МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных систем

Факультет автоматики и вычислительной техники

Кафедра систем автоматизации управления

**РЕАЛИЗАЦИЯ ИНФОРМАЦИОННОЙ**

**СИСТЕМЫ ПТИЦЕФАБРИКИ**

Пояснительная записка

Отчёт по дисциплине

«Объектно-ориентированное программирование»

ТПЖА 090302.015 ПЗ

Разработали студенты гр. ИТб-2301-01-00\_\_\_\_\_\_\_\_\_\_\_\_/ Ласкин М.В./

/ Морозов И.В./

(подпись)

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_/ Кузьминых М.М./

(подпись)

Работа защищена с оценкой «\_\_\_\_\_\_\_\_\_\_\_\_\_\_» «\_\_»\_\_\_\_\_\_\_\_\_\_2022 г.

Киров 2022

Реферат

Ласкин М.В. Морозов И.В. Реализация информационной системы учета работы клининговой компании: ТПЖА. 090302.015 ПЗ: Отчет по курсовой работе / ВятГУ, каф. САУ; рук. М.М. Кузьминых. – Киров, 2022. ПЗ 47 с., 16 рис., 1 табл., 5 источников, 4 прил.

ПЕРЕЧЕНЬ КЛЮЧЕВЫХ СЛОВ: ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ, ИНФОРМАЦИОННАЯ СИСТЕМА, C#, ОБЪЕКТНО-ОРИЕНТИРОВННОЕ ПРОГРАММИРОВАНИЕ, ENTITY FRAMEWORK, БАЗА ДАННЫХ.

Объектом исследования является высокоуровневый язык программирования C#, разработка с использованием современных информационных технологий.

Цель курсовой работы – закрепить теоретические знания, полученные в ходе освоения курса и приобрести практические навыки проектирования и программирования с использованием объектно-ориентированного подхода.

Реализована информационная система с использованием Entity Framework в среде программирования Visual Studio.

Разработано приложение, осуществляющее добавление клиентов в БД, расчет и оформление заявок, формирование отчетов за выбранный период времени.

|  |  |
| --- | --- |
| **Содержание**  Изм.  Лист  № докум.  Подпись  Дата  Лист  1  ТПЖА 090302.015 ПЗ  Разраб.  *Ласкин М.В. Морозов И.В.*  Провер.  Кузьминых М.М.  Реценз.  Н. Контр.  Утверд.  Литер  Листов  2  *Кафедра САУ,*  *гр. ИТб-2301-01-00* |  |
| Введение……………………………………………………………………….. | 3 |
| 1. Обзор использованной литературы……………………………………….. | 4 |
| 1. Анализ предметной области.....................................................…………… | 5 |
| 2.1 Процессы, протекающие в предметной области …………….............. | 5 |
| 2.2 Классы предметной области...................………………………………. | 6 |
| 1. Реализация классов на языке программирования C#..................……........ | 8 |
| 3.1 Отношение между классами...........................................……………… | 8 |
| 1. Создание базы данных с использованием Entity Framework..................... | 9 |
| 4.1 Создание контекста для базы данных....……...…………………… | 9 |
| 4.2 Миграции.....................................................................……….……... | 9 |
| 4.3 Создание БД................................................................……….……... | 9 |
| 1. Разработка приложения, реализующего базовую функциональность.. | 11 |
| 5.1 Вход в приложение....……...…………………..............................…..... | 11 |
| 5.2 Окна работников птицефабрики………………………………………. | 11 |
| 5.2.1 Окно работника репродуктора………………………............... | 12 |
| 5.2.2 Окно работника инкубатора…………………………................ | 12 |
| 5.2.3 Окно работника птичника……………………………………... | 13 |
| 5.2.4 Окно работника комплекса производства яиц……………….. | 14 |
| 5.2.5 Окно работника цеха сортировки яиц………………………… | 15 |
| 5.2.6 Окно работника убойно перерабатывающего комплекса……. | 16 |
| 5.2.7 Окно работника цеха производства меланжа………………… | 17 |
|  |  |
|  |  |
|  |  |
| 5.3 Окно руководителя………………………………………………………  Изм.  Лист  № докум.  Подпись  Дата  Лист  2  ТПЖА 090302.015 ПЗ  Разраб.  *Ласкин М.В. Морозов И.В.*  Провер.  Кузьминых М.М.  Реценз.  Н. Контр.  Утверд.  Литер  Листов  2  *Кафедра САУ,*  *гр. ИТб-2301-01-00* | 17 |
| 5.3.1 Вкладка с состоянием цехов и комплексов……………..………… | 18 |
| 5.3.2 Вкладка готовой продукции…………………................................... | 18 |
| 5.3.3 Вкладка с партиями яиц и кууриц..................................................... | 19 |
| 5.3.4 Вкладка с созданием поквартального отчета……………………... | 19 |
| 5.3.5 Вкладка управления………………………………………………… | 20 |
| Заключение………………………………………………………………..…… | 21 |
| Приложение А (обязательное) Ключевые классы………………………….. | 22 |
| Приложение Б (обязательное) Класс вспомогательных методов…………. | 29 |
| Приложение В (обязательное) Контекст БД………………………………… | 41 |
| Приложение Г (справочное) Библиографический список………………..... | 47 |

# Введение

Объектно-ориентированное программирование (ООП) — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования.

Целью курсовой работы является закрепление теоретических знаний об объектно-ориентированном подходе программирования на практике, и приобретение практических навыков проектирования информационной системы.

Основная целевая платформа для разработки – Microsoft .Net Framework, язык программирования – C#, библиотека ORM – Entity Framework.

Задачи в рамках написания курсовой работы:

· Овладение основами объектно-ориентированного анализа предметной области и выявления классов и отношений между ними;

· Освоение приемов объектно-ориентированного программирования в ходе реализации выявленных классов на выбранном языке программирования;

· Знакомство с библиотечными классами выбранной платформы программирования;

· Знакомство с приемами работы с базами данных с использованием библиотек, реализующих объектно-реляционное отображение (ORM).

# 

# **Обзор использованной литературы**

В первую очередь необходимо изучить объектно-ориентированную технологию на базе фреймворка .NET – Entity Framework для работы с реляционными данными. Для этого обратимся к сайтам в интернете [1-3, приложение К]. Данные статьи демонстрируют основные принципы работы с данными при использовании Entity Framework.

Особое внимание было уделено внесению изменений в имеющуюся базу данных при помощи миграций [4, приложение К]. Из статьи можно понять все преимущества миграций по сравнению с ручным изменением данных.

Для построения диаграммы классов необходимо изучить виды отношений между классами и объектами. Для этого обратимся к статье электронного ресурса METANIT.COM [5, приложение К]. Из статьи можно узнать о трех основных видах связей: ассоциация, композиция и агрегация. А также рекомендации по их применению.

# Анализ предметной области

Птицефабрика – крупное птицеводческое предприятие, производящее яйца и мясо птицы. Птицефабрики размещают в основном около крупных городов и промышленных центров. Весь технологический процесс на птицефабриках организован на промышленной основе.

* 1. Процессы, протекающие в предметной области

Начинается процесс с репродуктора и родительского стада кур, которое комплектуют несколько раз в год одновозрастной птицей. Это позволяет равномерно получать яйца круглый год, равномерно выращивать молодняк. Информацию вышедших партиях яиц и состоянии родительского оператор фиксирует в соответствующих полях формы. Загрузка недостающей птицы производит тот же оператор фиксируя из каких партий взрослой птицы были позаимствованы особи.

Далее цыплят выводят в инкубаторах, установленных в инкубатории. На инкубирование обычно берутся яйца от кур яичных пород старше 8-месячного возраста. На птицефабриках мясного направления (бройлерных) цыплят выращивают до 8-недельного возраста и используют на мясо, когда они весят 1,5—1,7 кг. Данные о загруженных партиях яиц и выгруженных партиях цыплят фиксируются.

Затем готовый молодняк поступает в птичник, где он прививается, сортируется по половому признаку и состоянию здоровья, и распределяется По соответствующим цехам выращивания молодняка. Все время содержания в птичнике цыплят обеспечивают комом и водой, а также в цехе поддерживается определенная температура, оптимальная для каждого кросса птицы. Данные о загруженных партиях цыплят и выгруженных партиях молодняка фиксируются.

После поступления в цех выращивания молодняка курам обеспечивается оптимальный температурный и световой режим, а также подача корма и воды в неограниченном количестве, до их полного взросления и готовности к дальнейшему использованию. Данные о загруженных партиях молодняка и выгруженных партиях взрослой птицы фиксируются.

