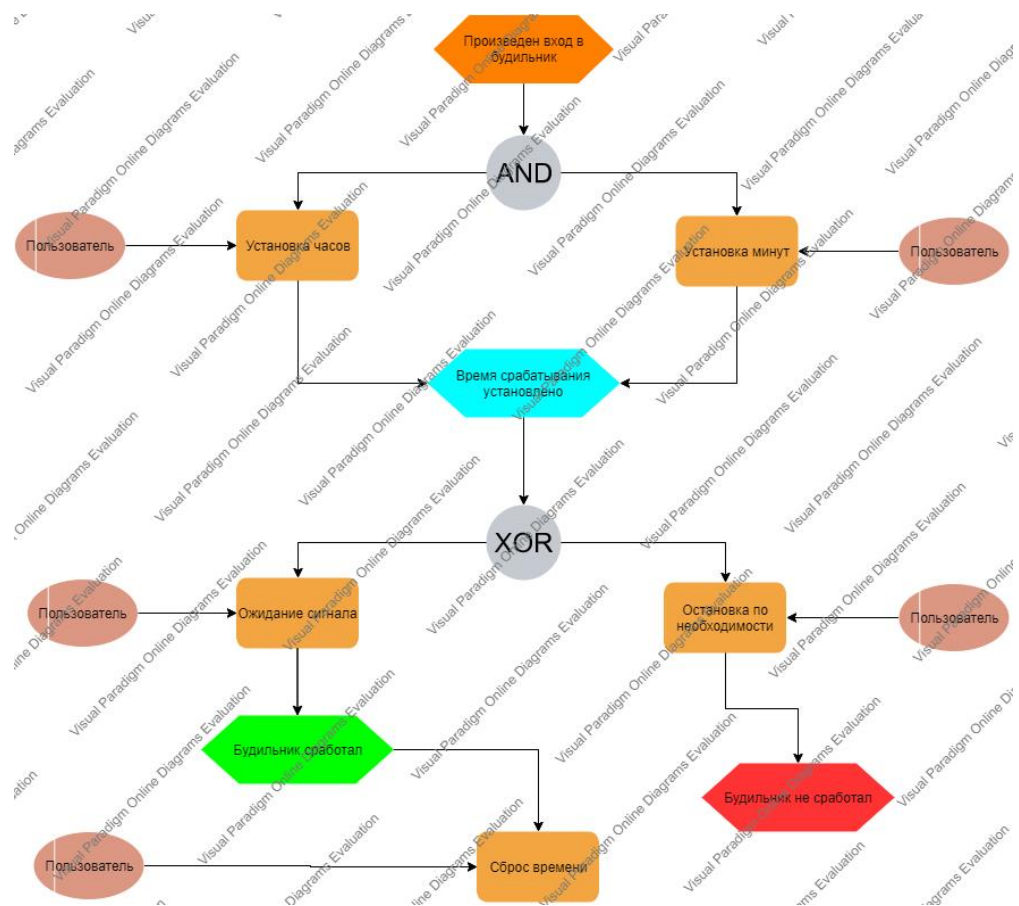


Рисунок 14 – EPC-диаграмма системы «Будильник»

6 BPMN

Спецификация BPMN описывает условные метки и их описание в XML для отображения бизнес-процессов в виде диаграмм бизнес-процессов. BPMN ориентируется как на технических специалистов, так и на корпоративных пользователей. Язык использует базовый набор интуитивных функций, которые позволяют определять сложные семантические предложения. Кроме того, спецификация BPMN определяет, как диаграммы, описывающие бизнес-процесс, могут быть преобразованы в исполняемые модели.

Основной задачей BPMN является создание стандартного набора условных меток, понятных всем бизнес-пользователям. Бизнес-пользователи включают в себя бизнес-аналитиков, которые создают и совершенствуют процессы, технических разработчиков, ответственных за реализацию процессов, и менеджеров, которые следят за процессами и управляются ими. Таким образом, BPMN призван служить связующим звеном между этапом проектирования бизнес-процесса и этапом внедрения. На рис. 10 представлена схема работы системы поддержки BPMN будильника.

