МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное
учреждение высшего образования
«Национальный исследовательский университет «МЭИ»

Институт ИнЭИ
Кафедра БИТ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(бакалаврская работа)

Направление 09.03.03 – Прикладная информатика
(код и наименование)

Направленность (профиль) и управлении **Прикладная информатика в экономике**

Форма обучения **Заочная**
(очная/очно-заочная/заочная)

Тема: **Разработка информационной системы каталога бактерий**

Студент **ИЭз-166-17** **Сизова А.А.**
группа подпись фамилия и инициалы

Научный руководитель **доцент** **Жнякин О.В.**
уч. степень должность подпись фамилия и инициалы

Консультант
уч. степень должность подпись фамилия и инициалы

Консультант
уч. степень должность подпись фамилия и инициалы

«Работа допущена к защите»

Зав. кафедрой **к.т.н.** **доцент** **Невский А. Ю.**
уч. степень звание подпись фамилия и инициалы

Дата

Москва, 2021

МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Национальный исследовательский университет «МЭИ»

Институт ИнЭИ
Кафедра БИТ

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
(бакалаврскую работу)

Направление 09.03.03 – Прикладная информатика
(код и наименование)

Направленность (профиль) **Прикладная информатика в экономике и управлении**

Форма обучения **Заочная**
(очная/очно-заочная/заочная)

Тема: **Разработка информационной системы каталога бактерий**

Студент	**ИЭз-166-17**			**Сизова А.А.**
	группа		подпись	фамилия и инициалы
Научный руководитель		**доцент**		**Жнякин О. В.**
	уч. степень	должность	подпись	фамилия и инициалы
Консультант				
	уч. степень	должность	подпись	фамилия и инициалы
Консультант				
	уч. степень	должность	подпись	фамилия и инициалы
Зав. кафедрой	**к.т.н.**	**доцент**		**Невский А.Ю.**
	уч. степень	звание	подпись	фамилия и инициалы

Место выполнения работы ФГБОУ ВО «НИУ «МЭИ»

СОДЕРЖАНИЕ РАЗДЕЛОВ ЗАДАНИЯ И ИСХОДНЫЕ ДАННЫЕ

Аннотация
Перечень сокращений
Введение
Глава 1. Аналитическая часть
Обоснование актуальности задачи создания каталога бактерий
Актуальность темы создания каталога бактерий
Анализ существующих разработок по данной теме
Обоснование выбора и описание методологии разрабатываемого ПО
Обоснование выбора и описание инструментальных средств разработки ПО

Глава 2. Проектная часть
Функциональная диаграмма «Разработка информационной системы»
Диаграмма потоков данных каталога бактерий
Проектирование схемы базы данных каталога бактерий
Интерфейс информационной системы каталога бактерий и демонстрация работы
Глава 3. Экономическая часть
Данные для расчета полной себестоимости
Расчет затрат на материалы
Расчет затрат на электроэнергию и амортизацию оборудования
Расчет расходов на заработную плату
Расчет полной себестоимости программы
Расчет цены разработки программы
Заключение
Список литературы
Приложение

ПЕРЕЧЕНЬ ГРАФИЧЕСКОГО МАТЕРИАЛА

Количество листов 78

Количество слайдов в презентации 16

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Статья про биологические базы данных [Электронный ресурс] - https://studme.org/307310/informatika/biologicheskie_bazy_dannyh
2. Статья про модели разработки [Электронный ресурс] - https://habr.com/ru/company/edison/blog/269789/
3. Статья по Microsoft SQL Server 2019 [Электронный ресурс] - https://softline.ru/about/blog/10-prichin-pereyti-na-microsoft-sql-server-2019
4. Документация по C# [Электронный ресурс] - https://docs.microsoft.com/ru-ru/dotnet/csharp/
5. Албахари Бен C# 7.0. Справочник. Полное описание языка / Албахари Бен, Албахари Джозеф
6. Методология функционального моделирования IDEF0 [Электронный ресурс] - https://nsu.ru/smk/files/idef.pdf

7. Цуканова О.А. Методология и инструментарий моделирования бизнес-процессов. – СПб.: Университет ИТМО, 2015. – 100 с
8. Сайт нуклеотидной базы данных EMBL [Электронный ресурс] - https://www.ebi.ac.uk/submission/
9. Сайт нуклеотидной базы данных DDBJ [Электронный ресурс] - https://www.ddbj.nig.ac.jp/data-categories-e.html
10. Статья о Visual Studio 2019 [Электронный ресурс] - https://docs.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2019

ГРАФИК ВЫПОЛНЕНИЯ РАЗДЕЛОВ ВКР

Дата начала работы над ВКР: 20.09.2020.

№	Содержание разделов	Трудоемкость	Срок выполнения
1	Исследовательская обзорная часть	30%	10.10.2020
2	Проектирование системы	40%	01.12.2020
3	Программная реализация системы, ее отладка и тестирование	15%	20.12.2020
4	Оценка затрат на разработку и эффекта от внедрения	5%	27.12.2020
5	Оформление ВКР, презентации, подготовка к защите	10%	15.01.2021

Примечания:

1. Задание брошюруется вместе с выпускной работой после титульного листа (страницы задания имеют номера 2, 3).
2. Отзыв руководителя, рецензия(и), отчет о проверке на объем заимствований и согласие студента на размещение работы в открытом доступе вкладываются в конверт (файловую папку) под обложкой работы.

АННОТАЦИЯ

Тема: «Разработка информационной системы каталога бактерий».

Объем дипломной работы: 78 страниц, 26 рисунков, 33 таблиц, 10 источников.

Процесс учета микроорганизмов, находящихся на хранении очень важен для коллекционной деятельности. В связи с этим разрабатываются различные программы и сервисы для каталогизации коллекции микроорганизмов. У каждой организации имеются свои определенные задачи, которые требуют разного подхода для удобного функционирования, поэтому не любое программное обеспечение может подойти. По этой причине возникает потребность создания своего каталога для учета микроорганизмов, учитывающего запросы организации.

ANNOTATION

Subject: “Development of an information system for the catalog of bacterial”.

The volume of diploma work: 78 pages, 26 pictures, 33 tables, 10 sources.

The process of accounting for microorganisms in storage is very important for collection work. In this regard, various programs and services are being developed to catalog the collection of microorganisms. Each organization has its own specific tasks that require a different approach for convenient operation, so not every software can be suitable. For this reason, there is a need to create your own catalog for accounting for microorganisms, taking into account the requests of the organization.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

БД — база данных

ИС — информационная система

ПО — программное обеспечение

СУБД — система управления базами данных

DFD — Data Flow Diagrams

IDEF0 – Icam DEFinition for Function Modeling

SQL – Structured Query Language

ФБУН ГНЦ ПМБ – Федеральной Бюджетное Учреждение Науки Государственный Научный Центр Прикладной Микробиологии и Биотехнологий

ГКПМ-Оболенск – Государственная Коллекция Патогенных Микроорганизмов Оболенск

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Основной функцией коллекционной деятельности является сбор и хранение природных, типовых, референсных, производственных, учебных и контрольных штаммов бактерий, в том числе генетически модифицированных. Согласно приказу Федеральной службы по надзору в сфере защиты прав потребителей и благополучия человека от 27.07.2010 г. № 297 на базе ФБУН ГНЦ ПМБ создана Государственная коллекция патогенных микроорганизмов и клеточных культур («ГКПМ-Оболенск»).

На хранении в коллекционном фонде «ГКПМ-Оболенск» находится более 8000 штаммов бактерий. Сохранение микроорганизмов подразумевает под собой постоянную работу по поддержанию в жизнеспособном состоянии фонда коллекции, а также осуществление микробиологических и молекулярно-генетических работ. Как один из методов исследования бактериальных культур используется метод полногеномного секвенирования. Использование этого метода позволяет определить полную последовательность генома микроорганизма, обнаруживать гены, ответственные за изменение клинических свойств штаммов, охарактеризовывать штаммы с атипичными генотипами, а также идентифицировать и описывать представителей, принадлежащих к ранее не известным таксономическим группам.

Результаты полногеномного секвенирования представляют собой цифровую информацию, которые содержат риды (короткие последовательности, полученные путем секвенирования ДНК) и сборки геномов (длинные последовательности нуклеотидов, составляющие хромосомы или плазмиды). Хранение и анализ полученных данных требует больших объемов дискового пространства.

Для осуществления учета результатов исследования бактерий, входящих в коллекционный фонд необходимо создание специальной информационной системы, позволяющей автоматизировать работу с данными для каждого конкретного микроорганизма.

Объектом исследования является «ГКПМ-Оболенск».

Предметом исследования является информационная система каталога бактерий.

Цель работы: создание информационной системы для работы с данными исследования бактерий, входящих в коллекционный фонд «ГКПМ-Оболенск».

Задачи работы:

1) создания каталога для учета микроорганизмов, находящихся на хранении в «ГКПМ-Оболенск»;
2) учет проведения раундов секвенирования;
3) учет метрических показателей полногеномного секвенирования;
4) формирование паспорта микроорганизмов, находящихся на хранении в «ГКПМ-Оболенск»;
5) учет депонированной информации, полученной методом полногеномного секвенирования геномов штаммов бактерий в базы данных сервиса NCBI.

1. Аналитическая часть

1.1. Обоснование актуальности задачи создания каталога бактерий

1.1.1. Актуальность темы создания каталога бактерий

Процесс учета микроорганизмов, находящихся на хранении очень важен для коллекционной деятельности. В связи с этим разрабатываются различные программы и сервисы для каталогизации коллекции микроорганизмов. У каждой организации имеются свои определенные задачи, которые требуют разного подхода для удобного функционирования, поэтому не любое программное обеспечение может подойти. По этой причине возникает потребность создания своего каталога для учета микроорганизмов, учитывающего запросы организации.

В «ГКПМ-Оболенск» структура каталога бактерий определяется следующей необходимой информацией: инвентарный номер, род, вид, вариант, название штамма, номер штамма, вид нуклеиновой кислоты, патогенность, форма депонирования, дата депонирования, номера в других коллекциях, авторы микроорганизма, авторы для депонирования полногеномного секвенирования, источник выделения, раунд секвенирования, платформа и модель секвенатора, метрические показатели полногеномного секвенирования, номера доступа в NCBI.

Требования, предъявляемые к каталогу «ГКПМ-Оболенск», должны учитывать деятельность, связанную с правовой охраной штаммов микроорганизмов, имеющих коммерческую и интеллектуальную ценность. По существующим в настоящее время правилам, при подаче заявки на изобретения, связанные с использованием микроорганизмов, обязательным требованием является депонированием штаммов в одной из государственных коллекций.

«ГКПМ-Оболенск» осуществляет депонирование штаммов бактерий, включая ПБА I-IV групп патогенности (биологической опасности); грибов, включая ПБА II-IV групп патогенности (биологической опасности); плазмид; бактериофагов; перевиваемых клеточных линий и гибридом.

В каталоге необходимо учитывать форму хранения каждого конкретного штамма микроорганизма. В «ГКПМ-Оболенск» осуществляется процедура депонирования штаммов микроорганизмов для целей национальной патентной процедуры и это должно быть отражено в характеристике штамма:

- хранение (штаммы, представляющие интерес для «ГКПМ-Оболенск» и учреждения в целом);
- гарантийное хранение (обеспечения сохранности штаммов, представляющих интерес для депозитора, осуществляется на платной договорной основе в течение срока, оговоренного при депонировании, «ГКПМ-Оболенск» сохраняет конфиденциальность информации о депонированном штамме в течение всего срока гарантийного хранения);
- депонирование для целей национальной патентной процедуры (штамм или способ его использования планируется для подачи заявки на патент Российской Федерации).

«ГКПМ-Оболенск» осуществляет депонирование штаммов бактерий I-IV групп патогенности (биологической опасности). Принадлежность к группам биологической опасности определяет соответствующий регламент работы с конкретным штаммом микроорганизма. Эту информацию необходимо учитывать при хранении, выдаче и транспортировке.

В каталоге «ГКПМ-Оболенск» необходимо учитывать следующие метрические показатели полногеномного секвенирования:

- количество ридов;
- количество оснований;
- количество контигов;
- размер полученного генома;
- размер наибольшего контига;
- покрытие;
- GC состав;

- N50.

Для депонирования информации, полученной методом полногеномного секвенирования геномов штаммов бактерий в базы данных сервиса NCBI проводится внутренняя комиссия ФБУН ГНЦ ПМБ для определения целесообразности. Депонирование результатов полногеномного секвенирования штаммов осуществляется только геномов, необходимых для публичного доступа.

1.1.2. Анализ существующих коллекций биологической информации

На сегодняшний день существуют сотни баз данных, например, базы данных нуклеотидных последовательностей, аминокислотных последовательностей, структурных особенностей белков, последовательностей РНК и другие. Каждая база имеет свой формат хранения данных, различную степень избыточности и взаимосвязи между родственными или аналогичными базами данных.

Базы данных коллекций биологической информации можно разделить на следующие типы [1]:

- архивные базы данных (за записи отвечает сам исследователь);
- курируемые базы данных (за записи отвечает куратор – эксперт из архивных баз);
- автоматические базы данных (записи генерируются компьютерными программами);
- производственные базы данных (записи создаются в результате компьютерной обработки данных из архивных и курируемых баз данных);
- интегрированные базы данных (объединяют информацию из разных баз данных).

GenBank, EMBL, DDBJ – три наиболее крупные и известные базы данных, содержащие биологическую информацию. (Таблица 1).

Таблица 1

Базы данных

Название	Институт	Страна	Ссылка	Тип
GenBank	NCBI	США	https://www.ncbi.nlm.nih.gov/	Архивные
EMBL	EBI	Великобритания	https://www.ebi.ac.uk/	Архивные
DDBJ	CIB	Япония	https://www.ddbj.nig.ac.jp/	Архивные

Банк данных GenBank (NCBI)

GenBank – база данных нуклеотидных последовательностей ДНК, РНК, аминокислотных последовательностей белков, снабженная литературными ссылками. GenBank курируется Национальным центром биотехнологической информации США, который входит в состав Национальных институтов здоровья в США. Интерфейс сайта, на базе которого работает база данных GenBank представлен на рисунке 1.

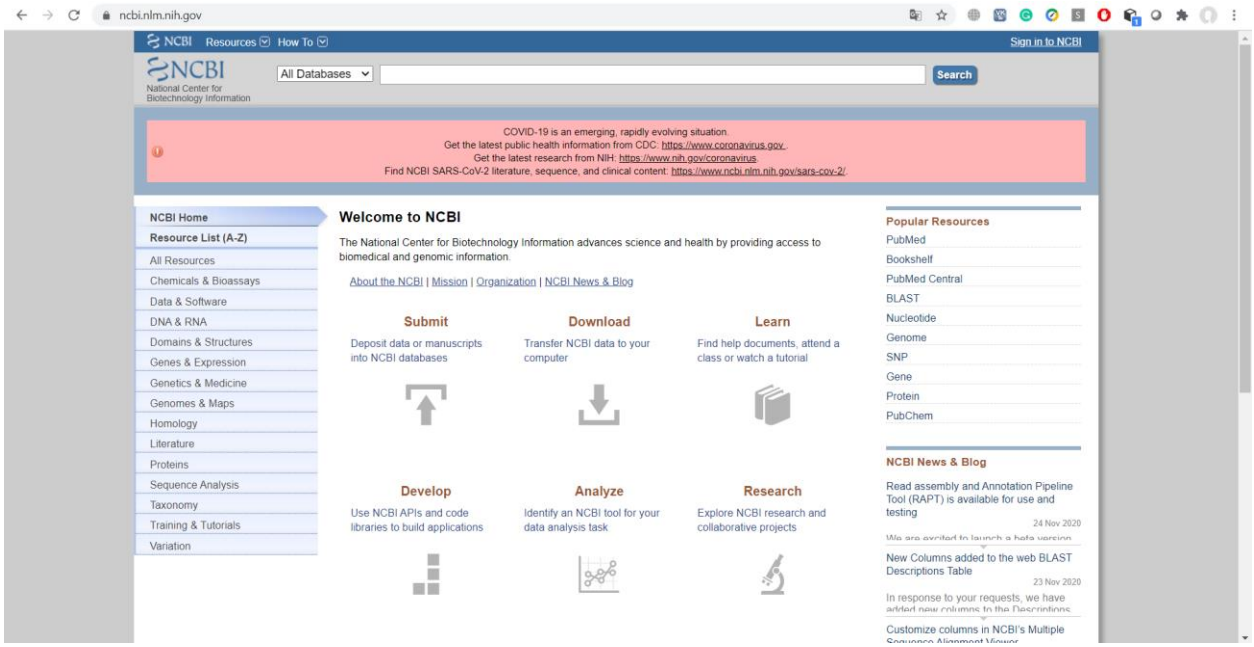


Рисунок 1 - Интерфейс сайта NCBI

В GenBank существует несколько разделов для хранения результатов высокопроизводительного секвенирования:

- genomes – раздел для хранения полных геномов прокариот и эукариот;

- WGS (whole genome shoutgun) – раздел для хранения сборки неполных геномов хромосом и плазмид прокариот или эукариот;
- TPA (Third Party Annotation) – раздел для хранения экспериментальных данных и результатов их обработки;
- TSA (Transcriptome Shotgun Assemble sequences) – раздел для хранения данных секвенирования транскриптомов;
- ENV (Environmental sample sequence) – раздел для хранения последовательностей образцов, которые были взяты из окружающей среды, но точный источник выделения неизвестен;
- EST (Expressed sequence tags) – раздел посвященный исследованиям по экспрессии генов;
- HTG (High-throughput genomic) – раздел для хранения масштабных геномных незавершенных данных, которые в будущем будут закончены;
- GSS (Genome survey sequences) – содержит необработанные данные, которые могут включать в себя 5’ и 3’ нетранслируемые области (UTR), части кодирующих областей и интроны. После обработки данные перемещаются в раздел соответствующего организма;
- CON (Contig records for assemblies of smaller records) – раздел для хранения длинных, но не целых последовательностей эукариотических хромосом.

EMBL (European Bioinformatics Institute – EBI)

EMBL – база данных нуклеотидных последовательностей Европейской молекулярно-биологической лаборатории, снабжена литературными, перекрестными ссылками на документы других баз данных. Интерфейс сайта, на базе которого работает база данных EMBL представлен на рисунке 2.

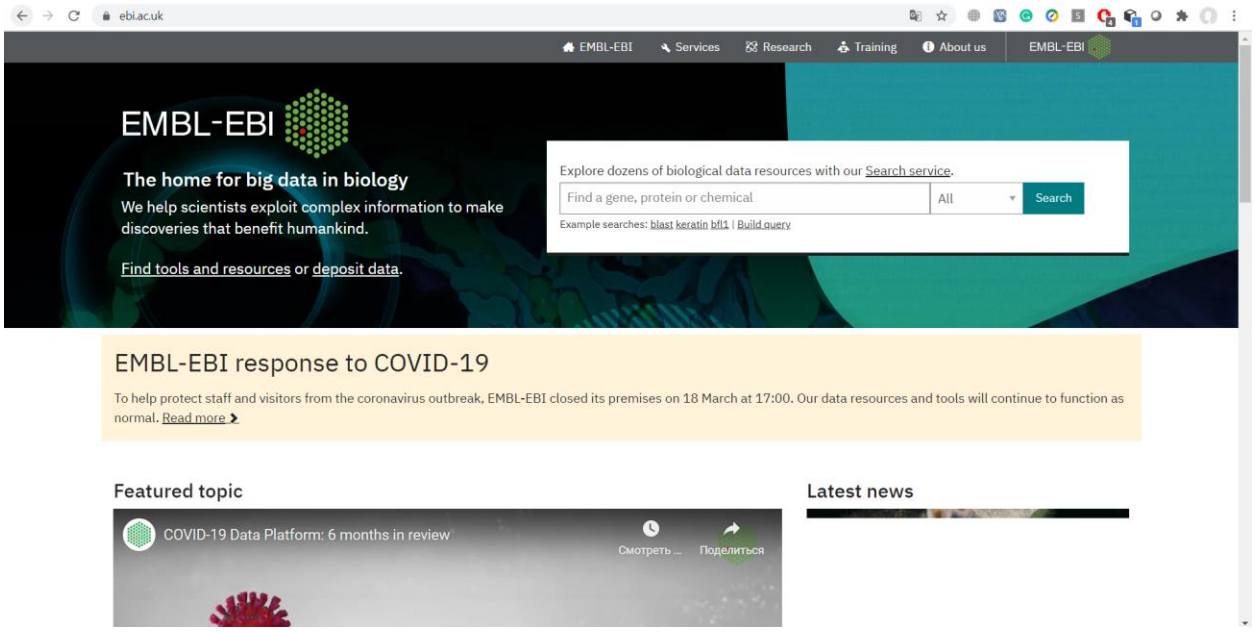


Рисунок 2 - Интерфейс сайта EBI

Хранилище данных EMBL содержит следующие разделы [8]:

- Array Express (functional genomics data – данные функциональной геномики) – раздел для хранения результатов данных полученных с помощью микрочипов и методом секвенирования нового поколения;
- BioModels (computational models – математические модели) – модели, описывающие поведения биологических систем;
- BioSamples (reference sample data – данные референсного образца) – раздел для хранения данных о биологических образцах, используемых в научных исследованиях;
- BioStudies (biological research data – данные биологических исследований) – раздел для хранения описаний биологических исследований;
- ChEBI (Chemical Entities of Biological Interest) – раздел для хранения данных химических соединений;
- EFO (Experimental Factor Ontology) – раздел для хранения данных экспериментов;

- EGA (European Genome-phenome Archive) – раздел для хранения генетических и фенотипических данных человека, имеющие ограниченный доступ;
- EMPIAR (Electron Microscopy Public Image Archive) – раздел для хранения данных электронной микроскопии;
- ENA (European Nucleotide Archive) – раздел для хранения нуклеотидных последовательностей;
- EVA (European Variation Archive) – раздел для хранения данных о генетических вариациях;
- IntAct (Molecular Interaction Database) – раздел для хранения данных о молекулярных взаимодействиях;
- IntEnz (Integrated relational Enzyme database) – раздел для хранения информации о номенклатуре ферментов;
- MetaboLights (database for metabolomics experiments and derived information) – раздел для хранения данных экспериментов по изучению метаболомов;
- Metagenomics (raw metagenomics sequence data and associated metadata to the European Nucleotide Archive (ENA) and analysis by MGnify) – раздел для хранения необработанных данных о метагеномных последовательностях и метаданные, связанные с другими разделами;
- wwPDB OneDep (System for Deposition, Biocuration, and Validation of Macromolecular Structures in the PDB Archive) – раздел для хранения данных электронной микроскопии, рентгеновской кристаллографии и ядерного магнитного резонанса);
- PRIDE (Proteomics Identifications database) – раздел для хранения данных о белках и пептидах;
- UniProt (Universal Protein Resource) – раздел для хранения белковых последовательностей.