Главным цехом на птицефабрике яичного направления является цех промышленного стада. Промышленное стадо пополняется и обновляется за счет кур в возрасте 120—140 суток, выращенных в птичнике. Данные о загруженных партиях взрослой птицы, выгруженных яйцах фиксируются.

Далее яйца из цех промышленного стада поступают в цех сортировки, где отсеиваются поврежденные экземпляры и сортируются по категории, а после упаковываются для продажи. Данные о загруженных яйцах, выгруженных партиях яиц и отбраковки фиксируются. Данные о загруженных партиях яиц и выгруженных партиях меланжа так же фиксируются.

Отсеянные экземпляры попадают в цех производства меланжа, где из них извлекают содержимое и разливают по бутылкам. Количество произведенной продукции фиксируются.

Главным цехом на птицефабрике мясного направления является убойно перерабатывающий комплекс. цыплят, выращенных в птичнике, используют на мясо, когда они достигают 8-недельного возраста и весят 1,5—1,7 кг. Основными этапами являются убой, обдирка и предварительная разделка. Далее тушки могут либо быть разделаны на фабрикаты, либо поставляться в целом виде, в зависимости от спроса в конкретном регионе. Убойно перерабатывающий комплекса фиксирует количество поставленных бройлеров и количество произведенных фабрикатов и/или тушек.

* 1. Классы предметной области

Выявленные классы предметной области указаны в таблице 1.

Таблица 1 – классы предметной области

|  |  |  |
| --- | --- | --- |
| № п/п | Категория | Классы |
| 1 | Физические или материальные объекты | Цеха, комплексы, партии |
| 2 | Спецификации, описания объектов | Тип птицы, вместимость цеха. |
| 3 | Места | Птицефабрика |
| 4 | Транзакции | Загрузка и выпуск партий из цехов, комплексов |
| 5 | Элементы транзакций | Партии |
| 6 | Роли людей | Руководитель, работники цехов |
| 7 | Контейнеры других объектов | Цеха и комплексы производств |
| 8 | Содержимое контейнеров | Партии яиц, птиц |
| 9 | Организации | Птицефабрика |
| 10 | События |  |
| 11 | Процессы | Загрузка и выпуск партий из цехов, комплексов |

Выявленные классы, их отношения и свойства, отраженные на диаграмме классов UML, изображены на рисунке 1.

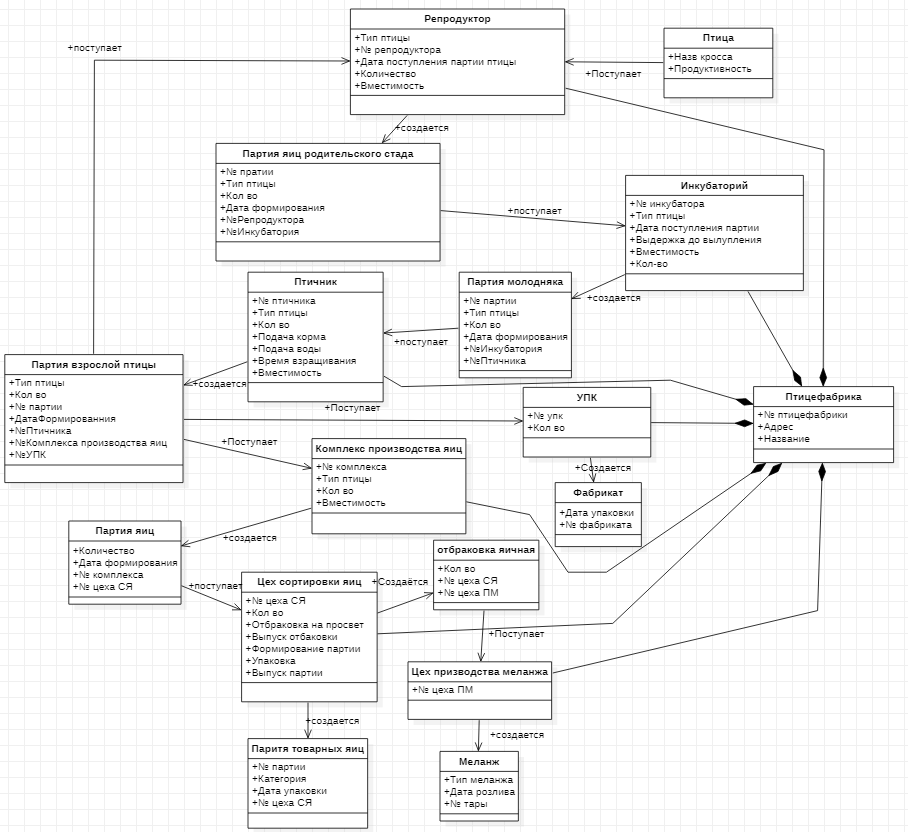


Рисунок 1 – Диаграмма классов UML

В данном разделе проанализирована предметная область, определены протекающие в ней процессы. Выявлены классы при помощи метода анализа по категориям. Создана диаграмма классов UML, отображающая выявленные классы с атрибутами и связями между ними.

# Реализация классов на языке программирования C#

C Sharp – объектно-ориентированный язык программирования, имеющий в своём арсенале широкий набор библиотек. Из-за большого разнообразия синтаксических конструкций и возможности работать с платформой .NET, C# позволяет быстрее, чем любой другой язык, разрабатывать программные решения. Именно на этом языке будет написано приложение.

В предметной области был выделен ряд ключевых классов: Reproductor, Incubator, UPK, Ptichnic, ComplexProizvodstvaEggs, CehSortEggs. Код ключевых классов представлен в приложении А. Для всех классов реализованы конструкторы, задающие корректное начальное состояние объектов.

Между классами репродуктор и партия яиц родительского класса реализован вид отношения ассоциация. Данное отношение продемонстрировано в приложении А. Ассоциация – это отношение, при котором объекты одного типа неким образом связаны с объектами другого типа.

Между классами птицефабрика и цехами реализован вид отношений композиция. Листинг данных классов находится в приложении А. Композиция определяет отношение "имеет".

Таким образом, в 3 разделе представлена реализация выявленных классов. Осуществлены описанные отношения между классами предметной области.

1. **Создание базы данных с использованием Entity Framework**

В рамках курсовой работы в качестве библиотеки ORM использовалась Entity Framework. Entity Framework — это решение для работы с базами данных, которое используется в программировании на языках семейства.NET. Оно позволяет взаимодействовать с СУБД с помощью сущностей (entity), а не таблиц. Также код с использованием EF пишется гораздо быстрее.

Например, работая с базами данных напрямую, разработчик должен беспокоиться о подключении, подготовке SQL и параметров, отправке запросов и транзакций. На Entity Framework всё это делается автоматически — программист же работает непосредственно с сущностями и только говорит EF, что нужно сохранить изменения.

* 1. Создание контекста для базы данных

Для взаимодействия с базой данных был создан класс ApplicationContext. Он является наследником DbContext. Код данного типа предоставлен в приложении З.

Для определения структуры БД используются аннотации: Required, MaxLength, MinLength. Перечисленные аннотации используются во многих классах, в том числе в уже представленных в приложении А.

* 1. **Миграции**

Для изменения моделей в Entity Framework, которые входят в контекст данных, например, добавление или удаление некоторых свойств, необходимо, чтобы база данных также применяла эти изменения. Для этого используется такой инструмент как миграция.

Сгенерированные Entity Framework миграции представлены в приложении И.

4.3 Создание БД

Диаграмма базы данных предоставлена на рисунке 2.