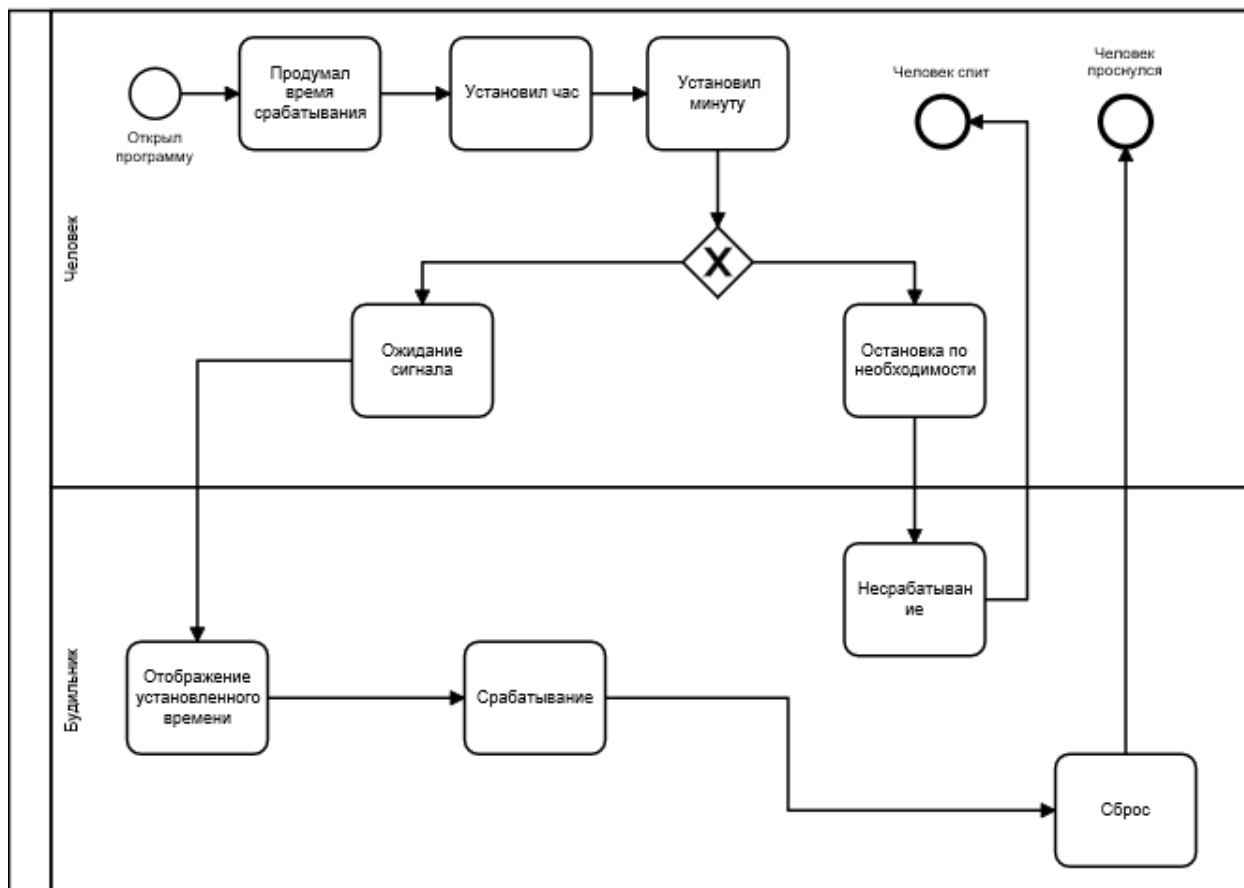


Рисунок 15 – Диаграмма BPMN «As-Is» и «To be»

Заключение

В результате этого курса был разработан проект "Будильник" на языке высокого уровня C#, который позволяет проиллюстрировать работу всех его компонентов. В ходе выполнения были получены различные графики, которые подробно описывают все процессы, происходящие в системе. Несмотря на размеры карт, они имеют довольно простую структуру, понятную даже для жителей города.

При создании графов использовались основные правила и принципы моделирования, содержащие графическое представление объектов и отношений между ними, иерархическую сеть, а также названия, отражающие обозначение того или иного факта или взаимодействия.