DnA Databank of Japan – DDBJ (Center for Informatio Biology – CIB)

DDBJ – это база данных нуклеотидных последовательностей, созданная в Национальном институте генетики Японии. Интерфейс сайта, на базе которого работает база данных DDBJ представлен на рисунке 3.

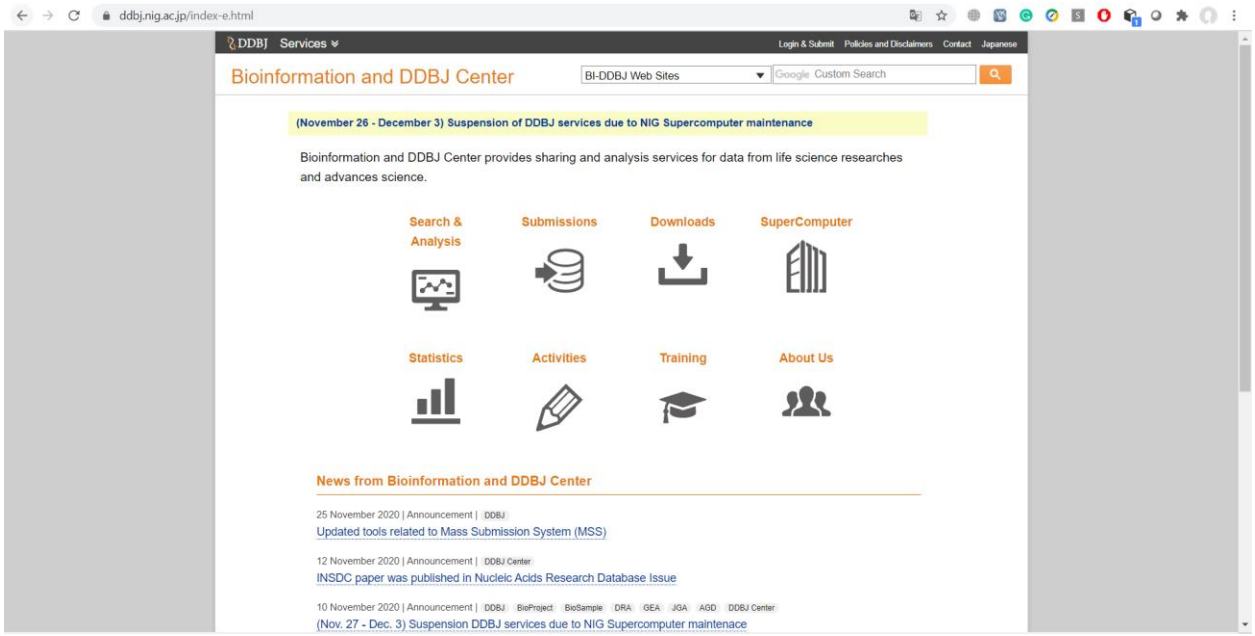


Рисунок 3 - Интерфейс сайта DDBJ

Существует несколько разделов для хранения данных [9]:

- ENV (Environmental sample sequence) – раздел для хранения последовательностей образцов, которые были взяты из окружающей среды, но источник выделения неизвестен;
- SYN (Synthetic Constructs) – раздел для хранения последовательностей, созданных искусственно;
- EST (Expressed sequence tags) – раздел для хранения данных исследований экспрессии генов;
- GSS (Genome survey sequences) – содержит необработанные данные, которые могут включать в себя 5’ и 3’ нетранслируемые области (UTR), части кодирующих областей и интроны. После обработки данные перемещаются в раздел соответствующего организма;
- HTC (High-throughput cDNA) – раздел для хранения последовательностей кДНК;

- HTG (High-throughput genomic) – раздел для хранения масштабных геномных незавершенных данных, которые в будущем будут закончены;
- CON (Contig records for assemblies of smaller records) – раздел для хранения длинных, но не целых последовательностей эукариотических хромосом;
- WGS (Whole Genome Shoutgun) – раздел для хранения сборки неполных геномов хромосом и плазмид прокариот и эукариот;
- TSA (Transcriptome Shotgun Assemble sequences) – раздел для хранения данных секвенирования транскриптомов;
- TLS (Targeted Locus Study) – раздел для хранения последовательностей целевых локусов, например, 16S рРНК;
- TPA (Third Party Annotation) – раздел для хранения экспериментальных данных и результатов их обработки.

Перечисленные три базы данных входят в консорциум International Nucleotide Sequence Database Collaboration. Каждый член консорциума собирает определенную информацию из доступных источников, ежедневно обмениваясь данными с другими членами.

Как правило, данные экспериментов депонируют в вышеперечисленные базы, чтобы они были общедоступными. Обычно это требования для публикации статей и оформления отчетных материалов по грантам. Всем депонированным экспериментальным данным присваивается уникальный номер, который должен быть приведен в статье или отчете.

Данные сервисы хранения биологической информации не подходят по следующим причинам, которые можно вывести из главы о доказательстве актуальности темы:

- необходимость наличия локальной базы, позволяющей осуществлять быстрый доступ к своим данным;

- часть данных связана с секретностью, медицинской или коммерческой тайной, поэтому публичный доступ к ним должен быть запрещен;
- часть полученных данных не обладает научной ценностью (проверка идентичности клонов – штаммов, находящихся на хранении; данные промежуточных экспериментов; анализ повторных штаммов исследуемых образцов);
- недостаток необходимой информации (авторы микроорганизма, цель депонирования, патогенность, форма отчетного материала);
- отсутствие информации о качестве данных полногеномного секвенирования (количество ридов, оснований, контигов, размер полученного генома, размер наибольшего контига, покрытие, GC состав, N50);
- проблематичность внесения дополнительной информации или редактирования уже существующих записей об исследовании в публичные базы данных.

1.2. Обоснование выбора и описание методологии разрабатываемого ПО

Жизненный цикл программного обеспечения – это период времени, начинающийся с идеи создания программного продукта и завершающийся в момент его полного изъятия из эксплуатации.

Ключевыми процессами разрабатываемого ПО являются:

1. анализ – процесс разработки, включающий в себя сбор требований к проекту, их систематизацию и документирование;
2. проектирование – процесс создания проекта (проектирование схемы базы данных, объектной модели, пользовательского интерфейса и т.д.);
3. реализация – процесс создания компьютерной программы;
4. документирование – процесс написания руководства пользователя, справки;

5. тестирование – процесс испытания программного продукта.

Существует несколько методологий разработки программного обеспечения [2]. Схемы данных методологий представлены на рисунках 4-10.

Каскадная модель (Рисунок 4) предполагает последовательное прохождение стадий, при это каждая из них должна быть закончена полностью до начала следующей. Данная модель дает хороший результат только в том случае, если требования четкие и заранее определены, при этом отсутствует возможность возврата на предыдущие стадии.

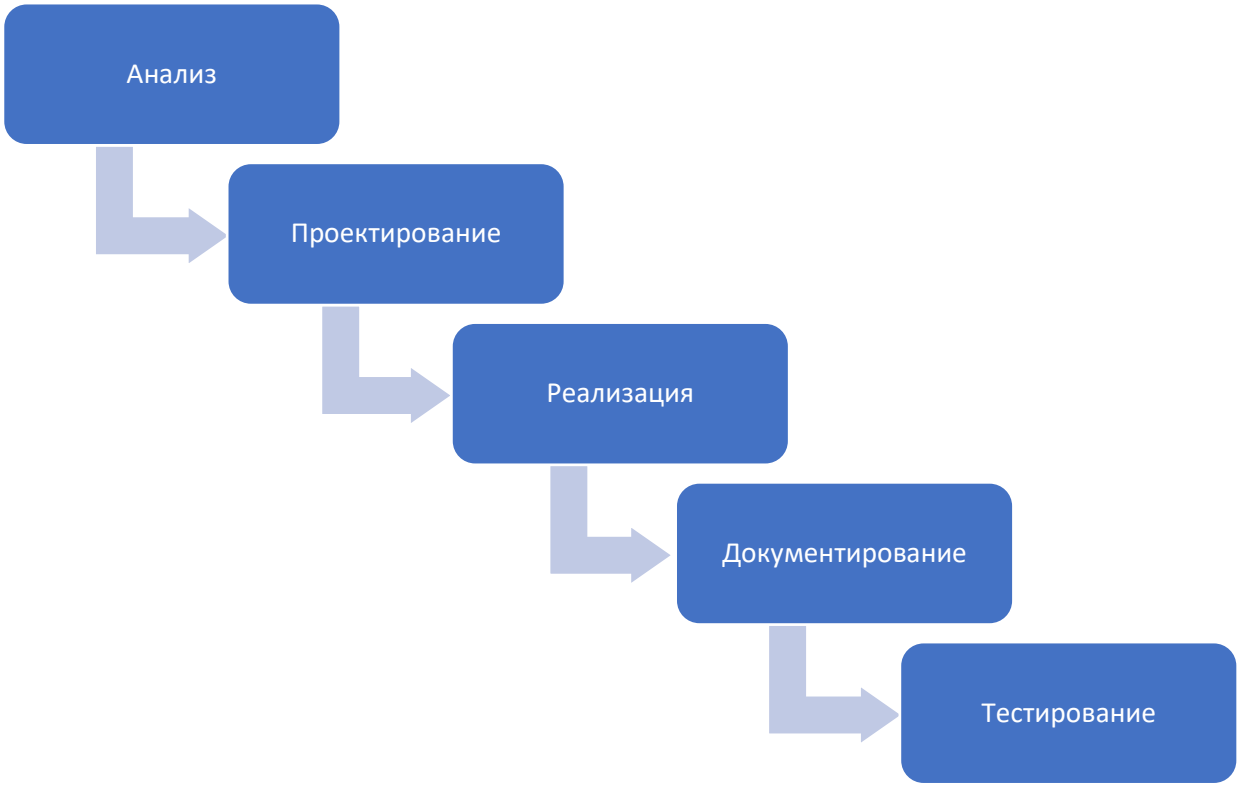


Рисунок 4 - Каскадная модель или "водопад"

V-образная модель (Рисунок 5) унаследовала структуру каскадной модели и подходит к проектам, для которых важно бесперебойное функционирование. Особенность модели в том, что она предполагает тщательную проверку и

тестирование продукта. Стадии тестирования проводятся параллельно с соответствующими стадиями разработки, например, во время кодирования пишутся модульные тесты.

Рисунок 5 - V-Model

Инкрементная модель (Рисунок 6) включает несколько циклов разработки, которые вместе составляют жизненный цикл «мульти-водопад». Цикл разделен на небольшие легко создаваемые модули. Каждый модуль проходит через все этапы. Соответственно полные требования к системе формируются для различных

модулей. Использование этой модели предполагает, что после релиза первой версии программного обеспечения следует последующее наращивание дополнительных инкрементов – новых функций.

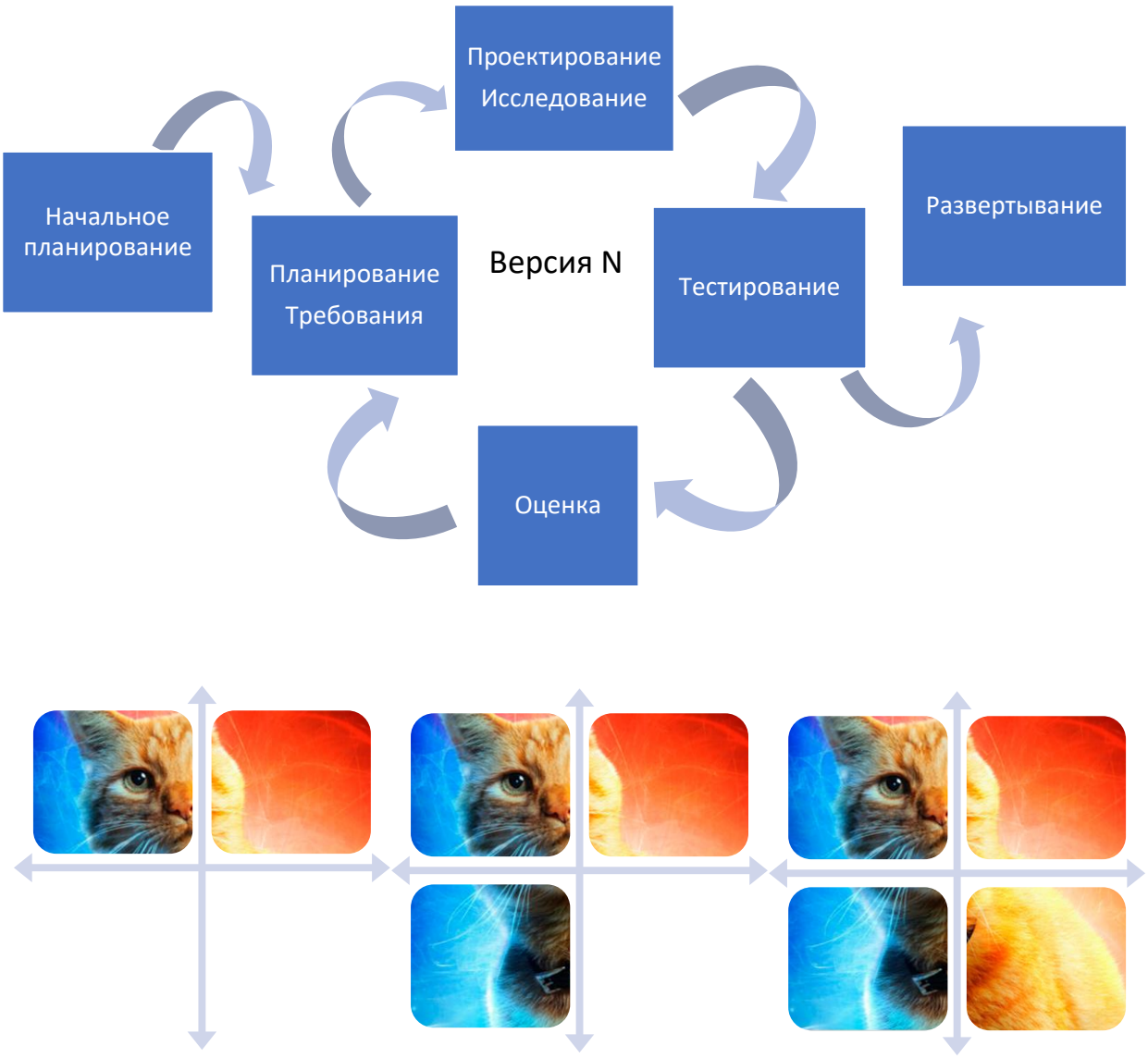


Рисунок 6 - Инкрементная модель

Модель быстрой разработки (RAD-модель) приложений работы (Рисунок 7) – разновидность инкрементной модели, направленная на быстрое получение результата в сжатые сроки и бюджет, а также нет необходимости в формировании четких требований для начала. Требования уточняются симультанно с процессом разработки. В RAD-модели модули или функции

разрабатываются несколькими командами одновременно, а затем интегрируются в один рабочий прототип.

Рисунок 7 - RAD-Model

При использовании **итерационной модели** (Рисунок 8) для начала создания проекта не требуется полная спецификация требований. Создание начинается с реализации базового функционала, после чего определяются дальнейшие требования. В результате каждой итерации расширяется функционал.

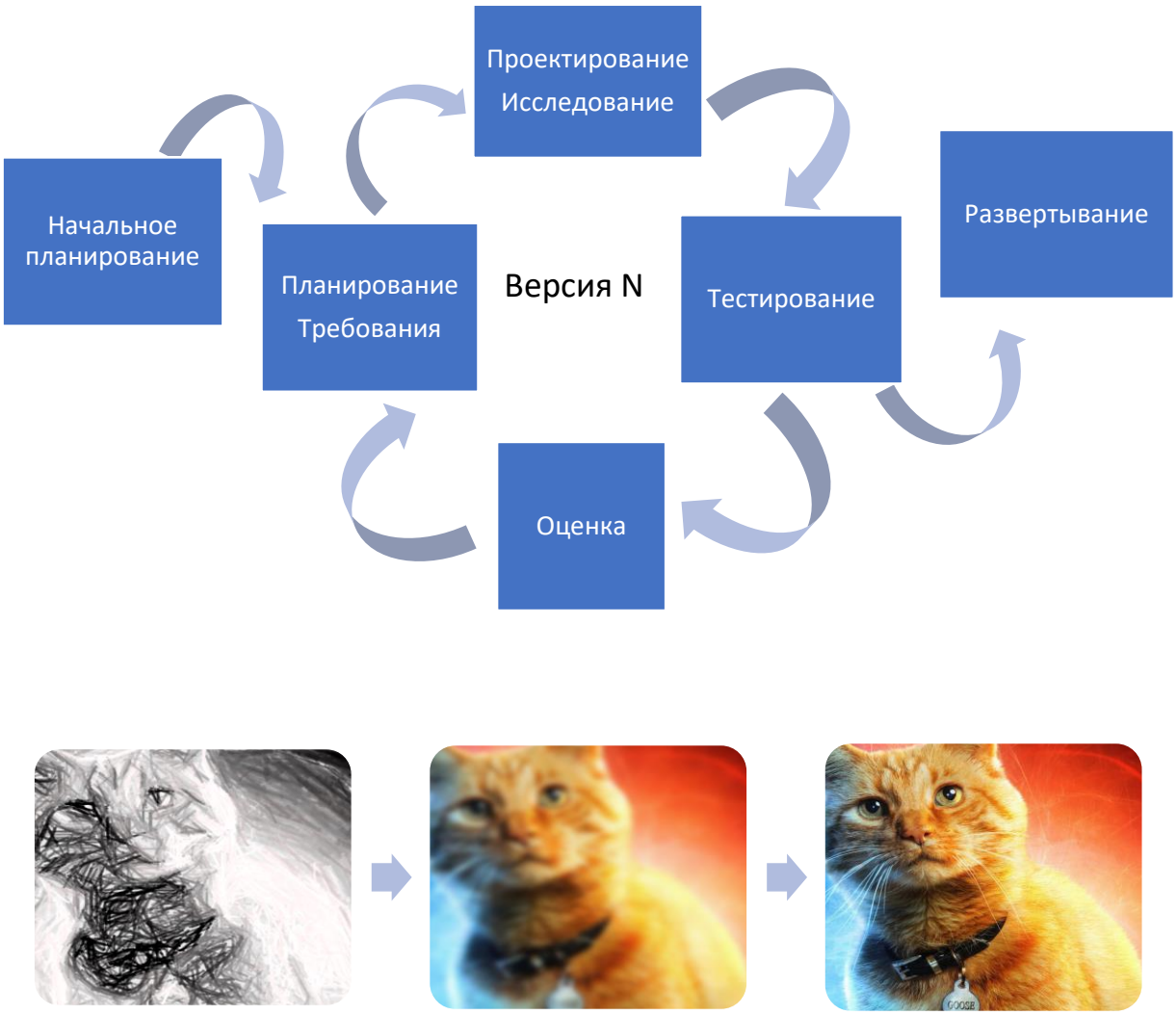


Рисунок 8 - Итерационная модель

Применяя **гибкую методологию разработки** (Рисунок 9), заказчик может отслеживать результаты после каждой итерации. В процессе реализации проекта требования могут изменяться. В основе этой методологии лежат ежедневные совещания, на которых обсуждаются продвижения в работе, текущие задачи и затруднения в разработке.