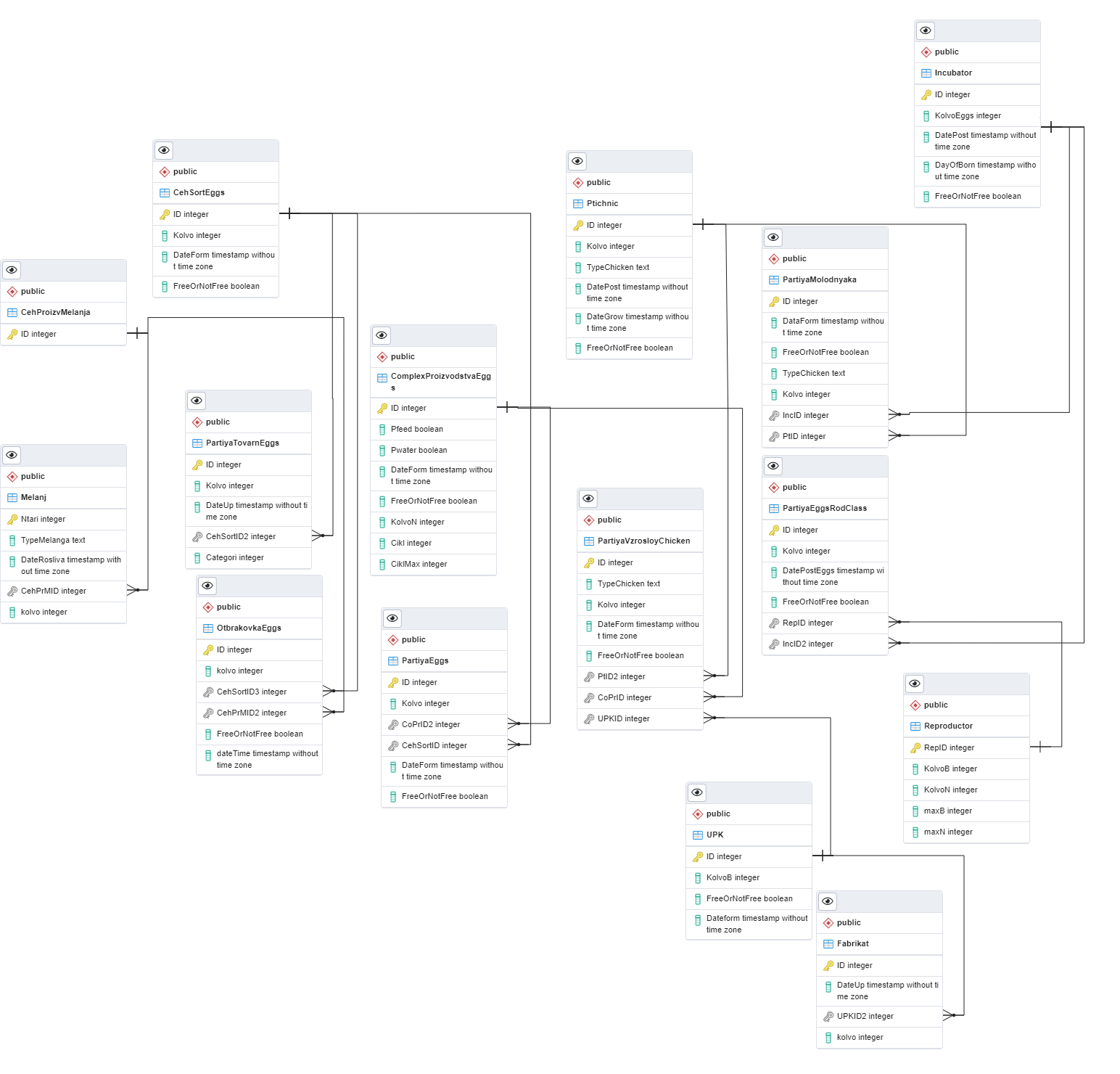


Рисунок 2 – Схема базы данных

На схемы отображены классы с их атрибутами, типами данных и ограничениями целостности. Между классами расставлены связи.

По итогу в данном разделе был создан контекст для работы с базой данных. Применена миграция для изменения базы данных. Смоделирована физическая модель данных.

1. **Разработка приложения, реализующего базовую функциональность**

В данном разделе приведена демонстрация работы программы. Ключевые фрагменты кода, реализующие работу с созданными классами представлены в приложении К.

В приложении разработан пользовательский индивидуальный интерфейс, при помощи технологии построения интерфейса WPF, для двух ролей: администратор и бригадир. Интерфейс для данных ролей разработан с учетом задач, которые должны решаться в выбранной предметной области для конкретной роли.

* 1. **Вход в приложение**

Окно для входа в приложение под определенной ролью представлена на рисунке 3.

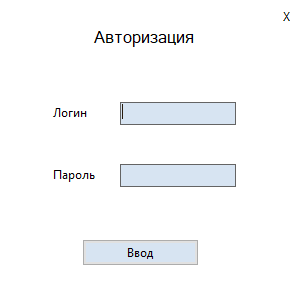


Рисунок 3 – Вход в приложение

Каждый администратор и сотрудник имеет свой уникальный логин и пароль, хранящиеся в hash, для входа в приложение.

**5.2 Окна работников птицефабрики**

Возможности работников: формирование партий с последующим добавлением в базу данных, добавление партий в цеха и отправление партий на следующие этапы.

В каждом окне имеется кнопка “Обновить данные”, она необходима для обновления данных в таблицах с выводом данных о партиях и цехах. Кнопки добавления в цеха позволяют добавлять партии яиц или птиц в цеха для инкубации/взращивания/производства яиц/сортировки яиц/производства фабриката/производства меланжа.

**5.2.1 Окно работника репродуктора**

Данное окно предназначено для формирования партий яиц из репродуктора, либо добавление недостающих особей в репродуктор. Данное окно представлено на рисунке 4.

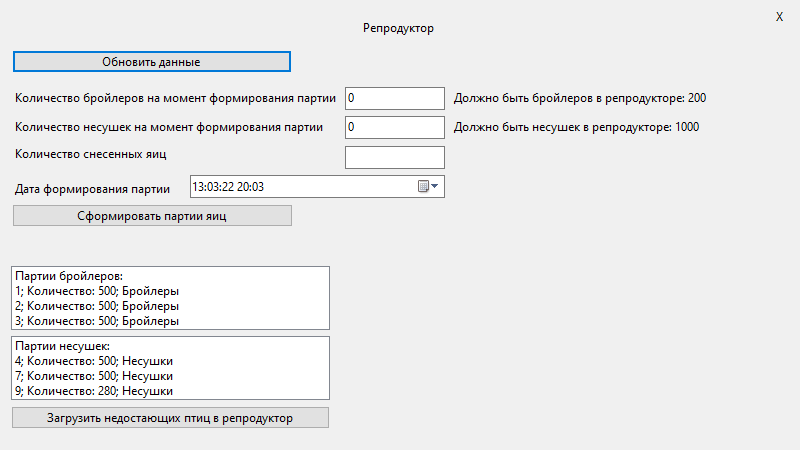


Рисунок 4 – Окно работника репродуктора

После заполнения всех полей и нажатия кнопки “Сформировать партии яиц”, в БД фиксируются количество несушек и бройлеров и партии яиц с количеством менее или равным вместимости инкубатора. Недостающих птиц можно восполнить из партий доступных в списках выбрав их и нажав на кнопку “Загрузить недостающих птиц в репродуктор”. Подсказка о том, сколько птиц должно находиться в репродукторе не статична, а настраивается администратором.

**5.2.2 Окно работника инкубатора**

Окно предназначено для занесения партии яиц из репродуктора в инкубаторий и формирования партий молодых бройлеров и несушек. Окно изображено на рисунке 5.

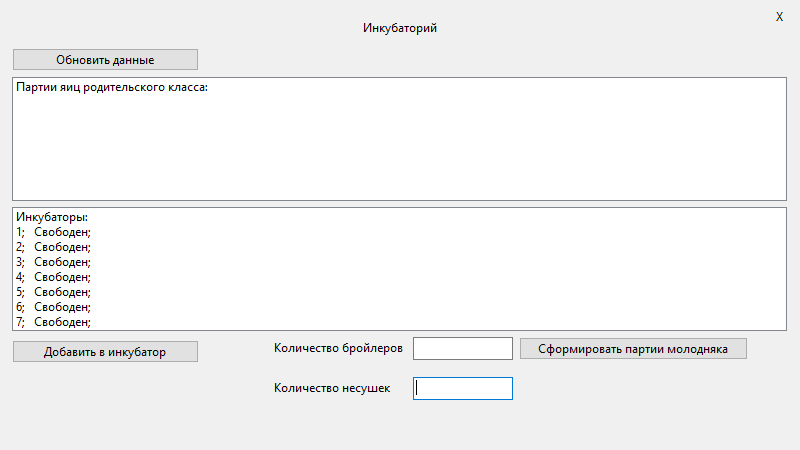


Рисунок 5 – Окно работника инкубатора

После выбора партии и инкубатора из соответствующих списков и по нажатию на кнопку “Добавить в инкубатор” в информация о партии яиц отобразится в инкубаторе, произойдет расчет даты вылупления и поменяется статут инкубатора. После выбора инкубатора, заполнения полей количества бройлеров и несушек и нажатия на кнопку “Сформировать партии молодняка”, в БД поступает информация о партиях при условии окончания периода вылупления и корректных данных о количестве особей.

**5.2.3 Окно работника птичника**

Окно предназначено для занесения партии молодняка в птичник и формирования партий бройлеров и несушек. Окно изображена на рисунке 6.