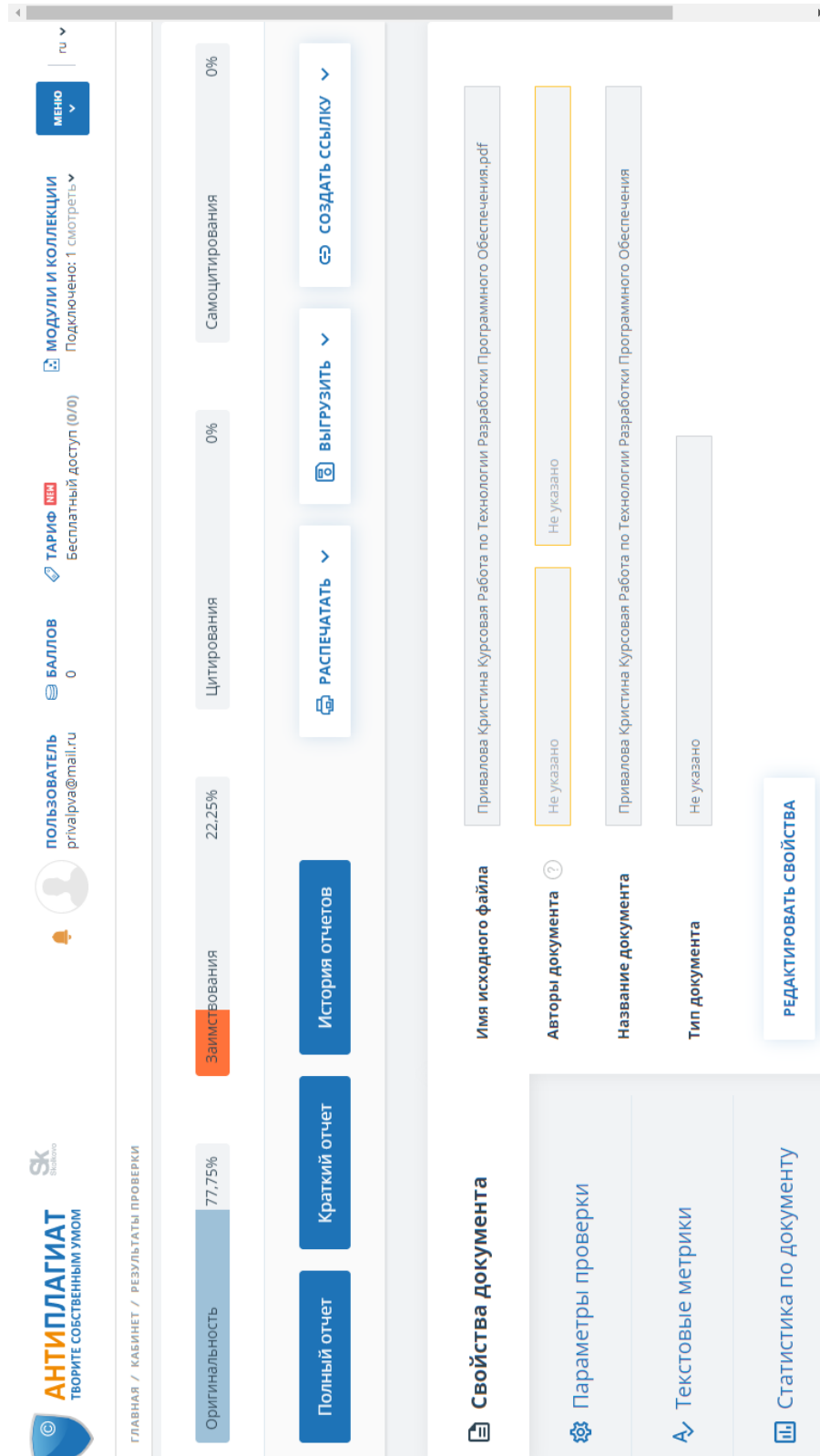
Благодаря детальному изучению анализа проекта с использованием чертежей предложения, полученных в процессе разработки, можно добиться того, что полученный "будильник" полностью соответствует современным стандартам качества и способен выполнять самые разнообразные функции, которые необходимы во всех крупных и малых коммерческих предприятиях.

Были получены важные знания и практические навыки в использовании объектно-ориентированного языка программирования C#, а также опыт построения различных графов, описывающих процессы, происходящие в системе.

Список использованных источников

1. Ларман, Крэг. Применение UML 2.0 и шаблонов проектирования. Введение в объектно-ориентированный анализ, проектирование и итеративную разработку / Крэг Ларман. - Москва: Гостехиздат, 2017. - 736 с.
2. Роберт А. Максимчук. UML для простых смертных / Роберт А.
3. SysAna – Требования к системе: классификация FURPS+ [Электронный ресурс]: - Режим доступа: <https://sysana.wordpress.com/2010/09/16/furps/>. (Дата обращения 15.10.2020)
4. Comindware – Нотация BPMN 2.0 [Электронный ресурс]: - Режим доступа: <https://comindware.com/ru/blog-нотация-bpmn-2-0-элементы-и-описание/> (Дата обращения 07.11.2020)
5. GitHub – yarajtf/intercom. [Электронный ресурс]: - Режим доступа: <https://github.com/yarajtf/intercom> (Дата обращения 27.09.2020).
6. Йордон, Эдвард. Объектно-ориентированный анализ и проектирование систем / Эдвард Йордон, Карл Аргила. - М.: ЛОРИ, 2014. - 264 с.
7. Habr – Что такое DFD? [Электронный ресурс]: - Режим доступа: <https://habr.com/ru/company/trinion/blog/340064/>.