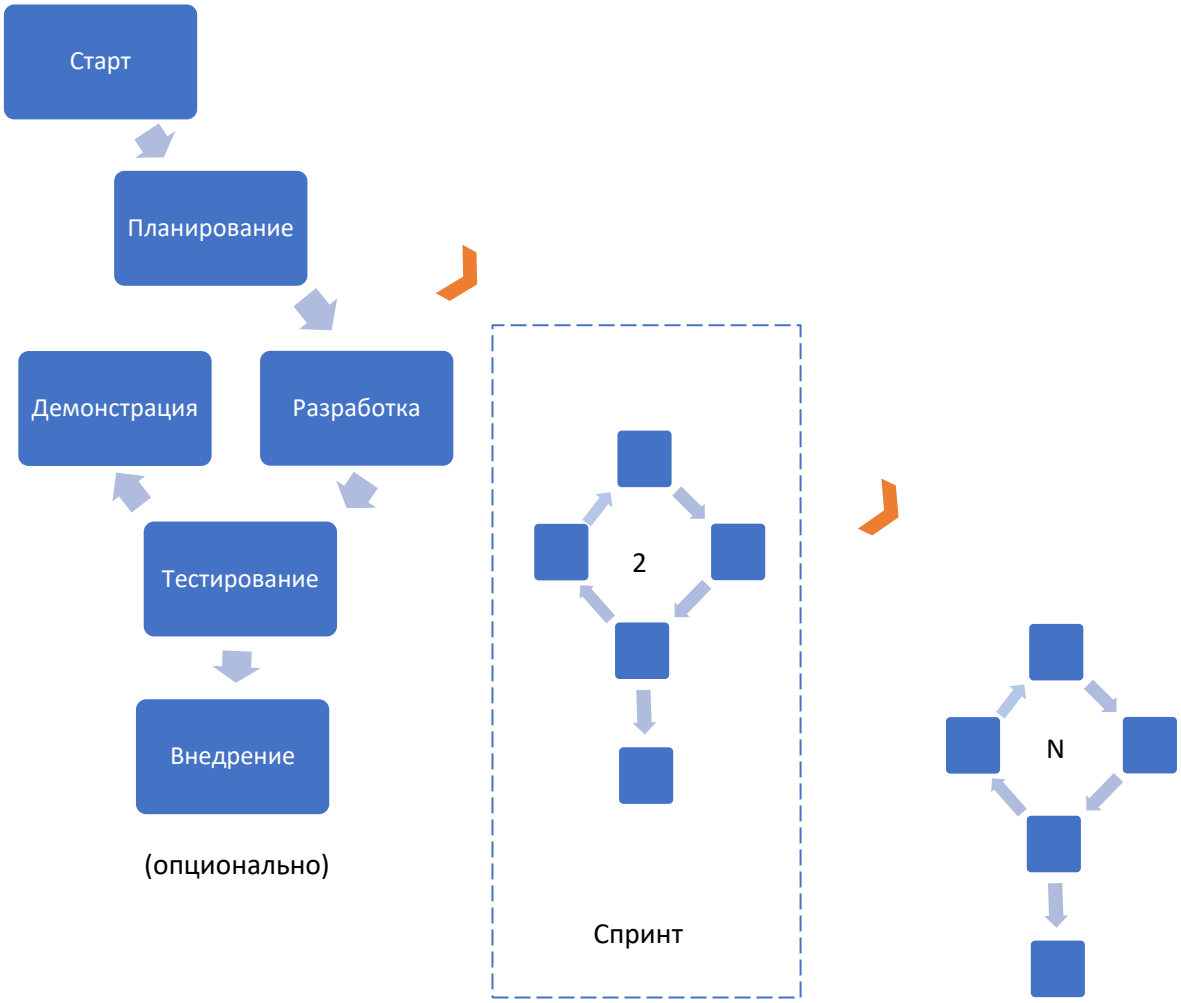


Рисунок 9 - Гибкая методология разработки

Спиральная модель (Рисунок 10) имеет сходство с инкрементной моделью. Каждый виток спирали соответствует версии программного обеспечения, на котором определяются цели, характеристики и требования для следующего витка спирали.

Рисунок 10 - Спиральная модель

Для разработки программного обеспечения была выбрана инкрементная модель разработки по следующим причинам:

- целесообразнее разделить разработку данного программного обеспечения на несколько циклов;
- требования не были сформированы до конца;
- желание Заказчика последующего наращивания функционала.

1.3. Обоснование выбора и описание инструментальных средств разработки ПО

Разработка информационной системы осуществляется с помощью использования следующих программных продуктов:

- Microsoft SQL Server 2019;
- Microsoft Visual Studio 2019.

Для удобного доступа к данным используется технология ADO.NET.

Microsoft SQL Server 2019

Для хранения данных «ГКПМ-Оболенск» было решено использовать реляционную базу данных. В качестве СУБД выступает Microsoft SQL Server 2019, которая в своей основе использует язык запросов Transact-SQL. Данная СУБД имеет наименьшее количество уязвимостей и отличается высокой производительностью. Использование данного программного обеспечения обусловлено наличием множества интегрированных служб, которые помогают расширить возможности применения разнообразной информации, например, осуществить поиск, выполнять синхронизацию и анализ, составлять запросы.

Microsoft SQL Server – хорошо масштабируемый реляционный, многопользовательский сервер баз данных, способный обрабатывать большие объемы данных для клиент-серверных приложений [3].

Имеются следующие объекты структуры БД:

- диаграммы;
- таблицы;
- виртуальные таблицы;
- хранимые процедуры;
- триггеры;
- пользователи;
- индексы;
- ключи;
- ограничения целостности;
- роли;
- правила;
- значения по умолчанию;
- пользовательские типы.
- Transact-SQL состоит из нескольких частей:

- DDL (Data Definition Language) – язык описания данных, служащий для создания базы данных и таблиц;
- DML (Data Manipulation Language) – язык манипулирования данными, служащий для добавления, изменения и удаления данных;
- DQL (Data Query Language) – язык запросов, служащий для вывода результата запроса.

Microsoft Visual Studio 2019

Для разработки программного обеспечения была выбрана интегрированная среда разработки Microsoft Visual Studio 2019. Использование Microsoft Visual Studio вызвано удобством при кодинге благодаря своим возможностям: выделение ошибок, рефакторинг, IntelliSense (показывает информацию о коде, автоматически создает отрывки кода), возможностью использования большого количества дополнительных библиотек, упрощающих процесс написания, которыми можно воспользоваться после их установки с помощью системы управления пакетами NuGet [10].

Microsoft Visual Studio поддерживает разработку на следующих языках программирования:

- C#;
- C++;
- Visual Basic;
- Python;
- JavaScript;
- F#.

Microsoft Visual Studio поддерживает разработку следующих видов приложений:

- приложения Windows;
- веб-приложения;
- приложения Office;

- приложения для интеллектуальных устройств;
- приложения для расширения среды.

Язык программирования C#

Для реализации информационной системы был выбран язык программирования C#. Данный язык подходит для разработки клиент-серверных приложений, так как он имеет возможность взаимодействовать с базами данных с помощью технологии ADO.NET Entity Framework.

C# – объектно- и компонентно-ориентированный язык программирования, относящийся к семейству языков C. C# предназначен для решения различных задач [4, 5].

Основными функциями языка, оснащающие приложения надежностью и устойчивостью:

- сборка мусора – очистка памяти от неиспользуемых объектов;
- обработка исключений – расширенный подход обнаружения ошибок;
- лямбда-выражение – поддержка приемов функционального программирования;
- синтаксис запросов – создание шаблона для работы с данными из какого-либо источника;
- поддержка операций асинхронного программирования – предоставляет возможность избавиться от узких мест производительности и увеличить общую скорость реакции приложения;
- составление шаблонов – использование сокращенного синтаксиса для применяемых алгоритмов.

Программа состоит из классов, внутри которых членами описывается алгоритм достижения какого-либо результата. Переменные, используемые внутри класса, являются полями класса. Переменные, используемые внутри метода класса, являются локальными переменными. Методом реализуется вычисление или действие, выполняемое классом.

У каждого члена класса есть свой уровень доступности, с помощью которого можно понять, откуда можно обращаться к нему. Существует шесть уровней доступности:

- public – доступ не ограничен;
- private – доступ возможен только из текущего класса;
- protected – доступ имеется у текущего класса и у классов, наследованных от него;
- internal – доступ ограничен этой сборкой;
- protected internal – доступ имеется у текущего класса и у классов, наследованных от него, либо классами той же сборке;
- private protected – доступ имеется у текущего класса, либо у классов, наследованных от данного типа в той же сборке.

ADO.NET

ADO.NET - технология, предоставляющая доступ и управление данными, основанная на платформе .NET Framework. Эта технология предоставляет набор классов, через которые устанавливается подключение к базе данных, отправляются запросы и ряд других операций.

За взаимодействия с базами данных в ADO.NET отвечают следующие объекты:

- Connection – объект для установки подключения к источнику данных;
- Command – объект для выполнения команд SQL с данными из источника данных;
- DataReader – объект, считывающий однопроходный поток данных из источника данных;
- DataSet – объект, хранящий данные из базы данных;
- DataAdapter – посредник между DataSet и источником данных.

Объект Command имеет несколько различных команд:

- ExecuteReader – возвращает данные в виде строк;

- ExecuteNonQuery – выполняет команды, которые изменяют данные в базе данных;
- ExecuteScalar – возвращает только одно значение;
- ExecuteXMLReader – получает данные из базы данных SQL Server 2000 с помощью XML-потока.

Вывод данных по главе

В данной главе был произведен обзор существующих коллекций биологической информации: NCBI, EMBL и DDJB. Причины создания своей информационной системы:

- необходимость наличия локальной базы;
- данные, связанные с секретностью, медицинской или коммерческой тайной;
- данные, не обладающие научной ценностью;
- потребность в учете дополнительной информации (цель депонирования, патогенность, форма отчетного материала);
- отсутствие метрических показателей.

Был сделан выбор методологии разработки, СУБД, среды разработки и языка программирования.

2. Проектная часть

2.1. Функциональная диаграмма «Разработка информационной системы каталога бактерий»

IDEF0 – методология графического моделирования, которая описывает структуру и функции системы, а также потоки информации и материальные объекты, связывающие эти функции [6].

Основные элементы диаграммы – блоки (процессы, функции, операции, действия) и стрелки (информационные и материальный ресурсы).

Выделяют следующие типы стрелок:

- «Вход» - материальный объект или информация, которые преобразуются функцией для получения результата;
- «Выход» - материальный объект или информация, которая является результатом выполнения функции;
- «Механизм» - ресурсы (средства), задействованные при выполнении функции;
- «Управление» - управляющие, регламентирующие и нормативные данные, которыми руководствуется функция.

Расположение стрелок относительно функции приведено на рисунке 11.

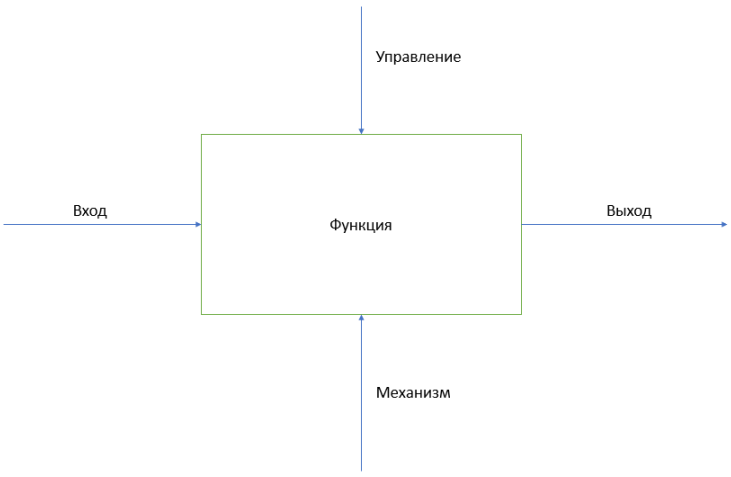


Рисунок 11 - Шаблон построения функциональной диаграммы

Основной блок – «Разработка информационной системы». На рисунке 12 представлена функциональная диаграмма: пользователь вносит входные данные, результатами чего являются заполненные справочники и паспорт штамма.

Рассмотрим элементы управления подробнее:

- СП 1.2.036-95 ПОРЯДОК УЧЕТА, ХРАНЕНИЯ, ПЕРЕДАЧИ И ТРАНСПОРТИРОВАНИЯ МИКРООРГАНИЗМОВ I – IV ГРУПП ПАТОГЕННОСТИ;
- Постановление Правительства РФ «О мерах по сохранению и рациональному использованию коллекций микроорганизмов» от 24.06.1996 г. №725-47.

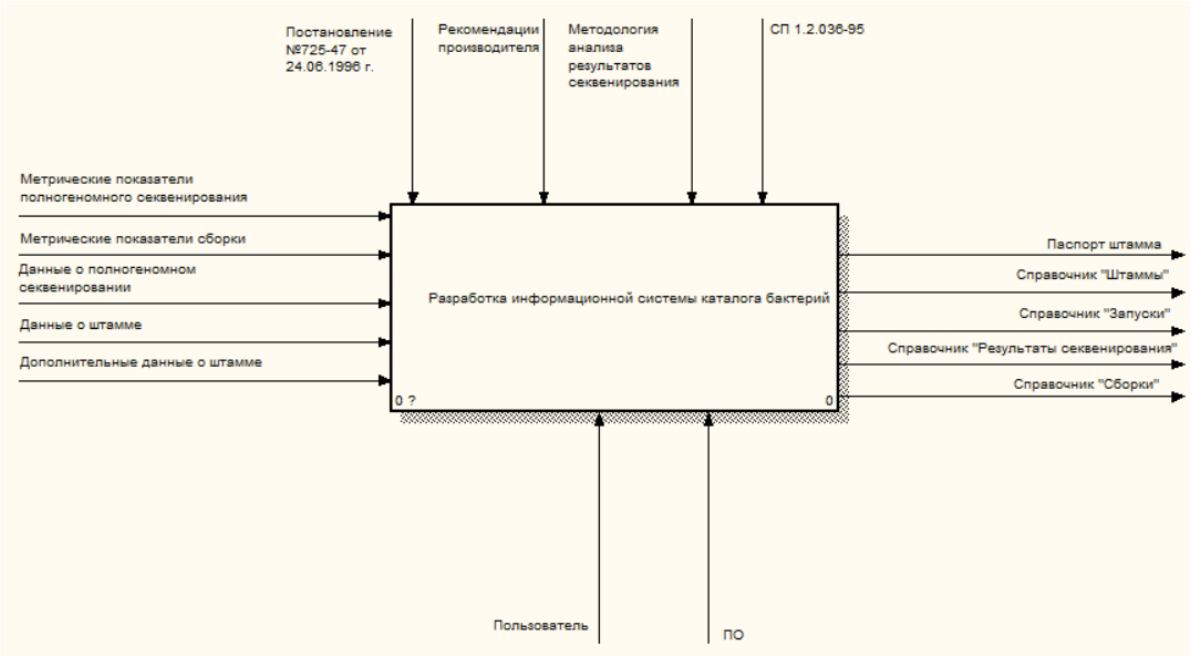


Рисунок 12 - Функциональная диаграмма "Разработка информационной системы"

На основе детализации функциональной диаграммы можно увидеть две основные функции – депонирование штаммов и учет запусков секвенирования. Депонирование происходит на основе постановления и санитарного правила, внося данные о штамме, в результате чего мы имеем паспорт штамма и заполненный справочник «Штаммы». На основе этих штаммов проводится секвенирование, которое также учитывается в системе согласно рекомендациям производителя и методологии анализа результатов секвенирования. Данные вносятся в систему, в

итоге имеются заполненные справочники. Детализация функциональной диаграммы представлена на рисунке 13.

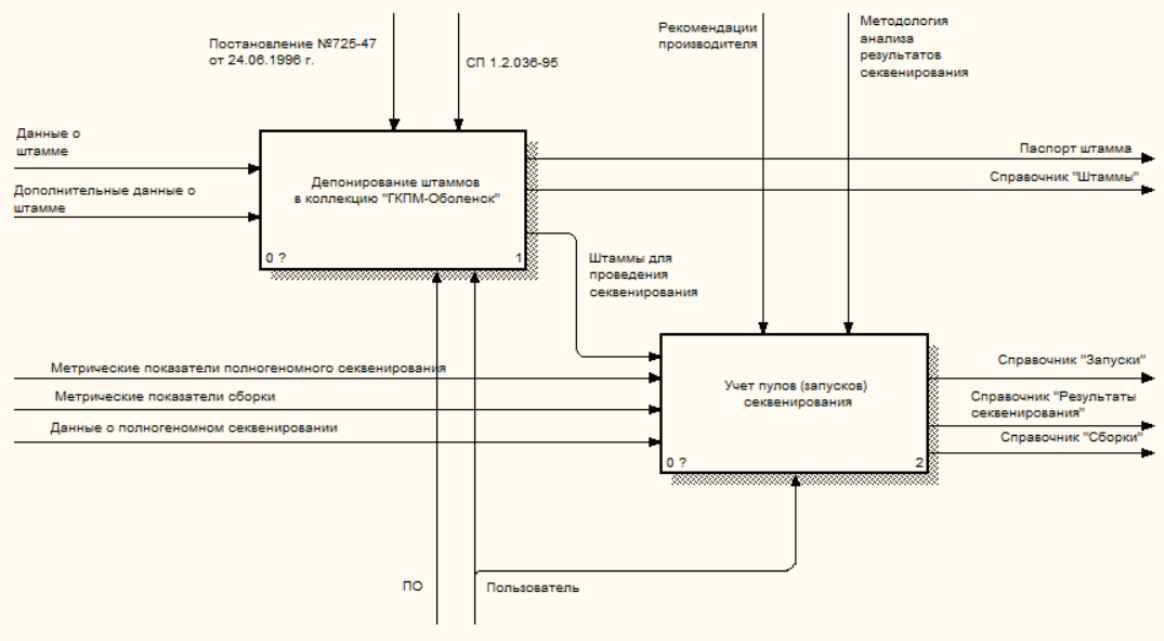


Рисунок 13 - Детализация функциональной диаграммы

При детализации процесса «Депонирование штаммов в коллекцию «ГКПМ-Оболенск»» (рис. 14) раскрываются следующие три процесса. Для добавления штамма пользователь вносит основные и дополнительные данные о штамме, в итоге программное обеспечение формирует паспорт штамма.

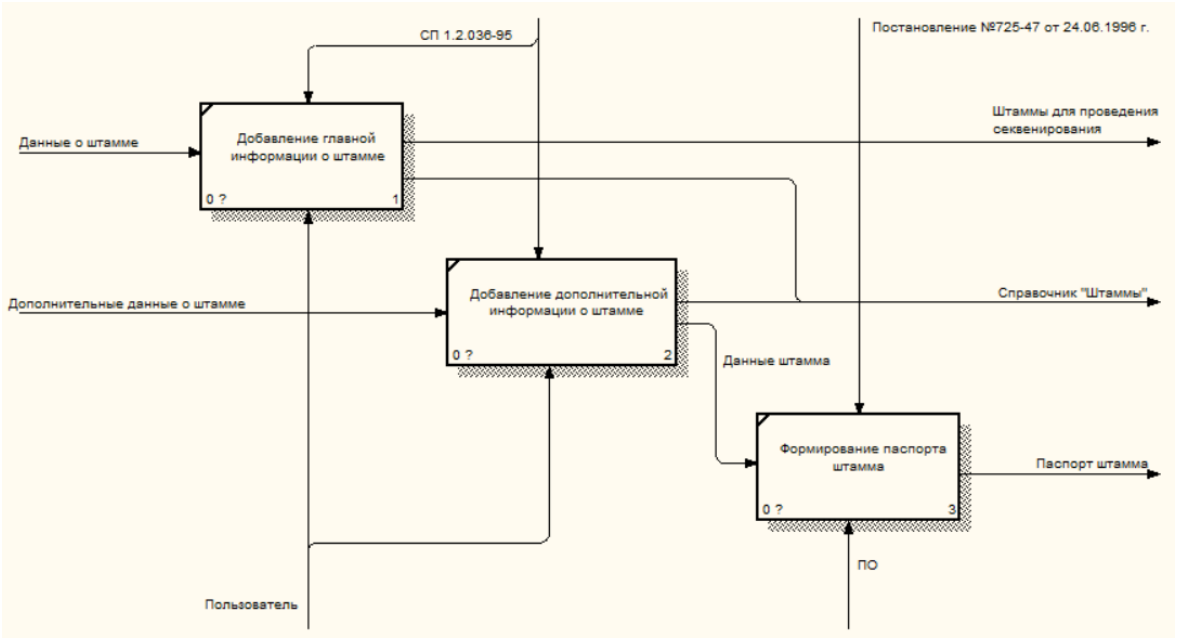


Рисунок 14 - Детализация процесса «Депонирование штаммов в коллекцию «ГКПМ-Оболенск»»

При детализации процесса «Учет пулов (запусков) секвенирования» (рис. 15) раскрываются следующие три процесса. Пользователь вносит данные о полногеномном секвенировании, затем пользователь вносит штаммы, которые были взяты для полногеномного секвенирования и их метрические показатели, после чего вносятся метрические показатели сборок. На выходе имеются заполненные справочики.