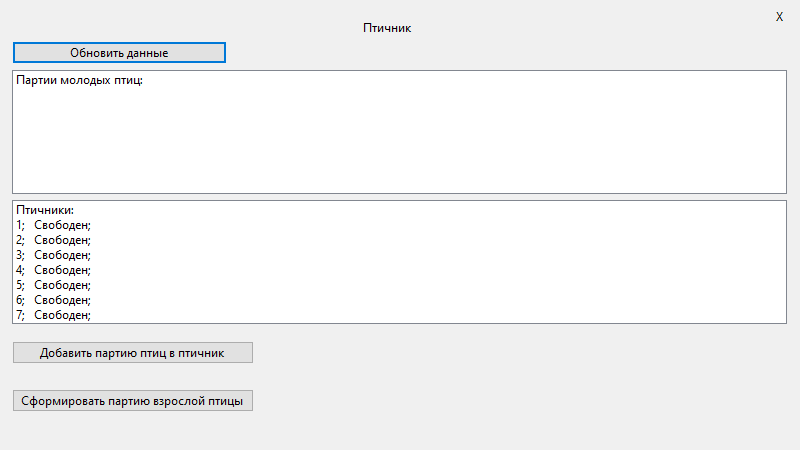


Рисунок 6 – Окно работника птичника

Для занесения партии в птичник работник выбирает нужную ему партию и птичник и нажимает кнопку “Добавить партию птиц в птичник”. После этого в поле птичника отобразится информация о загруженной партии, расчетная дата выроста и статус птичника. После выбора птичника и нажатия на кнопку “Сформировать партию взрослой птицы”, в БД поступает информация о партии при условии наступления расчетной даты выроста.

**5.2.4 Окно работника комплекса производства яиц**

Данное окно предназначено для внесения партии несушек в комплекс производства яиц и формирования партий яиц. Ознакомиться с данной формой можно на рисунке 7.

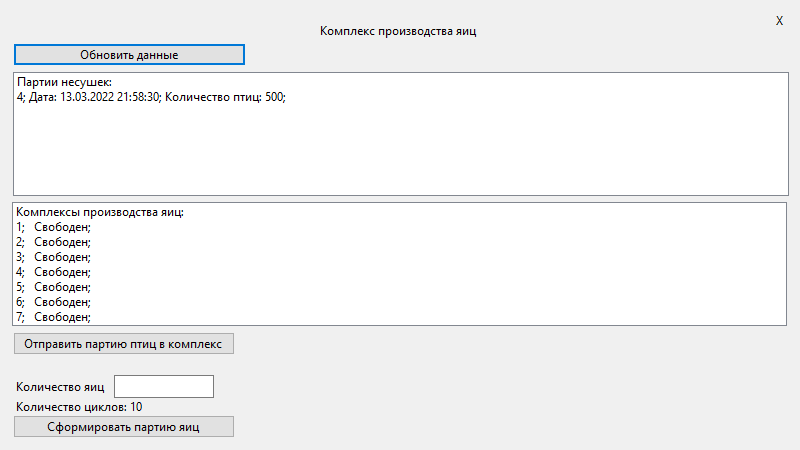


Рисунок 7 – Окно работника комплекса производства яиц

После выбора партии и комплекса из соответствующих списков и по нажатию на кнопку “Отправить партию птиц в комплекс” в комплексе отобразится информация о партии несушек и количество циклов сбора яиц, пройденных ей. После выбора птичника, заполненияполя количества яиц и нажатия на кнопку “Сформировать партию яиц”, в БД поступает информация о партии при условии, что количество пройденных циклов у партии, располагающейся в комплексе, не превышает порогового значения, установленного администратором. Если пороговое значение достигнуто, то птичник освобождается.

**5.2.5 Окно работника цеха сортировки яиц**

Окно предназначено для сортировки партии яиц по категориям. Окно изображена на рисунке 8.

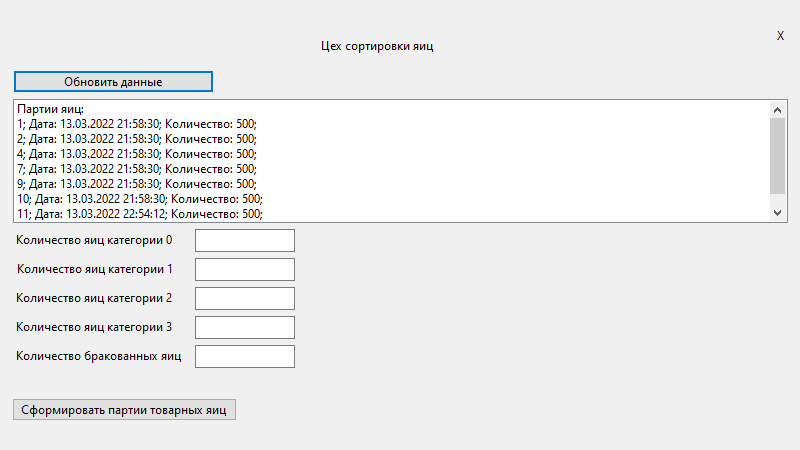


Рисунок 8 – Окно работника цеха сортировки яиц

После выбора партии яиц, заполненияполей количества яиц по категориям и нажатия на кнопку “Сформировать партии товарных яиц”, в БД поступает информация о партиях при условии, что общее указанное количество яиц не превышает изначальное.

**5.2.6 Окно работника убойно перерабатывающего комплекса**

Окно предназначено для преобразования партий бройлеров в партии фабрикатов. Окно изображена на рисунке 9.

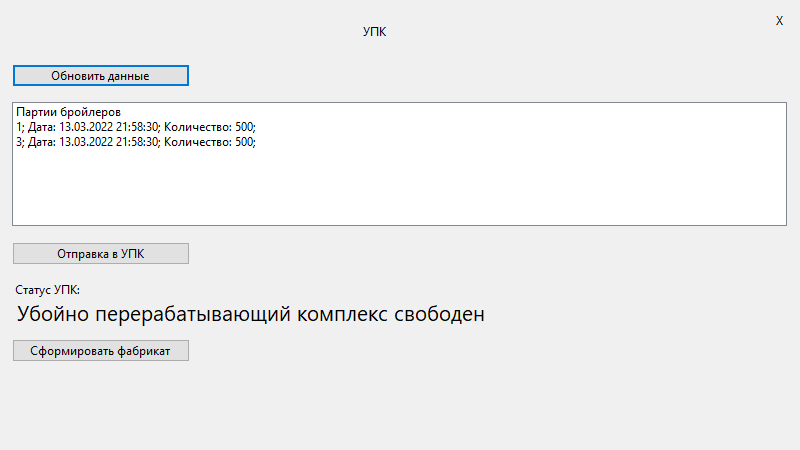


Рисунок 9 – Окно работника УПК

Для занесения партии в УПК работник выбирает нужную ему партию и нажимает кнопку “Отправить в УПК”. После этого статус птичника поменяется. После нажатия на кнопку “Сформировать фабрикат”, в БД поступает информация о партии фабриката.

**5.2.7 Окно работника цеха производства меланжа**

Окно предназначено для преобразования поврежденных яиц в меланж. Интерфейс данной формы представлен на рисунке 10.

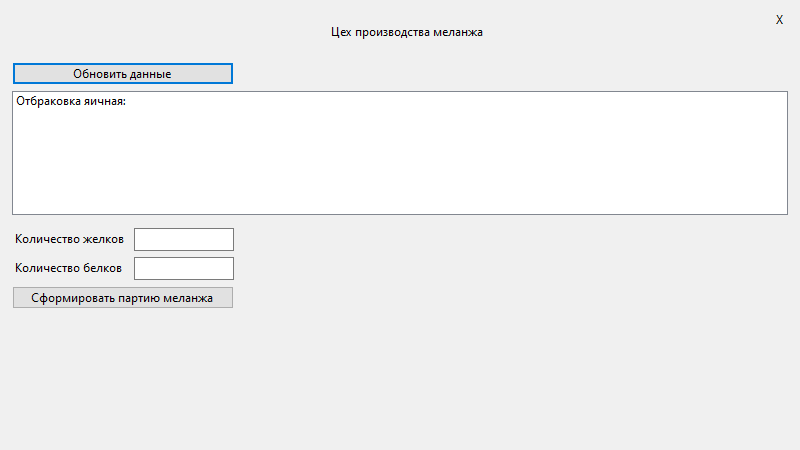


Рисунок 10 – Окно работника цеха производства меланжа

Для формирования меланжа работник выбирает нужную ему партию, заполняет формы и нажимает кнопку “Сформировать партию меланжа”. После нажатия в БД поступает информация о партиях меланжа.