Проверка на антиплагиат



Приложение Б

Листинг

```
using System;

using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Media;

namespace Alarm_clock1
{

    public partial class Form1 : Form
    {
        public Form1()
        {
            Timer timer1 = new Timer();
            InitializeComponent();
        }

        SoundPlayer sp = new SoundPlayer("D:\\Dram.wav");
        bool b = false;
        private void maskedTextBox1_MaskInputRejected(object sender, MaskInputRejectedEventArgs e)
        {

        }

        private void label1_Click(object sender, EventArgs e)
        {
            label1.Text = DateTime.Now.Hour.ToString("00") + ":" +
            DateTime.Now.Minute.ToString("00") + ":" +
            DateTime.Now.Second.ToString("00");
        }

        private void timer1_Tick(object sender, EventArgs e)
        {

```



```
}
```

```
private void Form1_Load(object sender, EventArgs e)
{
    button2.Enabled = false;
    timer1.Interval = 1000;
    timer1.Tick += new EventHandler(timer1_Tick);
    timer1.Start();
}
```

```
private void timer1_Tick_1(object sender, EventArgs e)
{
    label1.Text = DateTime.Now.Hour.ToString("00") + ":" +
DateTime.Now.Minute.ToString("00") + ":" +
DateTime.Now.Second.ToString("00");
}
```

```
private void button1_Click(object sender, EventArgs e)
{
    label2.Text = maskedTextBox1.Text;
    timer2.Start();
    maskedTextBox1.Visible = false;
    button1.Text = "Остановить";
    b = true;
}
```

```
private void button2_Click(object sender, EventArgs e)
{
    sp.Stop();
    button2.Enabled = false;
    maskedTextBox1.Visible = true;
    button1.Text = "Завести";
    b = false;
}
```

```
private void timer2_Tick(object sender, EventArgs e)
{
    if (label1.Text == label2.Text + ":00")
```

```

    {
        button2.Enabled = true;
        sp.Play();
    }
}

```

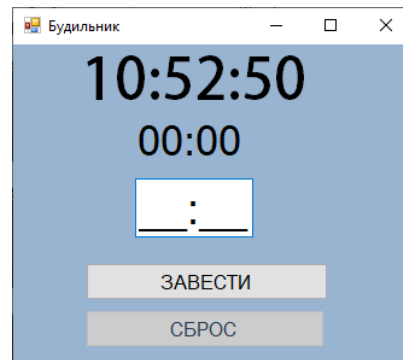
```

private void label2_Click(object sender, EventArgs e)
{
    if (b == true)
    {
        label2.Text = "00:00";
        timer2.Stop();
        maskedTextBox1.Visible = true;
        button1.Text = "Завести";
        b = false;
    }
}
}
}

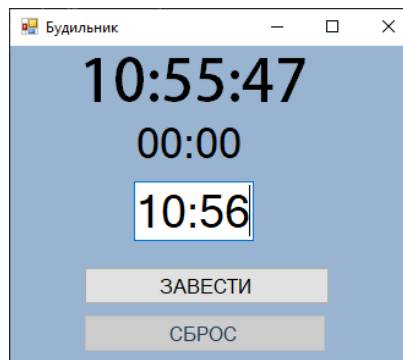
```

Приложение В

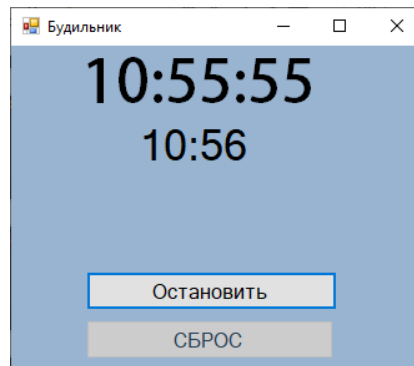
Тестирование



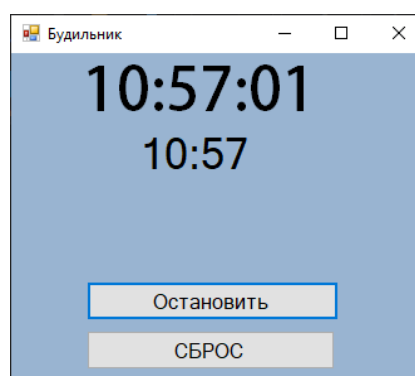
Начальное окно



Заведение будильника



Работа будильника



Момент срабатывания