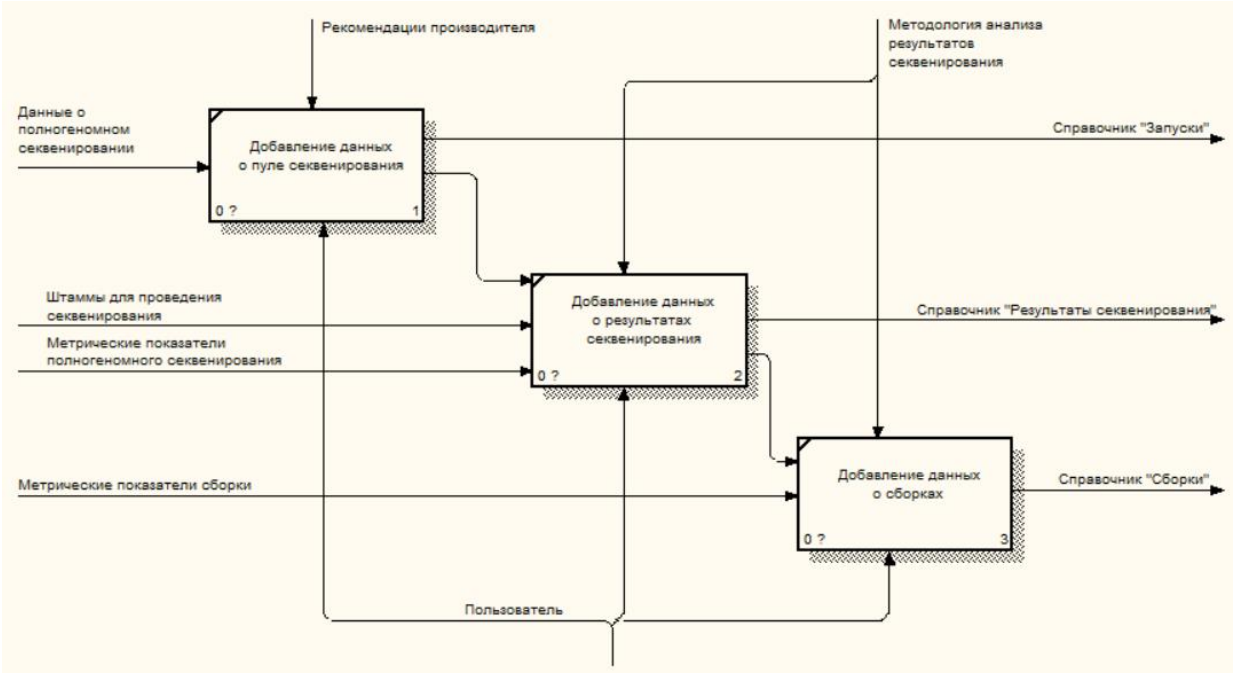


Рисунок 15 - Детализация процесса «Учет пулов (запусков) секвенирования»

2.2. Диаграмма потоков данных каталога бактерий

DFD (data flow diagrams) – это нотация, предназначенная для моделирования информационный систем с точки зрения хранения, обработки и передачи данных [7].

Для построения диаграммы потоков данных:

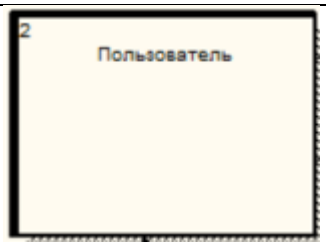
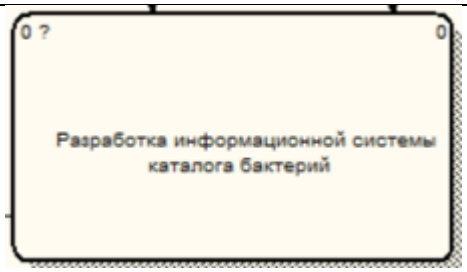
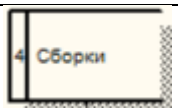
- Внешние сущности – объекты, не входящие в систему, но являющиеся источником информации и получателем информации из системы;
- Процесс – функция или последовательность действий, которые предпринимаются для преобразования входных данных в выходные данные;

- Хранилище данных – внутреннее хранилище данных, для хранения и последующего использования данных;
- Поток данных – маршруты, с помощью которых информация перемещается между внешними сущностями, процессами и хранилищами данных.

Представление элементов в таблице 2.

Таблица 2

Таблица «Представление элементов диаграммы потоков данных»

Нотация	Представление элементов
Внешняя сущность	
Процесс	
Хранилище данных	
Поток данных	

При построении диаграммы потоков данных (рис. 16) была определена одна внешняя сущность – пользователь, один процесс – разработка информационной системы и три потока данных.

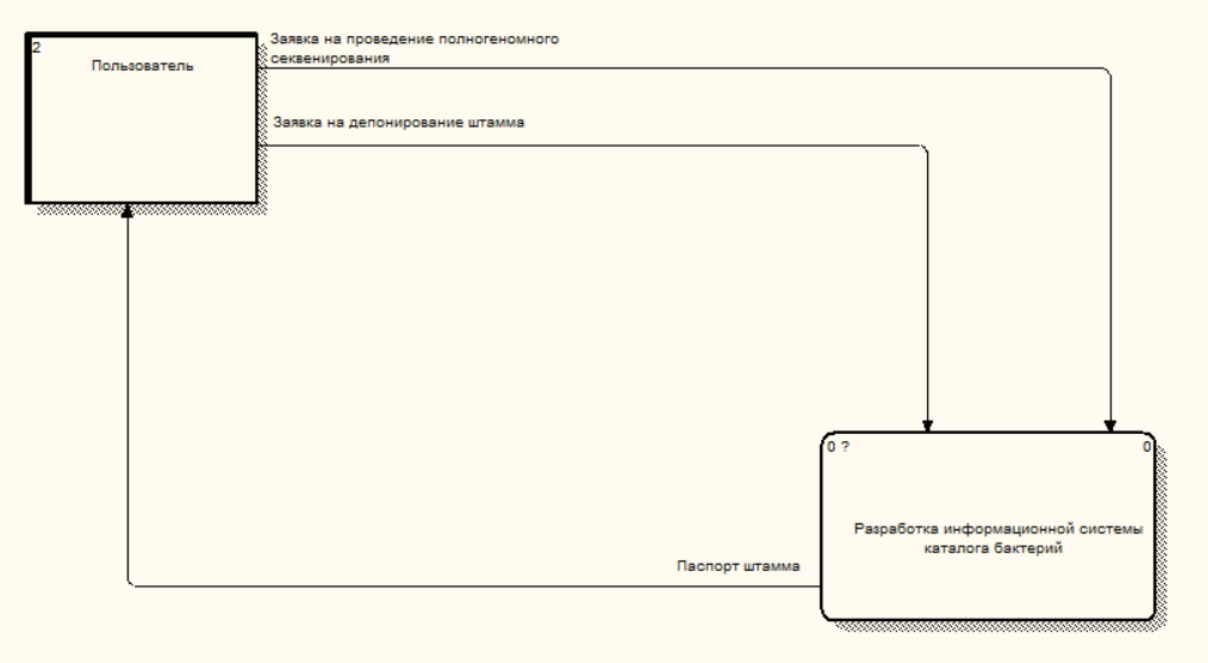


Рисунок 16 - Диаграмма потоков данных каталога бактерий

Сотруднику приходит заявка на депонирование штамма, он ее обрабатывает и вносит данные в таблицу «Штаммы». Это происходит на основании того, что на полногеномное секвенирование отправляются только штаммы, которые учтены в коллекции. Далее поступает заявка на секвенирование, она обрабатывается и определяется спецификация запуска, которая учитывается в таблице «Запуски». Далее передаются пробы для проведения полногеномного секвенирования. Когда метрические данные секвенирования получены, они заносятся в таблицу «Результаты секвенирования». Далее проходит анализ и сборка ридов, после чего данные заносятся в таблицу «Сборки». На основе имеющихся данных мы можем определить правильность определения таксономии. Если она верная, то сотрудник может получить паспорт штамма. Детализация диаграммы DFD представлена на рисунке 17.

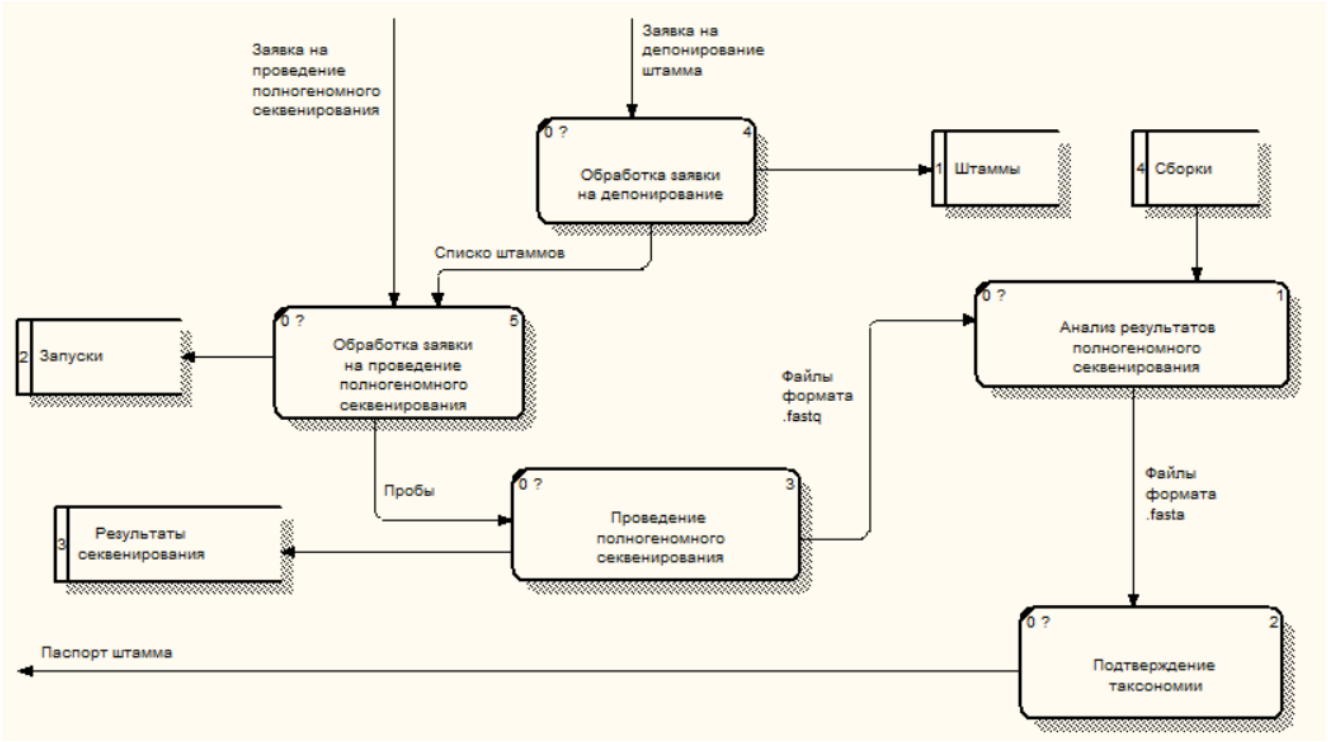


Рисунок 17 - Детализация диаграммы потоков данных

2.3. Проектирование схемы базы данных коллекции бактерий

Проектирование базы данных начинается с формирования логической модели данных, в которой выделяются основные объекты БД и определяются связи между этими объектами.

Логическая схема базы данных (рис. 18) была сформирована в ERWin. Схема включает в себя 19 сущностей.

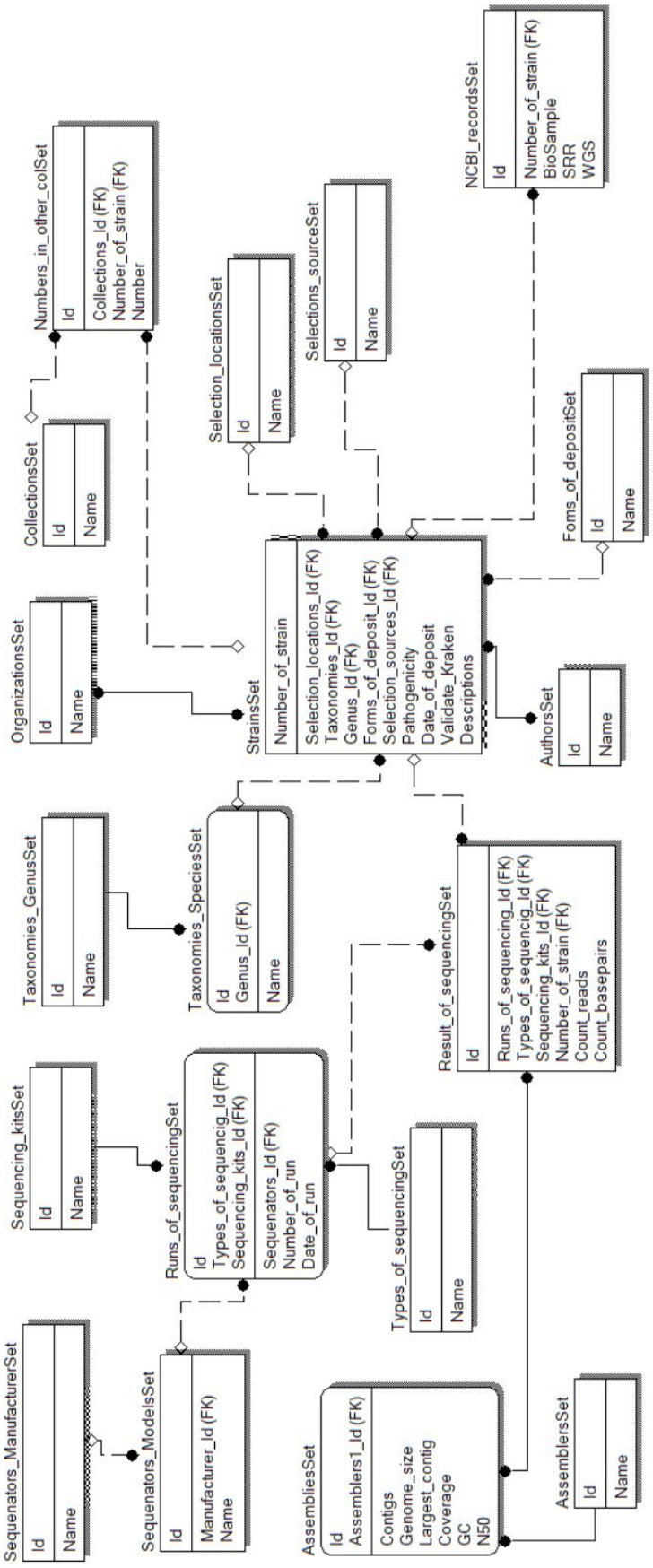


Рисунок 18 – Логическая модель данных

На основе логической модели данных разрабатывается физическая (рис. 19). Физическая модель данных – это модель данных, которая описывается средствами используемой СУБД. Разрабатываемая физическая модель данных включает в себя 22 таблиц. Код генерации БД представлен в Приложении А.

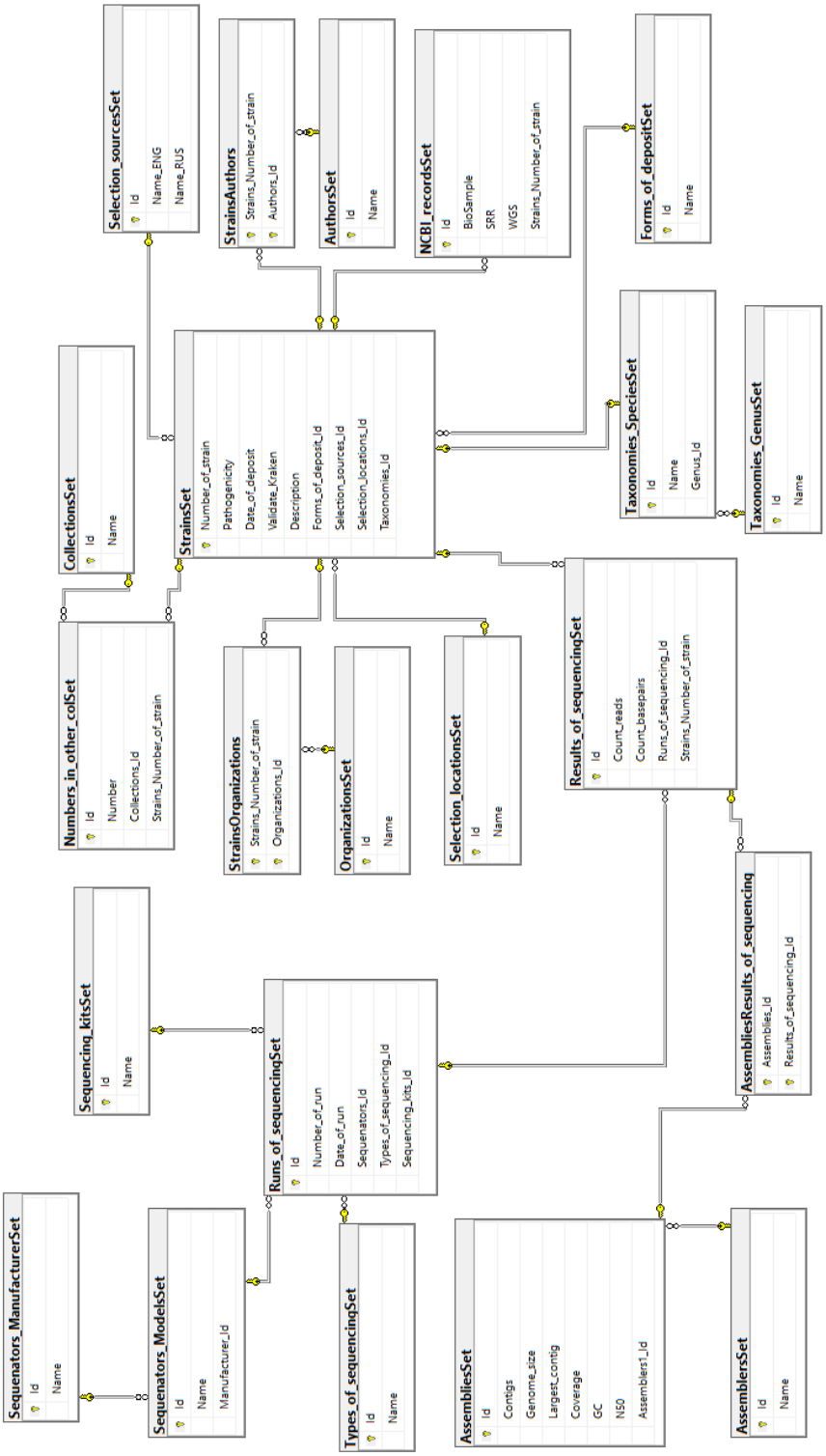


Рисунок 19 – Физическая модель данных

Основываясь на физической модели данных, опишем таблицы БД (таблицы 4 – 25).

Таблица «Assemblies» предназначена для хранения метрических показателей сборок. Метрические показатели характеризуют качество сборки.

Количество контигов – количество контигов, из которых состоит собранный геном.

Размер полученного генома – длина всех контигов собранного генома.

Размер наибольшего контига – длина максимального контига собранного генома.

Покрытие – среднее число ридов, которые ложатся на нуклеотид собранного генома.