**5.3 Окно руководителя**

Возможности руководителя: просмотр состояния всех цехов, комплексов, партий яиц, молодняка, несушек, бройлеров, готовой продукции. Удаление партий яиц, молодняка, несушек, бройлеров, готовой продукции и обнуление цехов и комплексов. Формирование квартального отчета по готовой продукции в виде Excel документов. Изменение необходимого количества бройлеров и несушек в репродукторе, количества циклов несения у несушек в комплексе производства яиц.

В верхней части формы находится навигация в виде вкладок, в правом углу находится кнопка выхода из окна в окно авторизации. Реализацию можно увидеть на рисунках 11-15.

**5.3.1 Вкладка с состоянием цехов и комплексов**

Вкладка предназначена для вывода состояния цехов и комплексов с возможностью очистки (обнуления цеха). Описанная вкладка располагается на рисунке 11.

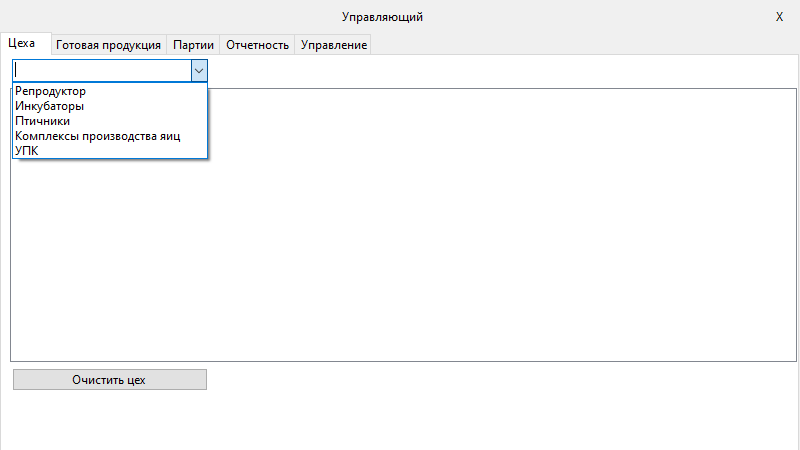


Рисунок 11 – Вкладка с состоянием цехов и комплексов

Функционал: выбор типа вывода цехов и комплексов с возможностью очистки цеха.

**5.3.2 Вкладка с готовой продукцией**

Вкладка предназначено для вывода готовой продукции с возможностью её удаления. Данная вкладка представлена на рисунке 12.

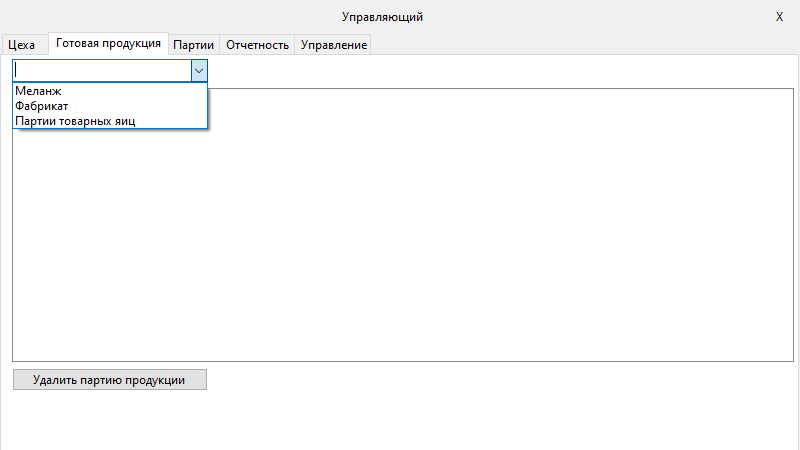


Рисунок 12 – Вкладка с готовой продукцией

Функционал: выбор типа вывода готовой продукции с возможностью её удаления.

**5.3.3 Вкладка с партиями яиц/куриц**

Вкладка предназначена для вывода партий с возможностью их удаления. Данная вкладка представлена на рисунке 13.

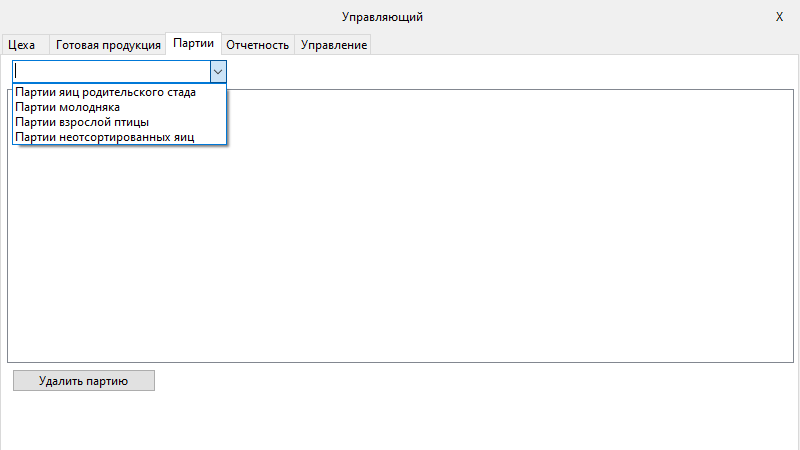


Рисунок 13 – Вкладка с партиями яиц/куриц

Функционал: выбор типа вывода партий с возможностью их удаления.

**5.3.4 Вкладка с созданием поквартального отчета**

Вкладка предназначена для создания поквартальных отчетов по готовой продукции в виде Excel документов. Данная вкладка представлена на рисунке 14, вид отчета из программы Excel представлен на рисунке 15.

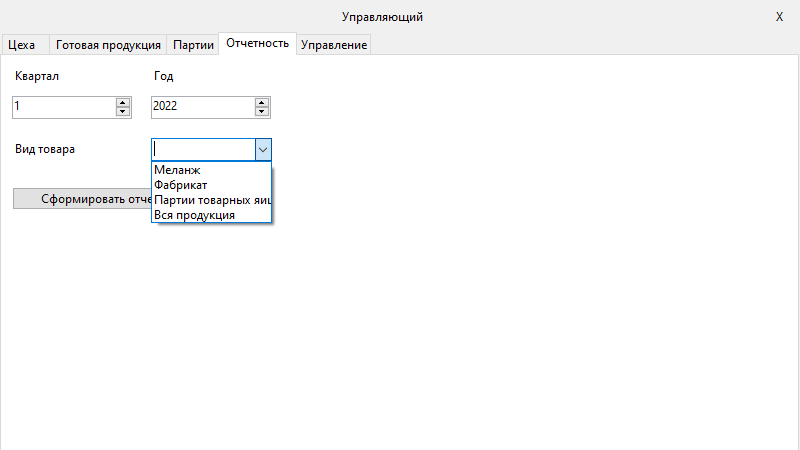


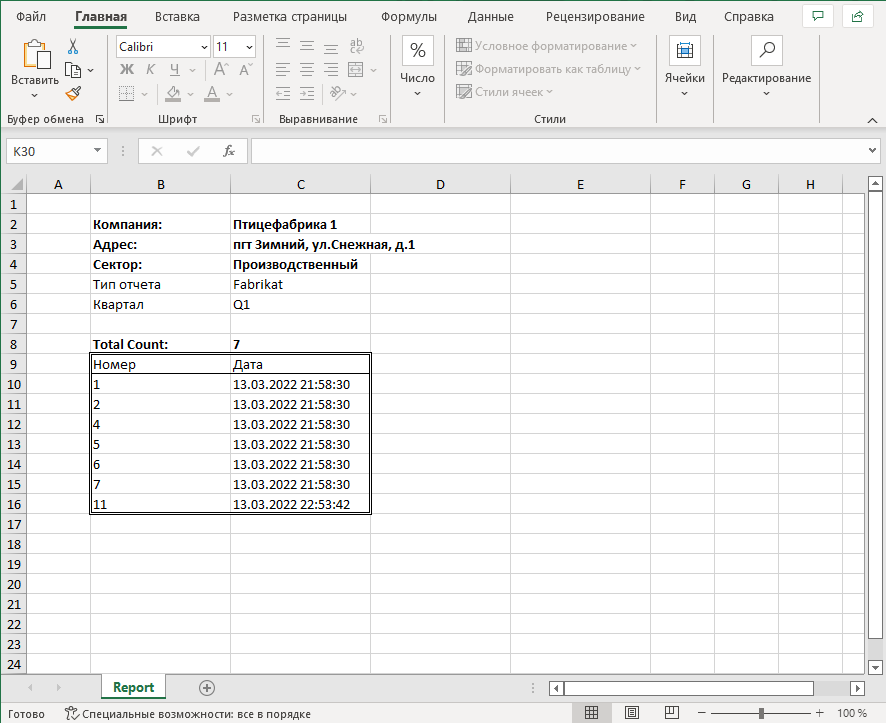
Рисунок 14 – Вкладка с партиями яиц/куриц

Рисунок 15 – Вид отчета из программы Excel

Функционал: выбор года, квартала, типа продукции в отчете. Создание отчета в виде Excel документа по вышеперечисленным критериям.