GC-состав – доля гуанина (G) и цитизина (C) в собранном геноме.

N50 (медиана) – длина наименьшего контига сборки, который при суммировании минимального числа контигов составляет не менее 50% генома.

Таблица 3

Таблица базы данных «Assemblies»

Наименование поля	Расшифровка	Тип данных	Ключ	Обязательно к заполнению
Id	Ид	Счетчик	Первичный	Да
Contigs	Количество контигов	Числовой	-	Да
Genome_size	Размер полученного генома	Числовой	-	Да
Largest_contig	Размер наибольшего контига	Числовой	-	Да
Coverage	Покрытие	Числовой	-	Да
GC	GC состав	Числовой	-	Да
N50	N50	Числовой	-	Да
Assemblers1_Id	Внешний ключ таблицы «Assemlers»	Числовой	Внешний	Да

Таблица «Sequenators_Manufacturer» предназначена для учета производителей секвенаторов.

Таблица 4

Таблица базы данных «Sequenators_Manufacturer»

Наименование поля	Расшифровка	Тип данных	Ключ	Обязательно к заполнению
ID	Ид	Счетчик	Первичный	Да
Name	Наименование	Текстовый(50)	-	Да

Таблица «Sequenators_Models» предназначена для учета моделей секвенаторов.

Таблица 5

Таблица базы данных «Sequenators_Models»

Наименование поля	Расшифровка	Тип данных	Ключ	Обязательно к заполнению
ID	Ид	Счетчик	Первичный	Да
Name	Наименование	Текстовый(50)	-	Да

Таблица «Types_of_sequencing» предназначена для хранения типов секвенирования. Например, геном, транскриптом.

Таблица 6

Таблица базы данных «Types_of_sequencing»

Наименование поля	Расшифровка	Тип данных	Ключ	Обязательно к заполнению
Id	Ид	Счетчик	Первичный	Да
Name	Наименование	Текстовый(15)	-	Да

Таблица «Runs_of_sequencing» предназначена для хранения истории раундов секвенирования.

Таблица 7

Таблица базы данных «Runs_of_sequencing»

Наименование поля	Расшифровка	Тип данных	Ключ	Обязательно к заполнению
ID	ИД	Счетчик	Первичный	Да
Number_of_run	Номер раунда	Числовой	-	Да
Date_of_run	Дата проведения раунда	Дата	-	Да
Sequenators_Id	Секвенатор	Внешний ключ таблицы «Sequenators_Models»	Внешний	Да
Types_of_sequencing_Id	Тип секвенирования	Внешний ключ таблицы «Types_of_sequencing»	Внешний	Да
Sequencing_kits_Id	Набор для секвенирования	Внешний ключ таблицы «Sequencing_kits»	Внешний	Да

Таблица «Results_of_sequencing» предназначена для хранения метрических показателей результатов секвенирования.

Таблица 8

Таблица базы данных «Results_of_sequencing»

Наименование поля	Расшифровка	Тип данных	Ключ	Обязательно к заполнению
Id	ИД	Счетчик	Первичный	Да
Count_reads	Количество ридов	Числовой	-	Да
Count_basepairs	Количество пар оснований	Числовой	-	Да
Runs_of_sequencing_Id	Внешний ключ таблицы «Runs_of_sequencing»	Числовой	Внешний	Да
Strains_Number_of_strain	Внешний ключ таблицы «Strains»	Числовой	Внешний	Да

Таблица «Sequencing_kits» предназначена для хранения используемых наборов для секвенирования.

Таблица 9

Таблица базы данных «Sequencing_kits»

Наименование поля	Расшифровка	Тип данных	Ключ	Обязательно к заполнению
Id	Ид	Счетчик	Первичный	Да
Name	Наименование	Текстовый(50)	-	Да

Таблица «Authors» предназначена для учета авторов, которые участвуют в депонировании штамма.

Таблица 10

Таблица базы данных «Authors»

Наименование поля	Расшифровка	Тип данных	Ключ	Обязательно к заполнению
Id	ИД	Счетчик	Первичный	Да
Name	Наименование	Текстовый(150)	-	Да

Таблица «Organizations» предназначена для учета организаций, из которых были переданы штаммы на депонирование.

Таблица 11

Таблица базы данных «Organizations»

Наименование поля	Расшифровка	Тип данных	Ключ	Обязательно к заполнению
Id	ИД	Счетчик	Первичный	Да
Name	Наименование	Текстовый(150)	-	Да

Таблица «Taxonomies_Genus» предназначена для определения рода штамма.

Таблица 12

Таблица базы данных «Taxonomies_Genus»

Наименование поля	Расшифровка	Тип данных	Ключ	Обязательно к заполнению
Id	ИД	Счетчик	Первичный	Да
Name	Наименование	Текстовый(50)	-	Да

Таблица «Taxonomies_Species» предназначена для определения рода штамма.

Таблица 13

Таблица базы данных «Taxonomies_Species»

Наименование поля	Расшифровка	Тип данных	Ключ	Обязательно к заполнению
Id	ИД	Счетчик	Первичный	Да
Name	Наименование	Текстовый(50)	-	Да

Таблица «Selection_locations» предназначена для учета мест выделения штаммов.

Таблица 14

Таблица базы данных «Selection_locations»

Наименование поля	Расшифровка	Тип данных	Ключ	Обязательно к заполнению
Id	ИД	Счетчик	Первичный	Да
Name	Наименование	Текстовый(250)	-	Да

Таблица «Numbers_of_other_col» предназначена для определения номеров в других коллекциях для штамма, депонированного в «ГКПМ-Оболенск».

Таблица 15

Таблица базы данных «Numbers_of_other_col»

Наименование поля	Расшифровка	Тип данных	Ключ	Обязательно к заполнению
Id	Ид	Счетчик	Первичный	Да
Number	Название/номер штамма	Текстовый(20)		Да
Collections_Id	Внешний ключ таблицы «Collections»	Числовой	Внешний	Да
Strains_Number_of_strain	Внешний ключ таблицы «Strains»	Числовой	Внешний	Да

Таблица «Collections» предназначена для учета коллекций, в которых хранится такой же штамм.

Таблица 16

Таблица базы данных «Collections»

Наименование поля	Расшифровка	Тип данных	Ключ	Обязательно к заполнению
Id	ИД	Счетчик	Первичный	Да
Name	Наименование	Текстовый(100)	-	Да

Таблица «NCBI_records» предназначена для учета номеров доступа к штамму, депонированному в базы данных сервиса NCBI.

Таблица 17

Таблица базы данных «NCBI_records»

Наименование поля	Расшифровка	Тип данных	Ключ	Обязательно к заполнению
Id	Ид	Счетчик	Первичный	Да
BioSample	Номер доступа к образцу	Текстовый(25)	-	Да
SRR	Номер доступа к результатам секвенирования	Текстовый(25)	-	Нет
WGS	Номер доступа к геному	Текстовый(25)	-	Нет
Strains_Number_of_strain	Внешний ключ таблицы «Strains»	Числовой	Внешний	Да

Таблица «Selection_sources» предназначена для источников выделения штаммов, депонированных в «ГКПМ-Оболенск».

Таблица 18

Таблица базы данных «Selection_sources»

Наименование поля	Расшифровка	Тип данных	Ключ	Обязательно к заполнению
Id	ИД	Счетчик	Первичный	Да
Name_ENG	Наименование на английском	Текстовый(50)	-	Да
Name_RUS	Наименование на русском	Текстовый(50)	-	Да

Таблица «Forms_of_deposit» предназначена для учета форм депонирования. Например, хранение, гарантийное хранение, национальное депонирование.

Таблица 19

Таблица базы данных «Forms_of_deposit»

Наименование поля	Расшифровка	Тип данных	Ключ	Обязательно к заполнению
Id	ИД	Счетчик	Первичный	Да
Name	Наименование	Текстовый(25)	-	Да

Таблица «Strains» предназначена для учета информации о штамме, депонированном в «ГКПМ-Оболенск».

Таблица 20

Таблица базы данных «Strains»

Наименование поля	Расшифровка	Тип данных	Ключ	Обязательно к заполнению
Number_of_ strain	Инвентарный номер	Числовой	Первичный	Да
Pathogenicity	Группа патогенности	Текстовый(5)	-	Да
Date_of_ deposit	Дата депонирования	Дата	-	Да
Description	Примечание	Текстовый(MAX)	-	Нет
Forms_of_ deposit_Id	Внешний ключ таблицы «Forms_of_deposit»	Числовой	Внешний	Да
Selection_ sources_Id	Внешний ключ таблицы «Selection_sources»	Числовой	Внешний	Нет
Selection_ locations_Id	Внешний ключ таблицы «Selection_locations»	Числовой	Внешний	Нет
Taxonomies_Id	Внешний ключ таблицы «Taxonomies_Species»	Числовой	Внешний	Да

Таблица «Assemblers» предназначена для учета сборщиков, на которых были собраны штаммы.

Таблица 21

Таблица базы данных «Assemblers»

Наименование поля	Расшифровка	Тип данных	Ключ	Обязательно к заполнению
Id	Ид	Счетчик	Первичный	Да
Name	Наименование	Текстовый(20)	-	Да

Таблица «AssembliesResults_of_sequencing» является смежной таблицей для организации связи «многие-ко-многим» между таблицами «Assemblies» и «Results_of_sequencing».

Таблица 22

Таблица базы данных «AssembliesResults_of_sequencing»

Наименование поля	Расшифровка	Тип данных	Ключ	Обязательно к заполнению
Assemblies_Id	Ключ таблицы «Assemblies»	Числовой	Первичный	Да
Results_of_sequencing_Id	Ключ таблицы «Results_of_sequencing»	Числовой	Первичный	Да

Таблица «StrainsAuthors» является смежной таблицей для организации связи «многие-ко-многим» между таблицами «Strains» и «Authors».

Таблица 23

Таблица базы данных «StrainsAuthors»

Наименование поля	Расшифровка	Тип данных	Ключ	Обязательно к заполнению
Strains_Number_of_strain	Ключ таблицы «Strains»	Числовой	Первичный	Да
Authors_Id	Ключ таблицы «Authors»	Числовой	Первичный	Да

Таблица «StrainsOrganizations» является смежной таблицей для организации связи «многие-ко-многим» между таблицами «Strains» и «Organizations».

Таблица 24

Таблица базы данных «StrainsOrganizations»

Наименование поля	Расшифровка	Тип данных	Ключ	Обязательно к заполнению
Strains_Number_of_strain	Ключ таблицы «Strains»	Числовой	Первичный	Да
Organizations_Id	Ключ таблицы «Organizations»	Числовой	Первичный	Да

2.4. Интерфейс информационной системы каталога бактерий и демонстрация работы

На главной форме имеется две основные таблицы (рисунки 20-21). Интерфейс разрабатывался интуитивно понятным, чтобы любой пользователь быстро смог разобраться в управлении. Так для создания записи в таблице пользователю нужно два раза кликнуть по одной из пустой ячеек, а для редактирования – по ячейке с нужной записью.

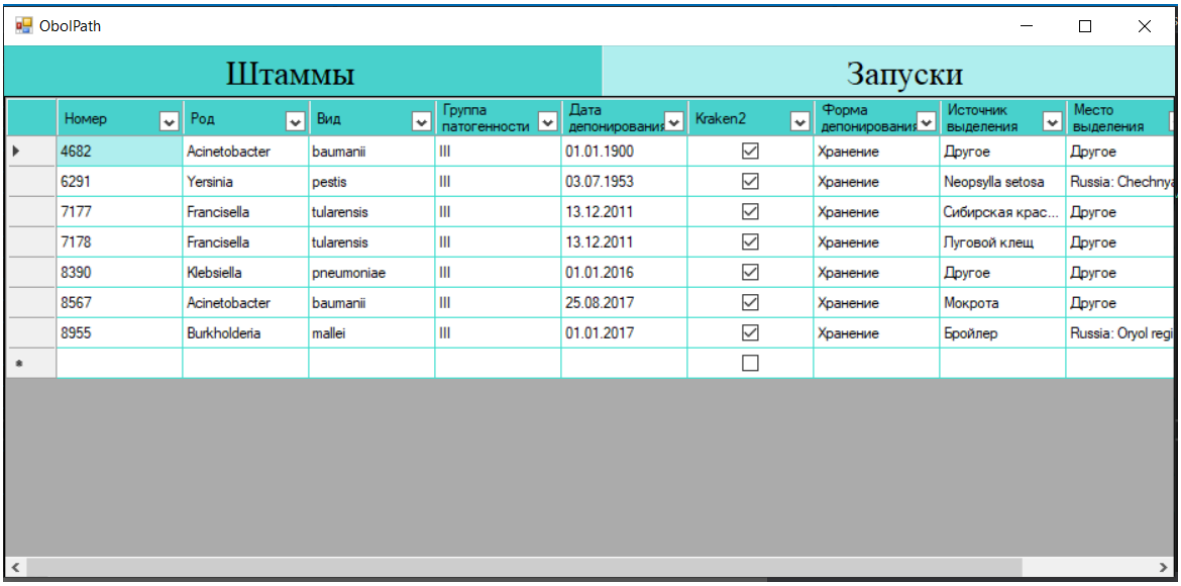


Рисунок 20 - Интерфейс главной формы. Таблица "Штаммы"

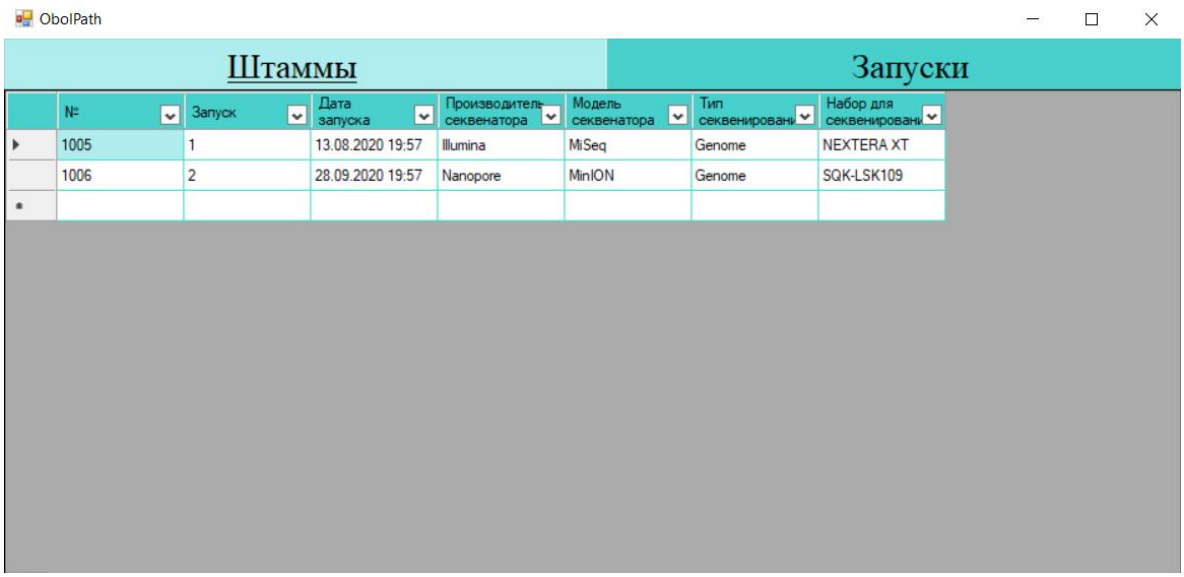


Рисунок 21 - Интерфейс главной формы. Таблица "Запуски"

Форма редактирования данных штамма имеет 4 вкладки с помощью, которых заполняются данные о штамме. Если в дополнительном справочнике нет нужного варианта, то по нажатию на Enter, она добавляется. На первой вкладке (рисунок 21) заполняется главная информация о штамме.

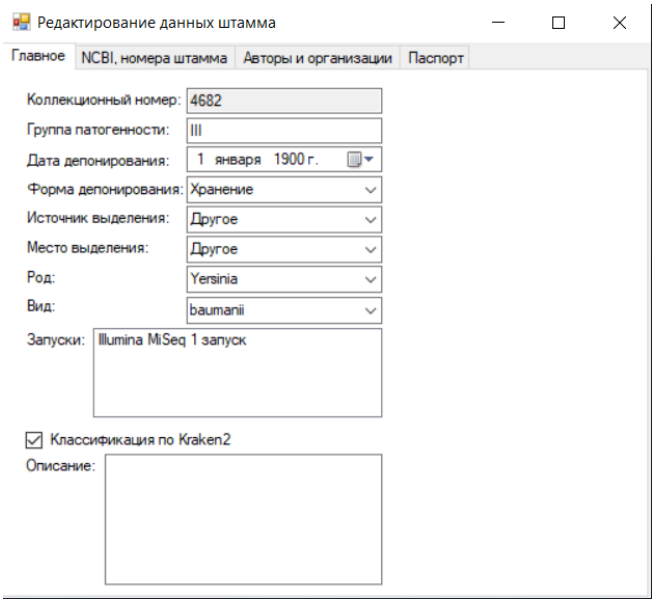


Рисунок 22 - Форма редактирования данных о штамме. Вкладка "Главное"

На вкладке «NCBI, номера штамма» (рисунок 23) заполняются две таблицы. В первой таблице заполняются номера записей депонированного штамма в сервисе NCBI. Во второй таблице заполняются данные о номерах штамма в других коллекциях.

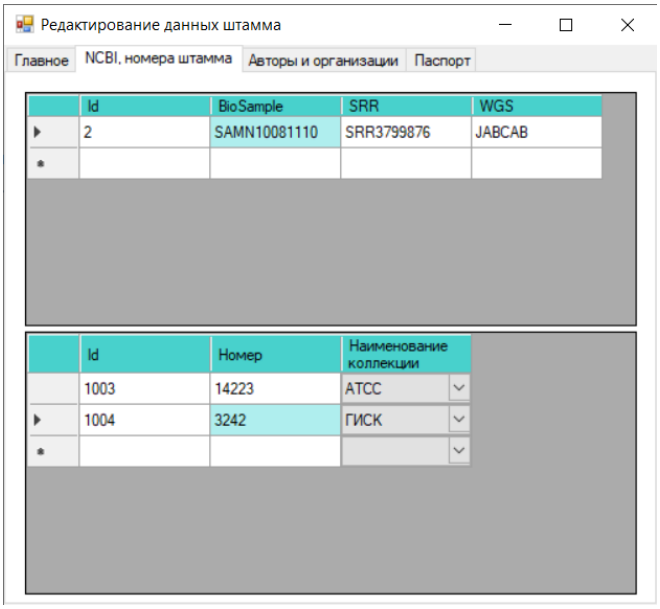


Рисунок 23 - Форма редактирования данных штамма. Вкладка "NCBI, номера штамма"

Вкладка «Авторы и организации» (рисунок 24) заполняются данные об авторах, которые были задействованы в анализе штаммов и организации, которыми был передан штамм.

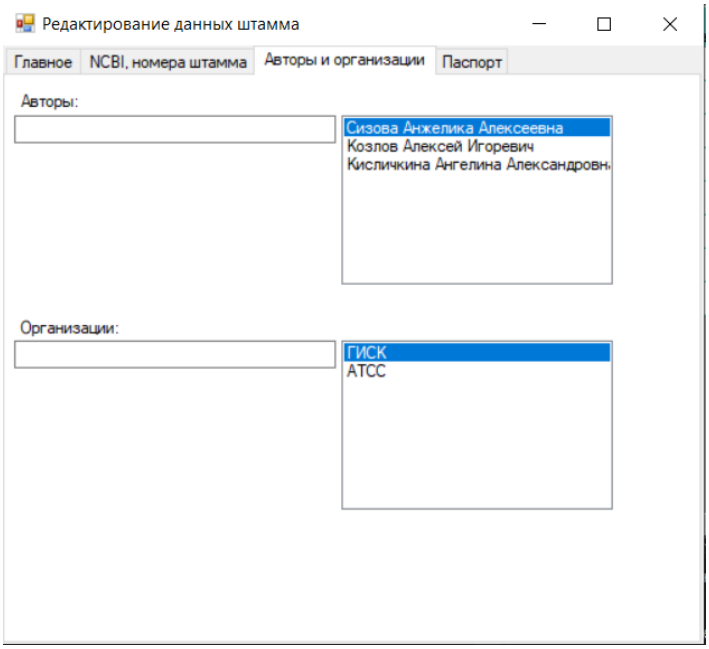


Рисунок 24 - Форма редактирования данных штамма. Вкладка "Авторы и организации"

На последней вкладке (рисунок 25) расположен паспорт штамма, который можно отсюда распечатать или сохранить в любом удобном формате.

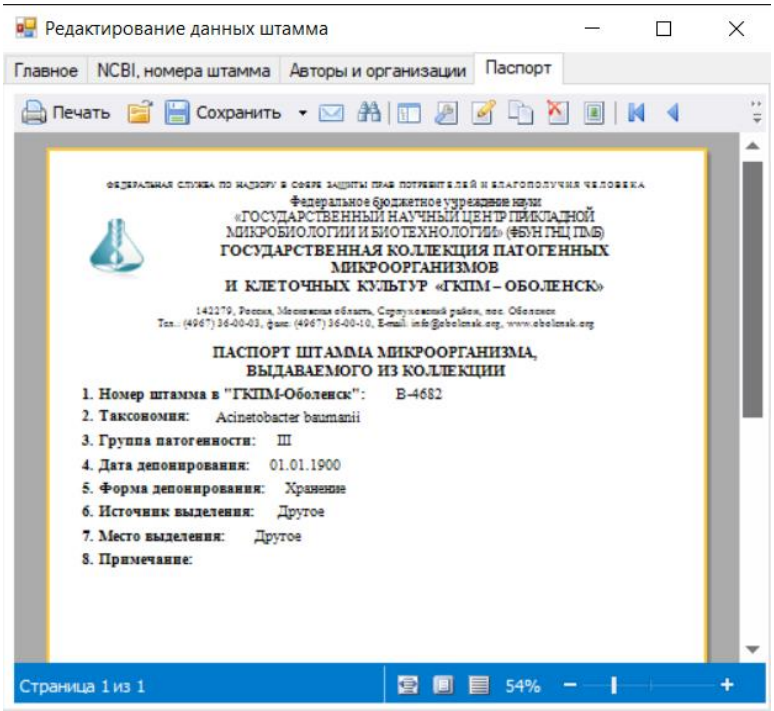


Рисунок 25 - Форма редактирования данных штамма. Вкладка "Паспорт"

Редактирование данных запуска (рисунок 26) имеет три активных зоны. В текстовых полях заполняется информация о запуске. В верхней таблице учитываются штаммы, которые были отсеквенированы. Переходя на нужную запись, снизу отображается информация о сборках для данного штамма.

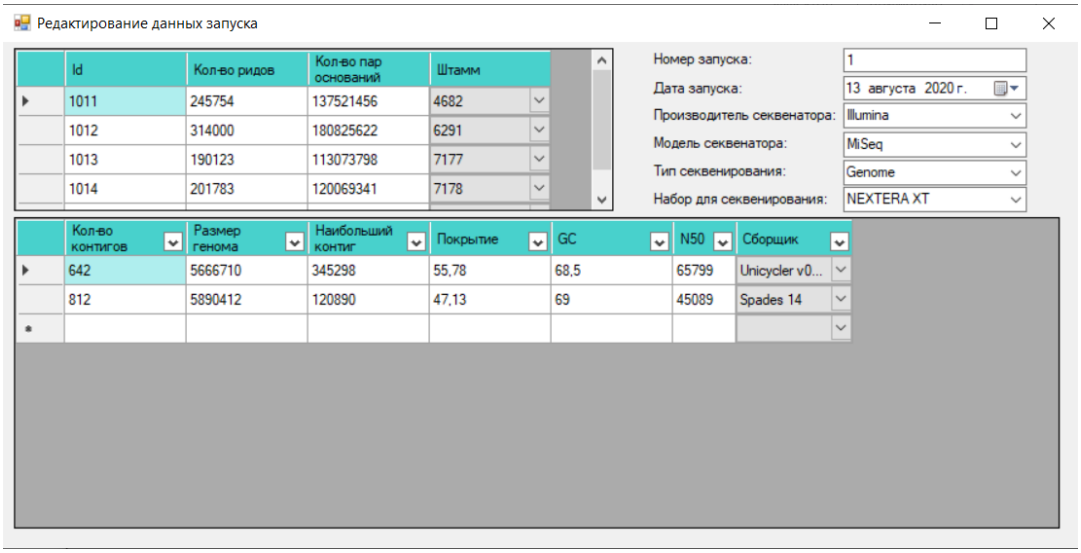


Рисунок 26 - Форма редактирования данных запуска

Вывод по главе

В этой главе были описаны бизнес-процессы с помощью IDEF0, а также была спроектирована информационная система с точки зрения хранения, обработки и передачи данных.

Была разработана база данных, содержащая в себе 22 таблицы. Система имеет две основные функции – депонирование штаммов в «ГКПМ-Оболенск» и учет пулов секвенирования.

Интерфейс программы разработан интуитивно понятным. Все функции доступны пользователю в пару кликов.

3. ЭКОНОМИЧЕСКАЯ ЧАСТЬ

В экономической части проекта производится расчет затрат на разработку программного обеспечения, которое будет обеспечивать высокую скорость обработки информации.

Цель данного раздела – расчет затрат на разработку программного обеспечения. В результате расчетов определяется себестоимость реализуемого программного обеспечения и расчет цены разработки.

Для расчета себестоимости проекта данного программного продукта, нужно произвести расчет следующих элементов:

- расчет затрат на материалы;
- расчет затрат на электроэнергию;
- расчет затрат на амортизацию оборудования;
- расчет расходов на заработную плату;
- прочие расходы.

3.1. Данные для расчета полной себестоимости

В этом подразделе приведены данные для расчета полной себестоимости на разработку программного обеспечения.

Данные для расчёта стоимости материалов представлены в таблице 25.

Таблица 25

Данные для расчёта стоимости материалов

Наименование товара	Единица измерения	Количество	Цена за единицу измерения, руб.
Диск CD-R	шт.	1,000	50,00
Бумага	упаковка	1,000	300,00

Данные для расчёта заработной платы основной представлены в таблице 26.

Таблица 26

Данные для расчёта заработной платы основной

Этапы работ	Оклад, руб.	Время работы, час.	Стоимость 1 часа работы, руб.
1. Разработка и утверждение технического задания	45 000,00	15,00	264,71
2. Разработка эскизного проекта	45 000,00	20,00	264,71
3. Разработка технического проекта	45 000,00	20,00	264,71
4. Разработка программы	45 000,00	80,00	264,71
5. Испытания программы	45 000,00	15,00	264,71
6. Разработка программной документации	45 000,00	12,00	264,71
7. Подготовка и передача программы	45 000,00	8,00	264,71

Данные для расчёта расходов на содержание оборудования и нематериальных активов представлены в таблице 27.

Таблица 27

Данные для расчёта расходов на содержание и эксплуатацию оборудования и нематериальных активов

Наименование оборудования	Стоимость, руб.	Срок службы, час	Стоимость эксплуатации оборудования за 1 час, руб.	Время работы, час
Ноутбук Lenovo Ideapad 320S-15IKB	65 000,00	6 096,00	10,66	170,00
Система разработки Visual Studio 2019 Professional (для государственных организаций)	28 320,00	6 096,00	4,65	170,00
Пакет прикладных программ Microsoft Office 2016	4 490,00	6 096,00	0,74	170,00
Среда MS SQL Server Management Studio 2019 (для государственных орагнизаций)	49 272,00	6 096,00	8,08	170,00

Стоимость 1 кВт*ч электроэнергии для предприятий составляет 3,52 рубля. Данные для расчёта расходов электроэнергии представлены в таблице 28.

Таблица 28

Данные для расчёта расхода электроэнергии

Наименование оборудования	Мощность, кВт	Время работы, час
Ноутбук Lenovo Ideapad 320S-15IKB	0,550	170,00

Транспортно-заготовительные расходы на материалы	10%
Дополнительная заработная плата	10,8%
Отчисления на социальные нужды	30%
Стоимость 1 кВт*ч электроэнергии	3,52 руб.
Накладные расходы	210%

3.1.1. Расчет затрат на материалы

Расчёт затрат на материалы представлен в таблице 29.

Таблица 29

Расчёт затрат на материалы

Наименование материала или покупного изделия	Единица измерения	Количество (m_i)	Цена за единицу измерения, руб. ($Ц_i$)	Стоимость, руб. (m_i*$Ц_i$)
Диск CD-R	шт.	1,000	50,00	50,00
Бумага	упаковка	1,000	300,00	300,00
Итого:				350,00

Стоимость материалов рассчитывается по формуле (1):

$$СМ = \sum_{i=1}^{n} m_i \times Ц_i \left(1 + \frac{\%ТЗРм}{100\%}\right) - \sum_{i=1}^{n} m_{i\ отх} \times Ц_{i\ отх}, \quad (1)$$

где m_i – норма расходов i-ого материала на единицу изделия, шт;

$Ц_i$ – цена единицы i-ого материала, руб.;

%ТЗРм – транспортно-заготовительные расходы на материалы, проценты;

$m_{i\ отх}$ – отходы i-ого материала, шт;

$Ц_{i\ отх}$ – цена единицы отходов i-ого материала, руб.

$$СМ = 350{,}00 \times \left(1 + \frac{10{,}00\%}{100\%}\right);$$

$$СМ = 350{,}00 \times 1{,}1;$$

$\text{СМ} = 385{,}00$ руб.

3.1.2. Расчет затрат на электроэнергию и амортизацию оборудования

Оборудование, на котором происходила разработка программы работает за счет электрической энергии сети переменного тока, напряжением 220В. Затраты на электроэнергию рассчитываются на основе потребляемой мощности устройства и тарифа на электроэнергию. В нашем случае используется ноутбук с мощностью 0,550 кВт*ч. Расчёт затрат на электроэнергию представлены в таблице 30.

Таблица 30

Расчёт затрат на электроэнергию

Наименование оборудования	Мощность, кВт	Время работы, час	Потреблённая электроэнергия, кВт*ч
Ноутбук Lenovo Ideapad 320S-15IKB	0,550	170,00	93,50
Итого:			93,50

Расходы на электроэнергию рассчитываются по формуле (2):

$$\text{Э} = \text{М} \times \text{Сэ}, \quad (2)$$

где М – суммарно потреблённая электроэнергия, кВт;

Сэ – стоимость 1 кВт*ч;

$$\text{Э} = 93{,}50 \times 3{,}52;$$

$$\text{Э} = 329{,}12 \text{ руб.}$$

Расходы на содержание оборудования рассчитываются из стоимости оборудования и времени его эксплуатации, по истечению которого, оно подлежит замене (в среднем это время не превышает трех лет). В течение года оборудование используется 254 рабочих дня при восьмичасовом рабочем дне. Следовательно срок службы составляет 6096 часов. Расчёт расхода на содержание оборудования и нематериальных активов представлен в таблице 31.

Таблица 31

Расчёт расхода на содержание и эксплуатацию оборудования и нематериальных активов

Наименование оборудования	Стоимость, руб.	Срок службы, час	Стоимость эксплуатации оборудования за 1 час, руб.	Время работы, час	Износ за время работы, руб.
Ноутбук Lenovo Ideapad 320S-15IKB	65 000,00	6096,00	10,66	170,00	1 812,20
Система разработки Visual Studio 2019 Professional (для государственных организаций)	28 320,00	6096,00	4,65	170,00	790,50
Пакет прикладных программ Microsoft Office 2016	4 490,00	6096,00	0,74	170,00	125,80
Среда MS SQL Server Management Studio 2019 (для государственных орагнизаций)	49 272,00	6096,00	8,08	170,00	1 373,60
Итого:					4 102,10

Расходы на содержание и эксплуатацию оборудования и нематериальных активов: РО = 4 102,10 руб.

3.1.3. Расчет расходов на заработную плату

Исходя из отработанного времени программиста, которое составило в сумме 170 часов, произведем расчет основной заработной платы, который представлен в таблице 32.

Таблица 32

Расчёт заработной платы основной

Этапы работ	Оклад, руб.	Время работы, час.	Стоимость 1 часа работы, руб.	Заработная плата по видам работ, руб.
1. Разработка и утверждение технического задания	45 000,00	15,00	264,71	3 970,65
2. Разработка эскизного проекта	45 000,00	20,00	264,71	5 294,2

3. Разработка технического проекта	45 000,00	20,00	264,71	5 294,2
4. Разработка программы	45 000,00	80,00	264,71	21 176,8
5. Испытания программы	45 000,00	15,00	264,71	3 970,65
6. Разработка программной документации	45 000,00	12,00	264,71	3 176,52
7. Подготовка и передача программы	45 000,00	8,00	264,71	2 117,68
Итого:				45 000,70

Заработная плата основная: ЗПо = 45 000,70 руб.

Для определения общей суммы расходов на оплату труда также необходимо учесть доплаты и надбавки. Примем удельный вид доплат и надбавок в размере 10,8% от основной заработной платы, руб. Дополнительная заработная плата программиста рассчитывается по формуле (3):

$$ЗПд = \frac{ЗПо \times \%ЗПд}{100\%}, \quad (3)$$

где ЗПд – заработная плата дополнительная, руб;

%ЗПд – процент дополнительной заработной платы, проценты;

ЗПд – это оплата отпуска и других социальных гарантий.

$$ЗПд = \frac{45\,000{,}70 \times 10{,}8\%}{100\%};$$

$ЗПд = 4\,860{,}08$ руб.

В отчисления на социальные нужды во внебюджетные фонды входят:

- 22% в пенсионный фонд России (ПФР);
- 2,9% в федеральный фонд социального страхования (ФФСС);
- 5,1% в федеральный фонд обязательного страхования (ФФОМС).

Что в сумме составляет 30% отчислений на социальные нужды.

Отчисления на социальные нужды рассчитываются по формуле (4):

$$ОС = \frac{(ЗПо + ЗПд) \times \%ОС}{100\%}, \quad (4)$$

где ОС – отчисления на социальные нужды, руб;

%ОС – процент отчислений на социальные нужды, проценты;

$$ОС = \frac{(45\,000{,}70 + 4\,860{,}08) \times 30\%}{100\%};$$

$$ОС = 14\,958{,}23 \text{ руб.}$$

Накладные расходы рассчитываются по формуле (5):

$$НР = \frac{(ЗПо+РО)\times\%НР}{100\%}; \quad (5)$$

где ЗПо – заработная плата основная программиста, руб.,

РО – расходы на содержание и эксплуатацию оборудования и нематериальных активов, руб.,

%НР – процент накладных расходов.

$$НР = \frac{(45\,000{,}70 + 4\,102{,}10) \times 210\%}{100\%};$$

$$НР = 103\,115{,}88 \text{ руб.}$$

3.1.4. Расчет полной себестоимости программы

На основании полученных расчетов затрат определяется полная себестоимость проекта. Полная себестоимость программы рассчитывается по формуле (6):

$$ПС = СМ + ЗПо + ЗПд + ОС + РО + Э + НР, \quad (6)$$

где СМ – стоимость материалов,

ЗПо – заработная плата основная программиста,

ЗПд – заработная плата дополнительная программиста,

ОС – отчисления на социальные нужды,

РО – расходы на содержание и эксплуатацию оборудования и нематериальных активов,

Э – стоимость потребляемой электроэнергии,

НР – накладные расходы.

$$ПС = 385{,}00 + 45\,000{,}70 + 4\,860{,}08 + 14\,958{,}23 + 4\,102{,}10 + 329{,}12 + 103\,115{,}88;$$

$$ПС = 172\,814{,}11 \text{ руб.}$$

Расчёт себестоимости разработки программы наглядно представлен в таблице 33 «Расчёт полной себестоимости программы».

Таблица 33

Расчёт полной себестоимости программы

Статья затрат	Сумма, руб.
1 Стоимость материалов	385,00
2 Основная заработная плата программиста	45 000,70
3 Дополнительная заработная плата программиста	4 860,08
4 Отчисления на социальные нужды	14 958,23
5 Расходы на содержание и эксплуатацию оборудования и нематериальных активов	4 102,10
6 Расходы на электроэнергию	329,12
7 Накладные расходы	103 115,88
Полная себестоимость	172 814,11

3.1.5. Расчет цены разработки программы

Цена разработки программы рассчитывается по формуле (7):

$$Ц = ПС \times (1 + \frac{\%Р_и}{100\%}), \quad (7)$$

где Ц – цена разработки программы, руб.;

ПС – полная (коммерческая) себестоимость, руб.;

$\%Р_и$ – рентабельность разработки программы, проценты ($\%Р_и$ = 10,00).

$$Ц = 172\,814{,}11 \times \left(1 + \frac{10{,}00\%}{100\%}\right);$$

$$Ц = 172\,814{,}11 \times 1{,}1;$$

$$Ц = 190\,095{,}52 \text{ руб.}$$

Вывод по главе

В этой главе были рассчитаны затраты по статьями. На разработку ушло 170 часов. Полная себестоимость программы составила 172 814,11 руб., цена программы составила 190 095,52 руб.

ЗАКЛЮЧЕНИЕ

Темой выпускной квалификационной работы является «Разработка информационной системы каталога бактерий» для ФБУН ГНЦ ПМБ. В процессе разработки были достигнуты следующие задачи:

1. Создан каталог для учета микроорганизмов, находящихся на хранении в «ГКПМ-Оболенск»;
2. Реализован учет проведения раундов секвенирования;
3. Реализован учет метрических показателей полногеномного секвенирования;
4. Реализована возможность формирования паспорта на основе хранящихся данных;
5. Реализован учет депонированной информации, полученной методом полногеномного секвенирования геномов штаммов бактерий в базах данных сервиса NCBI.

Целями данной выпускной квалификационной работы были достигнуты следующие цели:

- Разработана информационная система каталога бактерий;
- Сокращение времени, затрачиваемого на информационно-аналитическую деятельность, с увеличением ее результативности.

После окончания разработки, программное обеспечение было внедрено для работы в ФБУН ГНЦ ПМБ. Процесс работы с данными о штаммах, пулах секвенирования, а также их результатов стал проходить оперативно, благодаря снижению времени и затрат.

СПИСОК ЛИТЕРАТУРЫ

1. Статья про биологические базы данных [Электронный ресурс] - https://studme.org/307310/informatika/biologicheskie_bazy_dannyh

2. Статья про модели разработки [Электронный ресурс] - https://habr.com/ru/company/edison/blog/269789/

3. Статья по Microsoft SQL Server 2019 [Электронный ресурс] - https://softline.ru/about/blog/10-prichin-pereyti-na-microsoft-sql-server-2019

4. Документация по C# [Электронный ресурс] - https://docs.microsoft.com/ru-ru/dotnet/csharp/

5. Албахари Бен C# 7.0. Справочник. Полное описание языка / Албахари Бен, Албахари Джозеф

6. Методология функционального моделирования IDEF0 [Электронный ресурс] - https://nsu.ru/smk/files/idef.pdf

7. Цуканова О.А. Методология и инструментарий моделирования бизнес-процессов. – СПб.: Университет ИТМО, 2015. – 100 с

8. Сайт нуклеотидной базы данных EMBL [Электронный ресурс] - https://www.ebi.ac.uk/submission/

9. Сайт нуклеотидной базы данных DDBJ [Электронный ресурс] - https://www.ddbj.nig.ac.jp/data-categories-e.html

10. Статья о Visual Studio 2019 [Электронный ресурс] - https://docs.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2019