**5.3.5 Вкладка управления**

Вкладка предназначена для возможности изменения необходимого количества птиц в репродукторе и изменения количества циклов несения у взрослых несушек. Данная вкладка представлена на рисунке 16.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 16 – Вкладка управления

Функционал: изменение необходимого количества бройлеров и несушек в репродукторе, изменение количества циклов несения.

В данном разделе представлена реализация ключевых фрагментов кода, реализующих работу с библиотекой классов. Также были проиллюстрированы и подробно описаны все окна и страницы пользовательского приложения для ролей. Описаны возможности приложения для реализации перечня задач для каждой роли.

**Заключение**

В ходе выполнения курсовой работы были изучены основы объектно-ориентированного анализа предметной области, выявлены классы и отношения между ними. Для реализации выявленных классов были освоены приемы ООП, также была рассмотрена ORM система EntityFraemwork. Спроектирована и реализована модель базы данных. Реализованы все поставленные задачи предметной области.

**Приложение А**

(обязательное)

**Ключевые классы**

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace WinFormsLibrary1.Entity

{

public class CehPererabotkiOthodov

{

[Key]

[Required]

public int ID { get; set; }

[Required]

[MaxLength(100)]

public string NormsPererab { get; set; }

}

}

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace WinFormsLibrary1.Entity

{

public class CehProizvMelanja

{

[Key]

[Required]

public int ID { get; set; }

[ForeignKey("CehPrMID")]

public ICollection<Melanj> Melanjs { get; set; }

[ForeignKey("CehPrMID2")]

public ICollection<OtbrakovkaEggs> OtbrakovkaEggses { get; set; }

}

}

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System;

namespace WinFormsLibrary1.Entity

{

public class CehSortEggs

{

[Key]

[Required]

public int ID { get; set; }

public bool FreeOrNotFree { get; set; }

public DateTime DateForm { get; set; }

public int Kolvo { get; set; }

[ForeignKey("CehSortID")]

public ICollection<PartiyaEggs> PartiyaEggses { get; set; }

[ForeignKey("CehSortID2")]

public ICollection<PartiyaTovarnEggs> PartiyaTovarnEggses { get; set; }

[ForeignKey("CehSortID3")]

public ICollection<OtbrakovkaEggs> OtbrakovkaEggses { get; set; }

public void Otbracovka() { }

public void FormPartii() { }

public void Upacovca() { }

public void VipuscPartii() { }

}

}

using System;

using System.Collections.Generic;

using System.Text;

using System.ComponentModel.DataAnnotations;

namespace WinFormsLibrary1.Entity

{

public class Chicken

{

[Key] // Первичный ключ

public int ID { get; set; }

[Required] // NOT NULL

[MaxLength(100)] public string TypeChiсken { get; set; } // Максимальная длинна

}

}

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System;

namespace WinFormsLibrary1.Entity

{

public class ComplexProizvodstvaEggs

{

[Key]

[Required]

public int ID { get; set; }

[Required]

public int KolvoN { get; set; }

public bool FreeOrNotFree { get; set; }

public DateTime DateForm { get; set; }

public int Cikl { get; set; }

public int CiklMax { get; set; }

[Required]

public bool Pfeed { get; set; }

[Required]

public bool Pwater { get; set; }

[ForeignKey("CoPrID")]

public ICollection<PartiyaVzrosloyChicken> PartiyaVzrosloyChickens { get; set; }

[ForeignKey("CoPrID2")]

public ICollection<PartiyaEggs> PartiyaEggses { get; set; }

}

}

using System.ComponentModel.DataAnnotations;

using System;

namespace WinFormsLibrary1.Entity

{

public class Fabrikat

{

[Key]

[Required]

public int ID { get; set; }

public DateTime DateUp { get; set; }

public int UPKID2 { get; set; }

}

}

using System.ComponentModel.DataAnnotations;

using System;

using System.ComponentModel.DataAnnotations.Schema;

using System.Collections.Generic;

namespace WinFormsLibrary1.Entity

{

public class Incubator

{

[Key]

[Required]

public int ID { get; set; }

public int KolvoEggs { get; set; }

[Required]

public DateTime DatePost { get; set; }

public DateTime DayOfBorn { get; set; }

public bool FreeOrNotFree { get; set; }

//public int Temp; //Не нада метод

//public int vlagnost; //Не нада метод

[ForeignKey("IncID2")]

//public int PartiyaEggsRodID { get; set;}

public ICollection<PartiyaEggsRodClass> PartiyaEggsRodClass { get; set; }

[ForeignKey("IncID")]

public ICollection<PartiyaMolodnyaka> PartiyaMolodnyakas { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.Text;

using System.ComponentModel.DataAnnotations;

using WinFormsLibrary1;

using System.Linq;

namespace WinFormsLibrary1.Entity

{

public class LogPas

{

[Required]

public int ID { get; set; }

[Required]

public string login { get; set; }

[Required]

public string password { get; set; }

[Required]

public string role { get; set; }

}

}

using System.ComponentModel.DataAnnotations;

using System;

namespace WinFormsLibrary1.Entity

{

public class Melanj

{

[Key]

[Required]

public int Ntari { get; set; }

[Required]

public string TypeMelanga { get; set; }

[Required]

public DateTime DateRosliva { get; set; }

public int? CehPrMID { get; set; }

}

}

using System.ComponentModel.DataAnnotations;

using System;

namespace WinFormsLibrary1.Entity

{

public class OtbrakovkaEggs

{

[Key] // Первичный ключ

public int ID { get; set; }

[Required]

public int kolvo { get; set; }

public bool FreeOrNotFree { get; set; }

public DateTime dateTime { get; set; }

public int? CehSortID3 { get; set; }

public int? CehPrMID2 { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Text;

namespace WinFormsLibrary1.Entity

{

public class PartiyaEggs

{

[Key]

public int ID { get; set; }

public int Kolvo { get; set; }

public DateTime DateForm { get; set; }

public bool FreeOrNotFree { get; set; }

public int? CoPrID2 { get; set; }

public int? CehSortID { get; set; }

}

}

using System.ComponentModel.DataAnnotations;

using System;

namespace WinFormsLibrary1.Entity

{

public class PartiyaEggsRodClass

{

[Key]

public int ID { get; set; }

public int Kolvo { get; set; }

public DateTime DatePostEggs { get; set; }

public bool FreeOrNotFree { get; set; }

public int? RepID { get; set; }

public int? IncID2 { get; set; }

//public Reproductor Reproductor { get; set; }

public Incubator Incubator { get; set; }

}

}

using System.ComponentModel.DataAnnotations;

using System;

namespace WinFormsLibrary1.Entity

{

public class PartiyaMolodnyaka

{

[Key]

public int ID { get; set; }

public DateTime DataForm { get; set; }

public bool FreeOrNotFree { get; set; }

[Required]

public string TypeChicken { get; set; }

public int Kolvo { get; set; }

public int? IncID { get; set; }

public int? PtID { get; set; }

}

}

using System.ComponentModel.DataAnnotations;

using System;

namespace WinFormsLibrary1.Entity

{

public class PartiyaTovarnEggs

{

[Key]

public int ID { get; set; }

[Required]

public int Categori { get; set; }

[Required]

public DateTime DateUp { get; set; }

public int Kolvo { get; set; }

public int? CehSortID2 { get; set; }

}

}

using System.ComponentModel.DataAnnotations;

using System;

namespace WinFormsLibrary1.Entity

{

public class PartiyaVzrosloyChicken

{

[Key]

public int ID { get; set; }

public string TypeChiсken { get; set; }

public int Kolvo { get; set; }

public DateTime DateForm { get; set; }

public bool FreeOrNotFree { get; set; }

public int? PtID2 { get; set; }

public int? CoPrID { get; set; }

public int? UPKID { get; set; }

}

}

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System;

namespace WinFormsLibrary1.Entity

{

public class Ptichnic

{

[Key]

public int ID { get; set; }

public int Kolvo { get; set; }

[Required]

public string TypeChicken { get; set; }

[Required]

public DateTime DatePost { get; set; }

[Required]

public DateTime DateGrow { get; set; }

[Required]

public bool FreeOrNotFree { get; set; }

[ForeignKey("PtID")]

public ICollection<PartiyaMolodnyaka> PartiyaMolodnyakas { get; set; }

[ForeignKey("PtID2")]

public ICollection<PartiyaVzrosloyChicken> PartiyaVzrosloyChickens { get; set; }

public void TimerGrow() { }

public void FormPartii() { }

public void VipuscPartii() { }

}

}

using System.ComponentModel.DataAnnotations;