ПРИЛОЖЕНИЕ А

Код таблиц БД

```
-- --------------------------------------------------
-- Entity Designer DDL Script for SQL Server 2005, 2008, 2012 and Azure
-- --------------------------------------------------
-- Date Created: 01/28/2021 15:56:02
--                Generated                from                EDMX                file:
C:\Users\Sizova\source\repos\WindowsFormsApp1\WindowsFormsApp1\Model1.edmx
-- --------------------------------------------------

SET QUOTED_IDENTIFIER OFF;
GO
USE [CatalogForObolensk];
GO
IF SCHEMA_ID(N'dbo') IS NULL EXECUTE(N'CREATE SCHEMA [dbo]');
GO

-- --------------------------------------------------
-- Dropping existing FOREIGN KEY constraints
-- --------------------------------------------------

IF OBJECT_ID(N'[dbo].[FK_AssemblersAssemblies]', 'F') IS NOT NULL
    ALTER TABLE [dbo].[AssembliesSet] DROP CONSTRAINT [FK_AssemblersAssemblies];
GO
IF OBJECT_ID(N'[dbo].[FK_AssembliesResults_of_sequencing_Assemblies]', 'F') IS NOT NULL
    ALTER     TABLE     [dbo].[AssembliesResults_of_sequencing]     DROP     CONSTRAINT
[FK_AssembliesResults_of_sequencing_Assemblies];
GO
IF  OBJECT_ID(N'[dbo].[FK_AssembliesResults_of_sequencing_Results_of_sequencing]',  'F')
IS NOT NULL
    ALTER     TABLE     [dbo].[AssembliesResults_of_sequencing]     DROP     CONSTRAINT
[FK_AssembliesResults_of_sequencing_Results_of_sequencing];
GO
IF OBJECT_ID(N'[dbo].[FK_CollectionsNumbers_in_other_col]', 'F') IS NOT NULL
    ALTER       TABLE       [dbo].[Numbers_in_other_colSet]       DROP       CONSTRAINT
[FK_CollectionsNumbers_in_other_col];
GO
IF OBJECT_ID(N'[dbo].[FK_Forms_of_depositStrains]', 'F') IS NOT NULL
    ALTER TABLE [dbo].[StrainsSet] DROP CONSTRAINT [FK_Forms_of_depositStrains];
GO
IF OBJECT_ID(N'[dbo].[FK_Runs_Results_of_sequencingSet]', 'F') IS NOT NULL
    ALTER       TABLE       [dbo].[Results_of_sequencingSet]       DROP       CONSTRAINT
[FK_Runs_Results_of_sequencingSet];
GO
IF OBJECT_ID(N'[dbo].[FK_Selection_locationsStrains]', 'F') IS NOT NULL
    ALTER TABLE [dbo].[StrainsSet] DROP CONSTRAINT [FK_Selection_locationsStrains];
GO
IF OBJECT_ID(N'[dbo].[FK_Selection_sourcesStrains]', 'F') IS NOT NULL
    ALTER TABLE [dbo].[StrainsSet] DROP CONSTRAINT [FK_Selection_sourcesStrains];
GO
IF OBJECT_ID(N'[dbo].[FK_SequenatorsManufacturer]', 'F') IS NOT NULL
    ALTER       TABLE       [dbo].[Sequenators_ModelsSet]       DROP       CONSTRAINT
[FK_SequenatorsManufacturer];
GO
IF OBJECT_ID(N'[dbo].[FK_SequenatorsRuns_of_sequencing]', 'F') IS NOT NULL
    ALTER       TABLE       [dbo].[Runs_of_sequencingSet]       DROP       CONSTRAINT
[FK_SequenatorsRuns_of_sequencing];
GO
```

```
IF OBJECT_ID(N'[dbo].[FK_Sequencing_kitsRuns_of_sequencing]', 'F') IS NOT NULL
    ALTER TABLE [dbo].[Runs_of_sequencingSet] DROP CONSTRAINT
[FK_Sequencing_kitsRuns_of_sequencing];
GO
IF OBJECT_ID(N'[dbo].[FK_StrainsAuthors_Authors]', 'F') IS NOT NULL
    ALTER TABLE [dbo].[StrainsAuthors] DROP CONSTRAINT [FK_StrainsAuthors_Authors];
GO
IF OBJECT_ID(N'[dbo].[FK_StrainsAuthors_Strains]', 'F') IS NOT NULL
    ALTER TABLE [dbo].[StrainsAuthors] DROP CONSTRAINT [FK_StrainsAuthors_Strains];
GO
IF OBJECT_ID(N'[dbo].[FK_StrainsNCBI_records]', 'F') IS NOT NULL
    ALTER TABLE [dbo].[NCBI_recordsSet] DROP CONSTRAINT [FK_StrainsNCBI_records];
GO
IF OBJECT_ID(N'[dbo].[FK_StrainsNumbers_in_other_col]', 'F') IS NOT NULL
    ALTER TABLE [dbo].[Numbers_in_other_colSet] DROP CONSTRAINT
[FK_StrainsNumbers_in_other_col];
GO
IF OBJECT_ID(N'[dbo].[FK_StrainsOrganizations_Organizations]', 'F') IS NOT NULL
    ALTER TABLE [dbo].[StrainsOrganizations] DROP CONSTRAINT
[FK_StrainsOrganizations_Organizations];
GO
IF OBJECT_ID(N'[dbo].[FK_StrainsOrganizations_Strains]', 'F') IS NOT NULL
    ALTER TABLE [dbo].[StrainsOrganizations] DROP CONSTRAINT
[FK_StrainsOrganizations_Strains];
GO
IF OBJECT_ID(N'[dbo].[FK_StrainsResults_of_sequencing]', 'F') IS NOT NULL
    ALTER TABLE [dbo].[Results_of_sequencingSet] DROP CONSTRAINT
[FK_StrainsResults_of_sequencing];
GO
IF OBJECT_ID(N'[dbo].[FK_StrainsSet_Taxonomies]', 'F') IS NOT NULL
    ALTER TABLE [dbo].[StrainsSet] DROP CONSTRAINT [FK_StrainsSet_Taxonomies];
GO
IF OBJECT_ID(N'[dbo].[FK_TaxonomiesGenus]', 'F') IS NOT NULL
    ALTER TABLE [dbo].[Taxonomies_SpeciesSet] DROP CONSTRAINT [FK_TaxonomiesGenus];
GO
IF OBJECT_ID(N'[dbo].[FK_Types_of_sequencingRuns_of_sequencing]', 'F') IS NOT NULL
    ALTER TABLE [dbo].[Runs_of_sequencingSet] DROP CONSTRAINT
[FK_Types_of_sequencingRuns_of_sequencing];
GO

-- --------------------------------------------------
-- Dropping existing tables
-- --------------------------------------------------

IF OBJECT_ID(N'[dbo].[AssemblersSet]', 'U') IS NOT NULL
    DROP TABLE [dbo].[AssemblersSet];
GO
IF OBJECT_ID(N'[dbo].[AssembliesResults_of_sequencing]', 'U') IS NOT NULL
    DROP TABLE [dbo].[AssembliesResults_of_sequencing];
GO
IF OBJECT_ID(N'[dbo].[AssembliesSet]', 'U') IS NOT NULL
    DROP TABLE [dbo].[AssembliesSet];
GO
IF OBJECT_ID(N'[dbo].[AuthorsSet]', 'U') IS NOT NULL
    DROP TABLE [dbo].[AuthorsSet];
GO
IF OBJECT_ID(N'[dbo].[CollectionsSet]', 'U') IS NOT NULL
    DROP TABLE [dbo].[CollectionsSet];
GO
IF OBJECT_ID(N'[dbo].[Forms_of_depositSet]', 'U') IS NOT NULL
    DROP TABLE [dbo].[Forms_of_depositSet];
GO
```

```
IF OBJECT_ID(N'[dbo].[NCBI_recordsSet]', 'U') IS NOT NULL
    DROP TABLE [dbo].[NCBI_recordsSet];
GO
IF OBJECT_ID(N'[dbo].[Numbers_in_other_colSet]', 'U') IS NOT NULL
    DROP TABLE [dbo].[Numbers_in_other_colSet];
GO
IF OBJECT_ID(N'[dbo].[OrganizationsSet]', 'U') IS NOT NULL
    DROP TABLE [dbo].[OrganizationsSet];
GO
IF OBJECT_ID(N'[dbo].[Results_of_sequencingSet]', 'U') IS NOT NULL
    DROP TABLE [dbo].[Results_of_sequencingSet];
GO
IF OBJECT_ID(N'[dbo].[Runs_of_sequencingSet]', 'U') IS NOT NULL
    DROP TABLE [dbo].[Runs_of_sequencingSet];
GO
IF OBJECT_ID(N'[dbo].[Selection_locationsSet]', 'U') IS NOT NULL
    DROP TABLE [dbo].[Selection_locationsSet];
GO
IF OBJECT_ID(N'[dbo].[Selection_sourcesSet]', 'U') IS NOT NULL
    DROP TABLE [dbo].[Selection_sourcesSet];
GO
IF OBJECT_ID(N'[dbo].[Sequenators_ManufacturerSet]', 'U') IS NOT NULL
    DROP TABLE [dbo].[Sequenators_ManufacturerSet];
GO
IF OBJECT_ID(N'[dbo].[Sequenators_ModelsSet]', 'U') IS NOT NULL
    DROP TABLE [dbo].[Sequenators_ModelsSet];
GO
IF OBJECT_ID(N'[dbo].[Sequencing_kitsSet]', 'U') IS NOT NULL
    DROP TABLE [dbo].[Sequencing_kitsSet];
GO
IF OBJECT_ID(N'[dbo].[StrainsAuthors]', 'U') IS NOT NULL
    DROP TABLE [dbo].[StrainsAuthors];
GO
IF OBJECT_ID(N'[dbo].[StrainsOrganizations]', 'U') IS NOT NULL
    DROP TABLE [dbo].[StrainsOrganizations];
GO
IF OBJECT_ID(N'[dbo].[StrainsSet]', 'U') IS NOT NULL
    DROP TABLE [dbo].[StrainsSet];
GO
IF OBJECT_ID(N'[dbo].[sysdiagrams]', 'U') IS NOT NULL
    DROP TABLE [dbo].[sysdiagrams];
GO
IF OBJECT_ID(N'[dbo].[Taxonomies_GenusSet]', 'U') IS NOT NULL
    DROP TABLE [dbo].[Taxonomies_GenusSet];
GO
IF OBJECT_ID(N'[dbo].[Taxonomies_SpeciesSet]', 'U') IS NOT NULL
    DROP TABLE [dbo].[Taxonomies_SpeciesSet];
GO
IF OBJECT_ID(N'[dbo].[Types_of_sequencingSet]', 'U') IS NOT NULL
    DROP TABLE [dbo].[Types_of_sequencingSet];
GO

-- --------------------------------------------------
-- Creating all tables
-- --------------------------------------------------

-- Creating table 'AssemblersSet'
CREATE TABLE [dbo].[AssemblersSet] (
    [Id] int IDENTITY(1,1) NOT NULL,
    [Name] nvarchar(20)  NOT NULL
);
GO
```

```
-- Creating table 'AssembliesSet'
CREATE TABLE [dbo].[AssembliesSet] (
    [Id] int IDENTITY(1,1) NOT NULL,
    [Contigs] int  NOT NULL,
    [Genome_size] int  NOT NULL,
    [Largest_contig] int  NOT NULL,
    [Coverage] float  NOT NULL,
    [GC] float  NOT NULL,
    [N50] int  NOT NULL,
    [Assemblers1_Id] int  NOT NULL
);
GO

-- Creating table 'AuthorsSet'
CREATE TABLE [dbo].[AuthorsSet] (
    [Id] int IDENTITY(1,1) NOT NULL,
    [Name] nvarchar(150)  NOT NULL
);
GO

-- Creating table 'CollectionsSet'
CREATE TABLE [dbo].[CollectionsSet] (
    [Id] int IDENTITY(1,1) NOT NULL,
    [Name] nvarchar(100)  NOT NULL
);
GO

-- Creating table 'Forms_of_depositSet'
CREATE TABLE [dbo].[Forms_of_depositSet] (
    [Id] int IDENTITY(1,1) NOT NULL,
    [Name] nvarchar(25)  NOT NULL
);
GO

-- Creating table 'NCBI_recordsSet'
CREATE TABLE [dbo].[NCBI_recordsSet] (
    [Id] int IDENTITY(1,1) NOT NULL,
    [BioSample] nvarchar(25)  NOT NULL,
    [SRR] nvarchar(25)  NULL,
    [WGS] nvarchar(25)  NULL,
    [Strains_Number_of_strain] int  NOT NULL
);
GO

-- Creating table 'Numbers_in_other_colSet'
CREATE TABLE [dbo].[Numbers_in_other_colSet] (
    [Id] int IDENTITY(1,1) NOT NULL,
    [Number] nvarchar(20)  NOT NULL,
    [Collections_Id] int  NOT NULL,
    [Strains_Number_of_strain] int  NOT NULL
);
GO

-- Creating table 'OrganizationsSet'
CREATE TABLE [dbo].[OrganizationsSet] (
    [Id] int IDENTITY(1,1) NOT NULL,
    [Name] nvarchar(150)  NOT NULL
);
GO

-- Creating table 'Results_of_sequencingSet'
```

```
CREATE TABLE [dbo].[Results_of_sequencingSet] (
    [Id] int IDENTITY(1,1) NOT NULL,
    [Count_reads] int  NOT NULL,
    [Count_basepairs] int  NOT NULL,
    [Runs_of_sequencing_Id] int  NOT NULL,
    [Strains_Number_of_strain] int  NOT NULL
);
GO

-- Creating table 'Runs_of_sequencingSet'
CREATE TABLE [dbo].[Runs_of_sequencingSet] (
    [Id] int IDENTITY(1,1) NOT NULL,
    [Number_of_run] int  NOT NULL,
    [Date_of_run] datetime  NOT NULL,
    [Sequenators_Id] int  NOT NULL,
    [Types_of_sequencing_Id] int  NOT NULL,
    [Sequencing_kits_Id] int  NOT NULL
);
GO

-- Creating table 'Selection_locationsSet'
CREATE TABLE [dbo].[Selection_locationsSet] (
    [Id] int IDENTITY(1,1) NOT NULL,
    [Name] nvarchar(250)  NOT NULL
);
GO

-- Creating table 'Selection_sourcesSet'
CREATE TABLE [dbo].[Selection_sourcesSet] (
    [Id] int IDENTITY(1,1) NOT NULL,
    [Name_ENG] nvarchar(50)  NOT NULL,
    [Name_RUS] nvarchar(50)  NOT NULL
);
GO

-- Creating table 'Sequenators_ManufacturerSet'
CREATE TABLE [dbo].[Sequenators_ManufacturerSet] (
    [Id] int IDENTITY(1,1) NOT NULL,
    [Name] nvarchar(50)  NOT NULL
);
GO

-- Creating table 'Sequenators_ModelsSet'
CREATE TABLE [dbo].[Sequenators_ModelsSet] (
    [Id] int IDENTITY(1,1) NOT NULL,
    [Name] nvarchar(50)  NOT NULL,
    [Manufacturer_Id] int  NOT NULL
);
GO

-- Creating table 'Sequencing_kitsSet'
CREATE TABLE [dbo].[Sequencing_kitsSet] (
    [Id] int IDENTITY(1,1) NOT NULL,
    [Name] nvarchar(50)  NOT NULL
);
GO

-- Creating table 'StrainsSet'
CREATE TABLE [dbo].[StrainsSet] (
    [Number_of_strain] int  NOT NULL,
    [Pathogenicity] nvarchar(5)  NOT NULL,
    [Date_of_deposit] datetime  NOT NULL,
```

```
    [Validate_Kraken] bit  NULL,
    [Description] nvarchar(max)  NULL,
    [Forms_of_deposit_Id] int  NOT NULL,
    [Selection_sources_Id] int  NULL,
    [Selection_locations_Id] int  NULL,
    [Taxonomies_Id] int  NOT NULL
);
GO

-- Creating table 'sysdiagrams'
CREATE TABLE [dbo].[sysdiagrams] (
    [name] nvarchar(128)  NOT NULL,
    [principal_id] int  NOT NULL,
    [diagram_id] int IDENTITY(1,1) NOT NULL,
    [version] int  NULL,
    [definition] varbinary(max)  NULL
);
GO

-- Creating table 'Taxonomies_GenusSet'
CREATE TABLE [dbo].[Taxonomies_GenusSet] (
    [Id] int IDENTITY(1,1) NOT NULL,
    [Name] nvarchar(50)  NOT NULL
);
GO

-- Creating table 'Taxonomies_SpeciesSet'
CREATE TABLE [dbo].[Taxonomies_SpeciesSet] (
    [Id] int IDENTITY(1,1) NOT NULL,
    [Name] nvarchar(50)  NOT NULL,
    [Genus_Id] int  NULL
);
GO

-- Creating table 'Types_of_sequencingSet'
CREATE TABLE [dbo].[Types_of_sequencingSet] (
    [Id] int IDENTITY(1,1) NOT NULL,
    [Name] nvarchar(15)  NOT NULL
);
GO

-- Creating table 'AssembliesResults_of_sequencing'
CREATE TABLE [dbo].[AssembliesResults_of_sequencing] (
    [AssembliesSet_Id] int  NOT NULL,
    [Results_of_sequencingSet_Id] int  NOT NULL
);
GO

-- Creating table 'StrainsAuthors'
CREATE TABLE [dbo].[StrainsAuthors] (
    [AuthorsSet_Id] int  NOT NULL,
    [StrainsSet_Number_of_strain] int  NOT NULL
);
GO

-- Creating table 'StrainsOrganizations'
CREATE TABLE [dbo].[StrainsOrganizations] (
    [OrganizationsSet_Id] int  NOT NULL,
    [StrainsSet_Number_of_strain] int  NOT NULL
);
GO
```

```
-- --------------------------------------------------
-- Creating all PRIMARY KEY constraints
-- --------------------------------------------------

-- Creating primary key on [Id] in table 'AssemblersSet'
ALTER TABLE [dbo].[AssemblersSet]
ADD CONSTRAINT [PK_AssemblersSet]
    PRIMARY KEY CLUSTERED ([Id] ASC);
GO

-- Creating primary key on [Id] in table 'AssembliesSet'
ALTER TABLE [dbo].[AssembliesSet]
ADD CONSTRAINT [PK_AssembliesSet]
    PRIMARY KEY CLUSTERED ([Id] ASC);
GO

-- Creating primary key on [Id] in table 'AuthorsSet'
ALTER TABLE [dbo].[AuthorsSet]
ADD CONSTRAINT [PK_AuthorsSet]
    PRIMARY KEY CLUSTERED ([Id] ASC);
GO

-- Creating primary key on [Id] in table 'CollectionsSet'
ALTER TABLE [dbo].[CollectionsSet]
ADD CONSTRAINT [PK_CollectionsSet]
    PRIMARY KEY CLUSTERED ([Id] ASC);
GO

-- Creating primary key on [Id] in table 'Forms_of_depositSet'
ALTER TABLE [dbo].[Forms_of_depositSet]
ADD CONSTRAINT [PK_Forms_of_depositSet]
    PRIMARY KEY CLUSTERED ([Id] ASC);
GO

-- Creating primary key on [Id] in table 'NCBI_recordsSet'
ALTER TABLE [dbo].[NCBI_recordsSet]
ADD CONSTRAINT [PK_NCBI_recordsSet]
    PRIMARY KEY CLUSTERED ([Id] ASC);
GO

-- Creating primary key on [Id] in table 'Numbers_in_other_colSet'
ALTER TABLE [dbo].[Numbers_in_other_colSet]
ADD CONSTRAINT [PK_Numbers_in_other_colSet]
    PRIMARY KEY CLUSTERED ([Id] ASC);
GO

-- Creating primary key on [Id] in table 'OrganizationsSet'
ALTER TABLE [dbo].[OrganizationsSet]
ADD CONSTRAINT [PK_OrganizationsSet]
    PRIMARY KEY CLUSTERED ([Id] ASC);
GO

-- Creating primary key on [Id] in table 'Results_of_sequencingSet'
ALTER TABLE [dbo].[Results_of_sequencingSet]
ADD CONSTRAINT [PK_Results_of_sequencingSet]
    PRIMARY KEY CLUSTERED ([Id] ASC);
GO

-- Creating primary key on [Id] in table 'Runs_of_sequencingSet'
ALTER TABLE [dbo].[Runs_of_sequencingSet]
ADD CONSTRAINT [PK_Runs_of_sequencingSet]
    PRIMARY KEY CLUSTERED ([Id] ASC);
```

```
GO

-- Creating primary key on [Id] in table 'Selection_locationsSet'
ALTER TABLE [dbo].[Selection_locationsSet]
ADD CONSTRAINT [PK_Selection_locationsSet]
    PRIMARY KEY CLUSTERED ([Id] ASC);
GO

-- Creating primary key on [Id] in table 'Selection_sourcesSet'
ALTER TABLE [dbo].[Selection_sourcesSet]
ADD CONSTRAINT [PK_Selection_sourcesSet]
    PRIMARY KEY CLUSTERED ([Id] ASC);
GO

-- Creating primary key on [Id] in table 'Sequenators_ManufacturerSet'
ALTER TABLE [dbo].[Sequenators_ManufacturerSet]
ADD CONSTRAINT [PK_Sequenators_ManufacturerSet]
    PRIMARY KEY CLUSTERED ([Id] ASC);
GO

-- Creating primary key on [Id] in table 'Sequenators_ModelsSet'
ALTER TABLE [dbo].[Sequenators_ModelsSet]
ADD CONSTRAINT [PK_Sequenators_ModelsSet]
    PRIMARY KEY CLUSTERED ([Id] ASC);
GO

-- Creating primary key on [Id] in table 'Sequencing_kitsSet'
ALTER TABLE [dbo].[Sequencing_kitsSet]
ADD CONSTRAINT [PK_Sequencing_kitsSet]
    PRIMARY KEY CLUSTERED ([Id] ASC);
GO

-- Creating primary key on [Number_of_strain] in table 'StrainsSet'
ALTER TABLE [dbo].[StrainsSet]
ADD CONSTRAINT [PK_StrainsSet]
    PRIMARY KEY CLUSTERED ([Number_of_strain] ASC);
GO

-- Creating primary key on [diagram_id] in table 'sysdiagrams'
ALTER TABLE [dbo].[sysdiagrams]
ADD CONSTRAINT [PK_sysdiagrams]
    PRIMARY KEY CLUSTERED ([diagram_id] ASC);
GO

-- Creating primary key on [Id] in table 'Taxonomies_GenusSet'
ALTER TABLE [dbo].[Taxonomies_GenusSet]
ADD CONSTRAINT [PK_Taxonomies_GenusSet]
    PRIMARY KEY CLUSTERED ([Id] ASC);
GO

-- Creating primary key on [Id] in table 'Taxonomies_SpeciesSet'
ALTER TABLE [dbo].[Taxonomies_SpeciesSet]
ADD CONSTRAINT [PK_Taxonomies_SpeciesSet]
    PRIMARY KEY CLUSTERED ([Id] ASC);
GO

-- Creating primary key on [Id] in table 'Types_of_sequencingSet'
ALTER TABLE [dbo].[Types_of_sequencingSet]
ADD CONSTRAINT [PK_Types_of_sequencingSet]
    PRIMARY KEY CLUSTERED ([Id] ASC);
GO
```

```
-- Creating primary key on [AssembliesSet_Id], [Results_of_sequencingSet_Id] in table
'AssembliesResults_of_sequencing'
ALTER TABLE [dbo].[AssembliesResults_of_sequencing]
ADD CONSTRAINT [PK_AssembliesResults_of_sequencing]
    PRIMARY KEY CLUSTERED ([AssembliesSet_Id], [Results_of_sequencingSet_Id] ASC);
GO

-- Creating primary key on [AuthorsSet_Id], [StrainsSet_Number_of_strain] in table
'StrainsAuthors'
ALTER TABLE [dbo].[StrainsAuthors]
ADD CONSTRAINT [PK_StrainsAuthors]
    PRIMARY KEY CLUSTERED ([AuthorsSet_Id], [StrainsSet_Number_of_strain] ASC);
GO

-- Creating primary key on [OrganizationsSet_Id], [StrainsSet_Number_of_strain] in table
'StrainsOrganizations'
ALTER TABLE [dbo].[StrainsOrganizations]
ADD CONSTRAINT [PK_StrainsOrganizations]
    PRIMARY KEY CLUSTERED ([OrganizationsSet_Id], [StrainsSet_Number_of_strain] ASC);
GO

-- --------------------------------------------------
-- Creating all FOREIGN KEY constraints
-- --------------------------------------------------

-- Creating foreign key on [Assemblers1_Id] in table 'AssembliesSet'
ALTER TABLE [dbo].[AssembliesSet]
ADD CONSTRAINT [FK_AssemblersAssemblies]
    FOREIGN KEY ([Assemblers1_Id])
    REFERENCES [dbo].[AssemblersSet]
        ([Id])
    ON DELETE NO ACTION ON UPDATE NO ACTION;
GO

-- Creating non-clustered index for FOREIGN KEY 'FK_AssemblersAssemblies'
CREATE INDEX [IX_FK_AssemblersAssemblies]
ON [dbo].[AssembliesSet]
    ([Assemblers1_Id]);
GO

-- Creating foreign key on [Collections_Id] in table 'Numbers_in_other_colSet'
ALTER TABLE [dbo].[Numbers_in_other_colSet]
ADD CONSTRAINT [FK_CollectionsNumbers_in_other_col]
    FOREIGN KEY ([Collections_Id])
    REFERENCES [dbo].[CollectionsSet]
        ([Id])
    ON DELETE NO ACTION ON UPDATE NO ACTION;
GO

-- Creating non-clustered index for FOREIGN KEY 'FK_CollectionsNumbers_in_other_col'
CREATE INDEX [IX_FK_CollectionsNumbers_in_other_col]
ON [dbo].[Numbers_in_other_colSet]
    ([Collections_Id]);
GO

-- Creating foreign key on [Forms_of_deposit_Id] in table 'StrainsSet'
ALTER TABLE [dbo].[StrainsSet]
ADD CONSTRAINT [FK_Forms_of_depositStrains]
    FOREIGN KEY ([Forms_of_deposit_Id])
    REFERENCES [dbo].[Forms_of_depositSet]
        ([Id])
    ON DELETE NO ACTION ON UPDATE NO ACTION;
```

```
GO

-- Creating non-clustered index for FOREIGN KEY 'FK_Forms_of_depositStrains'
CREATE INDEX [IX_FK_Forms_of_depositStrains]
ON [dbo].[StrainsSet]
    ([Forms_of_deposit_Id]);
GO

-- Creating foreign key on [Strains_Number_of_strain] in table 'NCBI_recordsSet'
ALTER TABLE [dbo].[NCBI_recordsSet]
ADD CONSTRAINT [FK_StrainsNCBI_records]
    FOREIGN KEY ([Strains_Number_of_strain])
    REFERENCES [dbo].[StrainsSet]
        ([Number_of_strain])
    ON DELETE NO ACTION ON UPDATE NO ACTION;
GO

-- Creating non-clustered index for FOREIGN KEY 'FK_StrainsNCBI_records'
CREATE INDEX [IX_FK_StrainsNCBI_records]
ON [dbo].[NCBI_recordsSet]
    ([Strains_Number_of_strain]);
GO

-- Creating foreign key on [Strains_Number_of_strain] in table 'Numbers_in_other_colSet'
ALTER TABLE [dbo].[Numbers_in_other_colSet]
ADD CONSTRAINT [FK_StrainsNumbers_in_other_col]
    FOREIGN KEY ([Strains_Number_of_strain])
    REFERENCES [dbo].[StrainsSet]
        ([Number_of_strain])
    ON DELETE NO ACTION ON UPDATE NO ACTION;
GO

-- Creating non-clustered index for FOREIGN KEY 'FK_StrainsNumbers_in_other_col'
CREATE INDEX [IX_FK_StrainsNumbers_in_other_col]
ON [dbo].[Numbers_in_other_colSet]
    ([Strains_Number_of_strain]);
GO

-- Creating foreign key on [Runs_of_sequencing_Id] in table 'Results_of_sequencingSet'
ALTER TABLE [dbo].[Results_of_sequencingSet]
ADD CONSTRAINT [FK_Runs_Results_of_sequencingSet]
    FOREIGN KEY ([Runs_of_sequencing_Id])
    REFERENCES [dbo].[Runs_of_sequencingSet]
        ([Id])
    ON DELETE NO ACTION ON UPDATE NO ACTION;
GO

-- Creating non-clustered index for FOREIGN KEY 'FK_Runs_Results_of_sequencingSet'
CREATE INDEX [IX_FK_Runs_Results_of_sequencingSet]
ON [dbo].[Results_of_sequencingSet]
    ([Runs_of_sequencing_Id]);
GO

-- Creating foreign key on [Strains_Number_of_strain] in table 'Results_of_sequencingSet'
ALTER TABLE [dbo].[Results_of_sequencingSet]
ADD CONSTRAINT [FK_StrainsResults_of_sequencing]
    FOREIGN KEY ([Strains_Number_of_strain])
    REFERENCES [dbo].[StrainsSet]
        ([Number_of_strain])
    ON DELETE NO ACTION ON UPDATE NO ACTION;
GO
```

```
-- Creating non-clustered index for FOREIGN KEY 'FK_StrainsResults_of_sequencing'
CREATE INDEX [IX_FK_StrainsResults_of_sequencing]
ON [dbo].[Results_of_sequencingSet]
    ([Strains_Number_of_strain]);
GO

-- Creating foreign key on [Sequenators_Id] in table 'Runs_of_sequencingSet'
ALTER TABLE [dbo].[Runs_of_sequencingSet]
ADD CONSTRAINT [FK_SequenatorsRuns_of_sequencing]
    FOREIGN KEY ([Sequenators_Id])
    REFERENCES [dbo].[Sequenators_ModelsSet]
        ([Id])
    ON DELETE NO ACTION ON UPDATE NO ACTION;
GO

-- Creating non-clustered index for FOREIGN KEY 'FK_SequenatorsRuns_of_sequencing'
CREATE INDEX [IX_FK_SequenatorsRuns_of_sequencing]
ON [dbo].[Runs_of_sequencingSet]
    ([Sequenators_Id]);
GO

-- Creating foreign key on [Sequencing_kits_Id] in table 'Runs_of_sequencingSet'
ALTER TABLE [dbo].[Runs_of_sequencingSet]
ADD CONSTRAINT [FK_Sequencing_kitsRuns_of_sequencing]
    FOREIGN KEY ([Sequencing_kits_Id])
    REFERENCES [dbo].[Sequencing_kitsSet]
        ([Id])
    ON DELETE NO ACTION ON UPDATE NO ACTION;
GO

-- Creating non-clustered index for FOREIGN KEY 'FK_Sequencing_kitsRuns_of_sequencing'
CREATE INDEX [IX_FK_Sequencing_kitsRuns_of_sequencing]
ON [dbo].[Runs_of_sequencingSet]
    ([Sequencing_kits_Id]);
GO

-- Creating foreign key on [Types_of_sequencing_Id] in table 'Runs_of_sequencingSet'
ALTER TABLE [dbo].[Runs_of_sequencingSet]
ADD CONSTRAINT [FK_Types_of_sequencingRuns_of_sequencing]
    FOREIGN KEY ([Types_of_sequencing_Id])
    REFERENCES [dbo].[Types_of_sequencingSet]
        ([Id])
    ON DELETE NO ACTION ON UPDATE NO ACTION;
GO

--        Creating        non-clustered        index        for        FOREIGN        KEY
'FK_Types_of_sequencingRuns_of_sequencing'
CREATE INDEX [IX_FK_Types_of_sequencingRuns_of_sequencing]
ON [dbo].[Runs_of_sequencingSet]
    ([Types_of_sequencing_Id]);
GO

-- Creating foreign key on [Selection_locations_Id] in table 'StrainsSet'
ALTER TABLE [dbo].[StrainsSet]
ADD CONSTRAINT [FK_Selection_locationsStrains]
    FOREIGN KEY ([Selection_locations_Id])
    REFERENCES [dbo].[Selection_locationsSet]
        ([Id])
    ON DELETE NO ACTION ON UPDATE NO ACTION;
GO

-- Creating non-clustered index for FOREIGN KEY 'FK_Selection_locationsStrains'
```

```
CREATE INDEX [IX_FK_Selection_locationsStrains]
ON [dbo].[StrainsSet]
    ([Selection_locations_Id]);
GO

-- Creating foreign key on [Selection_sources_Id] in table 'StrainsSet'
ALTER TABLE [dbo].[StrainsSet]
ADD CONSTRAINT [FK_Selection_sourcesStrains]
    FOREIGN KEY ([Selection_sources_Id])
    REFERENCES [dbo].[Selection_sourcesSet]
        ([Id])
    ON DELETE NO ACTION ON UPDATE NO ACTION;
GO

-- Creating non-clustered index for FOREIGN KEY 'FK_Selection_sourcesStrains'
CREATE INDEX [IX_FK_Selection_sourcesStrains]
ON [dbo].[StrainsSet]
    ([Selection_sources_Id]);
GO

-- Creating foreign key on [Manufacturer_Id] in table 'Sequenators_ModelsSet'
ALTER TABLE [dbo].[Sequenators_ModelsSet]
ADD CONSTRAINT [FK_SequenatorsManufacturer]
    FOREIGN KEY ([Manufacturer_Id])
    REFERENCES [dbo].[Sequenators_ManufacturerSet]
        ([Id])
    ON DELETE NO ACTION ON UPDATE NO ACTION;
GO

-- Creating non-clustered index for FOREIGN KEY 'FK_SequenatorsManufacturer'
CREATE INDEX [IX_FK_SequenatorsManufacturer]
ON [dbo].[Sequenators_ModelsSet]
    ([Manufacturer_Id]);
GO

-- Creating foreign key on [Taxonomies_Id] in table 'StrainsSet'
ALTER TABLE [dbo].[StrainsSet]
ADD CONSTRAINT [FK_StrainsSet_Taxonomies]
    FOREIGN KEY ([Taxonomies_Id])
    REFERENCES [dbo].[Taxonomies_SpeciesSet]
        ([Id])
    ON DELETE NO ACTION ON UPDATE NO ACTION;
GO

-- Creating non-clustered index for FOREIGN KEY 'FK_StrainsSet_Taxonomies'
CREATE INDEX [IX_FK_StrainsSet_Taxonomies]
ON [dbo].[StrainsSet]
    ([Taxonomies_Id]);
GO

-- Creating foreign key on [Genus_Id] in table 'Taxonomies_SpeciesSet'
ALTER TABLE [dbo].[Taxonomies_SpeciesSet]
ADD CONSTRAINT [FK_TaxonomiesGenus]
    FOREIGN KEY ([Genus_Id])
    REFERENCES [dbo].[Taxonomies_GenusSet]
        ([Id])
    ON DELETE NO ACTION ON UPDATE NO ACTION;
GO

-- Creating non-clustered index for FOREIGN KEY 'FK_TaxonomiesGenus'
CREATE INDEX [IX_FK_TaxonomiesGenus]
ON [dbo].[Taxonomies_SpeciesSet]
```

```
    ([Genus_Id]);
GO

-- Creating foreign key on [AssembliesSet_Id] in table 'AssembliesResults_of_sequencing'
ALTER TABLE [dbo].[AssembliesResults_of_sequencing]
ADD CONSTRAINT [FK_AssembliesResults_of_sequencing_AssembliesSet]
    FOREIGN KEY ([AssembliesSet_Id])
    REFERENCES [dbo].[AssembliesSet]
        ([Id])
    ON DELETE NO ACTION ON UPDATE NO ACTION;
GO

-- Creating foreign key on [Results_of_sequencingSet_Id] in table
'AssembliesResults_of_sequencing'
ALTER TABLE [dbo].[AssembliesResults_of_sequencing]
ADD CONSTRAINT [FK_AssembliesResults_of_sequencing_Results_of_sequencingSet]
    FOREIGN KEY ([Results_of_sequencingSet_Id])
    REFERENCES [dbo].[Results_of_sequencingSet]
        ([Id])
    ON DELETE NO ACTION ON UPDATE NO ACTION;
GO

-- Creating non-clustered index for FOREIGN KEY
'FK_AssembliesResults_of_sequencing_Results_of_sequencingSet'
CREATE INDEX [IX_FK_AssembliesResults_of_sequencing_Results_of_sequencingSet]
ON [dbo].[AssembliesResults_of_sequencing]
    ([Results_of_sequencingSet_Id]);
GO

-- Creating foreign key on [AuthorsSet_Id] in table 'StrainsAuthors'
ALTER TABLE [dbo].[StrainsAuthors]
ADD CONSTRAINT [FK_StrainsAuthors_AuthorsSet]
    FOREIGN KEY ([AuthorsSet_Id])
    REFERENCES [dbo].[AuthorsSet]
        ([Id])
    ON DELETE NO ACTION ON UPDATE NO ACTION;
GO

-- Creating foreign key on [StrainsSet_Number_of_strain] in table 'StrainsAuthors'
ALTER TABLE [dbo].[StrainsAuthors]
ADD CONSTRAINT [FK_StrainsAuthors_StrainsSet]
    FOREIGN KEY ([StrainsSet_Number_of_strain])
    REFERENCES [dbo].[StrainsSet]
        ([Number_of_strain])
    ON DELETE NO ACTION ON UPDATE NO ACTION;
GO

-- Creating non-clustered index for FOREIGN KEY 'FK_StrainsAuthors_StrainsSet'
CREATE INDEX [IX_FK_StrainsAuthors_StrainsSet]
ON [dbo].[StrainsAuthors]
    ([StrainsSet_Number_of_strain]);
GO

-- Creating foreign key on [OrganizationsSet_Id] in table 'StrainsOrganizations'
ALTER TABLE [dbo].[StrainsOrganizations]
ADD CONSTRAINT [FK_StrainsOrganizations_OrganizationsSet]
    FOREIGN KEY ([OrganizationsSet_Id])
    REFERENCES [dbo].[OrganizationsSet]
        ([Id])
    ON DELETE NO ACTION ON UPDATE NO ACTION;
GO
```

```
-- Creating foreign key on [StrainsSet_Number_of_strain] in table 'StrainsOrganizations'
ALTER TABLE [dbo].[StrainsOrganizations]
ADD CONSTRAINT [FK_StrainsOrganizations_StrainsSet]
    FOREIGN KEY ([StrainsSet_Number_of_strain])
    REFERENCES [dbo].[StrainsSet]
        ([Number_of_strain])
    ON DELETE NO ACTION ON UPDATE NO ACTION;
GO

-- Creating non-clustered index for FOREIGN KEY 'FK_StrainsOrganizations_StrainsSet'
CREATE INDEX [IX_FK_StrainsOrganizations_StrainsSet]
ON [dbo].[StrainsOrganizations]
    ([StrainsSet_Number_of_strain]);
GO

-- --------------------------------------------------
-- Script has ended
-- --------------------------------------------------
```