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

namespace WinFormsLibrary1.Entity

{

public class Reproductor

{

[Key]

public int RepID { get; set; }

public int KolvoB { get; set; }

public int KolvoN { get; set; }

public int maxB { get; set; }

public int maxN { get; set; }

[ForeignKey("RepID")]

public ICollection<PartiyaEggsRodClass> PartiyaEggsRodClasses { get; set; }

}

}

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System;

namespace WinFormsLibrary1.Entity

{

public class UPK

{

[Key]

public int ID { get; set; }

public int KolvoB { get; set; }

public bool FreeOrNotFree { get; set; }

public DateTime Dateform { get; set; }

[ForeignKey("UPKID")]

public ICollection<PartiyaVzrosloyChicken> PartiyaVzrosloyChickens { get; set; }

[ForeignKey("UPKID2")]

public ICollection<Fabrikat> Fabrikats { get; set; }

}

}

**Приложение Б**

(обязательное)

**Класс вспомогательных методов**

using Npgsql;

using System;

using System.Data;

using System.Windows.Forms;

using WinFormsLibrary1;

using WinFormsLibrary1.Entity;

using ApplicationContext = WinFormsLibrary1.ApplicationContext;

using System.Collections.Generic;

using System.Linq;

namespace WinFormsLibrary1

{

public class Logical

{

public void ReproductorLog(int a, int b, int c, DateTime date) // Загрузка данных в репродуктор и создание родительских партий

{

ApplicationContext context = new ApplicationContext();

Reproductor reproductor = context.Reproductor.FirstOrDefault();

reproductor.KolvoB = a;

reproductor.KolvoN = b;

var repr = context.Reproductor.ToList();

if (c > 0)

{

for (int i = 0; i < c / 1000; i++)

{

PartiyaEggsRodClass partiyaEggsRodClass = new PartiyaEggsRodClass

{

Kolvo = 1000,

RepID = repr[repr.Count - 1].RepID,

DatePostEggs = date,

};

context.PartiyaEggsRodClass.Add(partiyaEggsRodClass);

context.SaveChanges();

}

if (c%1000 != 0)

{

PartiyaEggsRodClass partiyaEggsRodClass1 = new PartiyaEggsRodClass

{

Kolvo = c %1000,

RepID = repr[repr.Count - 1].RepID,

DatePostEggs = date

};

context.PartiyaEggsRodClass.Add(partiyaEggsRodClass1);

}

//PartiyaEggsRodClass partiyaEggsRodClass = context.PartiyaEggsRodClass.Where(h => h.ID == 1).FirstOrDefault(); //Удаление сущности из таблицы

//context.PartiyaEggsRodClass.Remove(partiyaEggsRodClass);

context.SaveChanges();

}

//context.Reproductor.Update();

}

public string ReprLoad(string a, string b)

{

ApplicationContext context = new ApplicationContext();

int k = 0;

k = a.IndexOf(";");

a = a.Substring(0, k);

k = b.IndexOf(";");

b = b.Substring(0, k);

int IDb = Convert.ToInt32(a);

int IDn = Convert.ToInt32(b);

PartiyaVzrosloyChicken partiyaVzrosloyChickenB = context.PartiyaVzrosloyChicken.Where(h => h.ID == IDb).FirstOrDefault();

PartiyaVzrosloyChicken partiyaVzrosloyChickenN = context.PartiyaVzrosloyChicken.Where(h => h.ID == IDn).FirstOrDefault();

Reproductor reproductor = context.Reproductor.FirstOrDefault();

int kolvob = reproductor.maxB;

int kolvon = reproductor.maxN;

if (reproductor.KolvoB < kolvob || reproductor.KolvoN < kolvon)

{

partiyaVzrosloyChickenB.Kolvo += -(kolvob - reproductor.KolvoB);

reproductor.KolvoB += (kolvob - reproductor.KolvoB);

partiyaVzrosloyChickenN.Kolvo += -(kolvon - reproductor.KolvoN);

reproductor.KolvoN += (kolvon - reproductor.KolvoN);

context.SaveChanges();

return "";

}

else

{

return "Проверьте корректность выбора";

}

}

public string IncubatorLoad(string a, string b) //Загрузка данных в инкубатор

{

ApplicationContext context = new ApplicationContext();

int k = 0;

k = a.IndexOf(";");

a = a.Substring(0, k);

k = b.IndexOf(";");

b = b.Substring(0, k);

int IDa = Convert.ToInt32(a);

int IDb = Convert.ToInt32(b);

PartiyaEggsRodClass partiyaEggsRodClass = context.PartiyaEggsRodClass.Where(h => h.ID == IDa).FirstOrDefault();

Incubator incubator = context.Incubator.Where(c => c.ID == IDb).FirstOrDefault();

if (incubator.FreeOrNotFree)

{

incubator.KolvoEggs = partiyaEggsRodClass.Kolvo;

incubator.DatePost = DateTime.Now;

incubator.DayOfBorn = partiyaEggsRodClass.DatePostEggs.AddDays(21);

incubator.FreeOrNotFree = false;

//partiyaEggsRodClass.Kolvo = 0;

partiyaEggsRodClass.FreeOrNotFree = false;

partiyaEggsRodClass.IncID2 = incubator.ID;

context.SaveChanges();

return "Готово!";

}

else { return "Инкубатор заполнен"; }

}

public string IncubatorMolod(string a, int b, int c) // Создание партии молодняка из вылупившихся яиц в инкубаторе

{

ApplicationContext context = new ApplicationContext();

int k = a.IndexOf(";");

a = a.Substring(0, k);

int IDb = Convert.ToInt32(a);

Incubator incubator = context.Incubator.Where(c => c.ID == IDb).FirstOrDefault();

if (DateTime.Compare(DateTime.Now, incubator.DayOfBorn) < 0)

{

PartiyaMolodnyaka partiyaMolodnyaka = new PartiyaMolodnyaka

{

TypeChicken = "Бройлер",

Kolvo = b,

IncID = incubator.ID,

DataForm = DateTime.Now,

};

PartiyaMolodnyaka partiyaMolodnyaka1 = new PartiyaMolodnyaka

{

TypeChicken = "Несушка",

Kolvo = c,

IncID = incubator.ID,

DataForm = DateTime.Now,

};

context.PartiyaMolodnyaka.AddRange(partiyaMolodnyaka, partiyaMolodnyaka1);

incubator.FreeOrNotFree = true;

incubator.DayOfBorn = DateTime.Now.AddYears(100);

incubator.KolvoEggs = 0;

context.SaveChanges();

return "Готово!";

}

else { return "Яйца не вылупились"; }

}

public string PtichnicLoad(string a, string b)

{

ApplicationContext context = new ApplicationContext();

int k = 0;

k = a.IndexOf(";");

a = a.Substring(0, k);

k = b.IndexOf(";");

b = b.Substring(0, k);

int IDa = Convert.ToInt32(a);

int IDb = Convert.ToInt32(b);

PartiyaMolodnyaka partiyaMolodnyaka = context.PartiyaMolodnyaka.Where(h => h.ID == IDa).FirstOrDefault();

Ptichnic ptichnic = context.Ptichnic.Where(c => c.ID == IDb).FirstOrDefault();

int days = 60;

if (partiyaMolodnyaka.TypeChicken == "Бройлер")

days = 20;

else

days = 60;

if (ptichnic.FreeOrNotFree)

{

ptichnic.Kolvo = partiyaMolodnyaka.Kolvo;

ptichnic.DatePost = DateTime.Now;

ptichnic.DateGrow = partiyaMolodnyaka.DataForm.AddDays(days);

ptichnic.FreeOrNotFree = false;

ptichnic.TypeChicken = partiyaMolodnyaka.TypeChicken;

//partiyaMolodnyaka.Kolvo = 0;

partiyaMolodnyaka.FreeOrNotFree = false;

partiyaMolodnyaka.PtID = ptichnic.ID;

context.SaveChanges();

return "Готово!";

}

else { return "Птичник заполнен"; }

}

public string PtichnicFormPart(string a) // Создание партии взрослой птицы из птенцов птичника