ПРИЛОЖЕНИЕ Б

Код хранимой процедуры для генерации паспорта штамма

```
CREATE PROCEDURE PasportOfStrain
@NumberOfStrain int
AS
SELECT Number_of_strain, Taxonomies_GenusSet.Name, Taxonomies_SpeciesSet.Name,
Pathogenicity, Date_of_deposit, Validate_Kraken,
Forms_of_depositSet.Name, Selection_sourcesSet.Name_RUS, Selection_locationsSet.Name,
Description
FROM StrainsSet, Taxonomies_GenusSet, Taxonomies_SpeciesSet, Forms_of_depositSet,
Selection_sourcesSet, Selection_locationsSet
WHERE Taxonomies_GenusSet.Id = Taxonomies_SpeciesSet.Genus_Id
AND Taxonomies_SpeciesSet.Id = StrainsSet.Taxonomies_Id
AND Forms_of_depositSet.Id = StrainsSet.Forms_of_deposit_Id
AND Selection_sourcesSet.Id = StrainsSet.Selection_sources_Id
AND Selection_locationsSet.Id = StrainsSet.Selection_locations_Id
AND Number_of_strain = @NumberOfStrain
```

Фрагмент кода для генерации паспорта штамма

```
            using (SqlConnection conn = new
SqlConnection(Properties.Settings.Default.CatalogForObolenskConnectionString))
            {
                conn.Open();
                cmd = new SqlCommand("PasportOfStrain", conn);
                cmd.CommandType = CommandType.StoredProcedure;
                cmd.Parameters.AddWithValue("@NumberOfStrain",
Convert.ToInt32(number_of_strainTextBox.Text));
                cmd.ExecuteNonQuery();

this.pasportOfStrainTableAdapter.Fill(this.catalogForObolenskDataSet.PasportOfStrain,
Convert.ToInt32(number_of_strainTextBox.Text));
                report1.Preview = previewControl1;
                report1.Show();
            }
```

Фрагмент кода для добавления автора к штамму

```
        private void authorsTextBox_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.KeyCode == Keys.Enter)
            {
                //проверяем список авторов штамма
                if (authorsListBox.FindStringExact(authorsTextBox.Text) >= 0)
                {
                    authorsTextBox.Text = null;
                }
                //проверяем список всех авторов (создаем, если такого не существует)
                else if
(authorsTextBox.AutoCompleteCustomSource.IndexOf(authorsTextBox.Text) == -1)
                {
                    using (SqlConnection conn = new
SqlConnection(Properties.Settings.Default.CatalogForObolenskConnectionString))
                    {
                        conn.Open();
                        SQL_select = String.Format("INSERT AuthorsSet(Name) VALUES (@Name)
");
                        cmd = new SqlCommand(SQL_select, conn);
                        cmd.Parameters.AddWithValue("@Name", authorsTextBox.Text);
```

```
                        cmd.ExecuteNonQuery();
                    }
                    CreateStrainsAuthors();
                }
                //добавляем существующего автора к данному штамму
                else CreateStrainsAuthors();
            }
        }

        private void CreateStrainsAuthors()
        {
            using (SqlConnection conn = new
SqlConnection(Properties.Settings.Default.CatalogForObolenskConnectionString))
            {
                conn.Open();
                SQL_select = String.Format("INSERT StrainsAuthors(Strains_number_of_strain,
Authors_Id)" +
"SELECT @number_of_strain, Id FROM AuthorsSet WHERE Name = @Name");
                cmd = new SqlCommand(SQL_select, conn);
                cmd.Parameters.AddWithValue("@number_of_strain",
number_of_strainTextBox.Text);
                cmd.Parameters.AddWithValue("@Name", authorsTextBox.Text);
                cmd.ExecuteNonQuery();
            }
            authorsTextBox.Text = null;

///this.strainsAuthorsTableAdapter.Fill(this.catalogObolenskDataSet.StrainsAuthors);
            this.authorsSetTableAdapter.FillBy(this.catalogForObolenskDataSet.AuthorsSet,
((int)(System.Convert.ChangeType(number_of_strainTextBox.Text, typeof(int)))));
            ViewAuthors();
        }
```