{

ApplicationContext context = new ApplicationContext();

int k = a.IndexOf(";");

a = a.Substring(0, k);

int IDb = Convert.ToInt32(a);

Ptichnic ptichnic = context.Ptichnic.Where(c => c.ID == IDb).FirstOrDefault();

if (DateTime.Compare(DateTime.Now, ptichnic.DateGrow) < 0 && ptichnic.Kolvo != 0)

{

PartiyaVzrosloyChicken partiyaVzrosloyChicken = new PartiyaVzrosloyChicken

{

TypeChiсken = ptichnic.TypeChicken,

Kolvo = ptichnic.Kolvo,

PtID2 = ptichnic.ID,

DateForm = DateTime.Now,

};

context.PartiyaVzrosloyChicken.Add(partiyaVzrosloyChicken);

ptichnic.FreeOrNotFree = true;

ptichnic.DateGrow = DateTime.Now.AddYears(100);

ptichnic.Kolvo = 0;

ptichnic.TypeChicken = "";

context.SaveChanges();

return "Готово!";

}

else { return "Птицы не выросли"; }

}

public string UPKLoad(string a)

{

ApplicationContext context = new ApplicationContext();

int k = 0;

k = a.IndexOf(";");

a = a.Substring(0, k);

int IDa = Convert.ToInt32(a);

PartiyaVzrosloyChicken partiyaVzrosloyChicken = context.PartiyaVzrosloyChicken.Where(h => h.ID == IDa).FirstOrDefault();

UPK upk = context.UPK.FirstOrDefault();

if (upk.FreeOrNotFree)

{

upk.KolvoB = partiyaVzrosloyChicken.Kolvo;

upk.FreeOrNotFree = false;

upk.Dateform = DateTime.Now;

partiyaVzrosloyChicken.UPKID = upk.ID;

partiyaVzrosloyChicken.FreeOrNotFree = false;

context.SaveChanges();

return "Готово!";

}

else { return "УПК заполнен"; }

}

public string UPKFormFabricat()

{

ApplicationContext context = new ApplicationContext();

UPK upk = context.UPK.FirstOrDefault();

if (upk.FreeOrNotFree == false)

{

for(int i = 0; i< upk.KolvoB; i++)

{

Fabrikat fabrikat = new Fabrikat

{

UPKID2 = upk.ID,

DateUp = DateTime.Now

};

context.Fabrikat.Add(fabrikat);

context.SaveChanges();

}

upk.KolvoB = 0;

upk.FreeOrNotFree = true;

context.SaveChanges();

return "Готово!";

}

else { return "Птиц нет"; }

}

public string KPELoad(string a, string b)

{

ApplicationContext context = new ApplicationContext();

int k = 0;

k = a.IndexOf(";");

a = a.Substring(0, k);

k = b.IndexOf(";");

b = b.Substring(0, k);

int IDa = Convert.ToInt32(a);

int IDb = Convert.ToInt32(b);

PartiyaVzrosloyChicken partiyaVzrosloyChicken = context.PartiyaVzrosloyChicken.Where(h => h.ID == IDa).FirstOrDefault();

ComplexProizvodstvaEggs KPE = context.ComplexProizvodstvaEggs.Where(h => h.ID == IDb).FirstOrDefault();

if (KPE.FreeOrNotFree)

{

KPE.KolvoN = partiyaVzrosloyChicken.Kolvo;

KPE.FreeOrNotFree = false;

KPE.DateForm = DateTime.Now;

KPE.Pfeed = true;

KPE.Pwater = true;

partiyaVzrosloyChicken.CoPrID = KPE.ID;

partiyaVzrosloyChicken.FreeOrNotFree = false;

context.SaveChanges();

return "Готово!";

}

else { return "КПЯ заполнен"; }

}

public string KPEFormEggs(int b, string a)

{

ApplicationContext context = new ApplicationContext();

int k = a.IndexOf(";");

a = a.Substring(0, k);

int IDb = Convert.ToInt32(a);

ComplexProizvodstvaEggs KPE = context.ComplexProizvodstvaEggs.Where(h => h.ID == IDb).FirstOrDefault();

if (KPE.FreeOrNotFree == true)

return "Птиц нет";

if(KPE.Cikl >= KPE.CiklMax && KPE.FreeOrNotFree == false)

{

KPE.KolvoN = 0;

KPE.FreeOrNotFree = true;

KPE.Pfeed = false;

KPE.Pwater = false;

KPE.Cikl = 0;

context.SaveChanges();

return "Курицы слишком стары и были отправлены на убой";

}

else

{

PartiyaEggs partiyaEggs = new PartiyaEggs

{

CoPrID2 = KPE.ID,

DateForm = DateTime.Now,

Kolvo = b,

FreeOrNotFree = true

};

KPE.Cikl++;

context.PartiyaEggs.Add(partiyaEggs);

context.SaveChanges();

return "Готово!";

}

}

public string CSEFormEgg(int b, string a, int[] category)

{

int c = 0;

ApplicationContext context = new ApplicationContext();

int k = a.IndexOf(";");

a = a.Substring(0, k);

int IDb = Convert.ToInt32(a);

PartiyaEggs partiyaEggs = context.PartiyaEggs.Where(h => h.ID == IDb).FirstOrDefault();

CehSortEggs cse = context.CehSortEggs.FirstOrDefault();

for (int i = 0; i < category.Length; i++) { c += category[i]; }

if (c > partiyaEggs.Kolvo)

return "Неточность введенных данных";

if (partiyaEggs.FreeOrNotFree == true)

{

partiyaEggs.CehSortID = cse.ID;

partiyaEggs.FreeOrNotFree = false;

context.SaveChanges();

for (int i = 0; i < category.Length; i++ )

{

PartiyaTovarnEggs partiyaTovarnEggs = new PartiyaTovarnEggs

{

CehSortID2 = cse.ID,

DateUp = partiyaEggs.DateForm,

Kolvo = category[i],

Categori = i,

};

context.PartiyaTovarnEggs.Add(partiyaTovarnEggs);

context.SaveChanges();

}

OtbrakovkaEggs otbrakovkaEggs = new OtbrakovkaEggs

{

CehSortID3 = cse.ID,

dateTime = partiyaEggs.DateForm,

kolvo = b,

};

context.OtbrakovkaEggs.Add(otbrakovkaEggs);

context.SaveChanges();

return "Готово";

}

else

{

return "Проверьте правильность выбранных данных";

}

}

public string CPMFormMelanj(int b, int j, string a)

{

ApplicationContext context = new ApplicationContext();

int k = a.IndexOf(";");

a = a.Substring(0, k);

int IDb = Convert.ToInt32(a);

OtbrakovkaEggs otbrakovkaEggs = context.OtbrakovkaEggs.Where(h => h.ID == IDb).FirstOrDefault();

CehProizvMelanja cpm = context.CehProizvMelanja.FirstOrDefault();

if (b+j > otbrakovkaEggs.kolvo)

return "Неточность введенных данных";

if (otbrakovkaEggs.FreeOrNotFree == true)

{

otbrakovkaEggs.CehPrMID2 = cpm.ID;

otbrakovkaEggs.FreeOrNotFree = false;

context.SaveChanges();

for (int i = 0; i < b; i++)

{

Melanj melanj = new Melanj

{

CehPrMID = cpm.ID,

DateRosliva = otbrakovkaEggs.dateTime,

TypeMelanga = "Белок",

};

context.Melanj.Add(melanj);

context.SaveChanges();

}

for (int i = 0; i < j; i++)

{

Melanj melanj = new Melanj

{

CehPrMID = cpm.ID,

DateRosliva = otbrakovkaEggs.dateTime,

TypeMelanga = "Желток",

};

context.Melanj.Add(melanj);

context.SaveChanges();

}

return "Готово";

}

else

{

return "Проверьте правильность выбранных данных";

}

}

}

}

**Приложение В**

(обязательное)

**Класс ApplicationСontext**

using Microsoft.EntityFrameworkCore;

using WinFormsLibrary1.Entity;

using System;

namespace WinFormsLibrary1

{

public class ApplicationContext : DbContext

{

public DbSet<CehProizvMelanja> CehProizvMelanja { get; set; }

public DbSet<CehSortEggs> CehSortEggs { get; set; }

public DbSet<Chicken> Chicken { get; set; }

public DbSet<ComplexProizvodstvaEggs> ComplexProizvodstvaEggs { get; set; }

public DbSet<Fabrikat> Fabrikat { get; set; }

public DbSet<Incubator> Incubator { get; set; }

public DbSet<Melanj> Melanj { get; set; }

public DbSet<OtbrakovkaEggs> OtbrakovkaEggs { get; set; }

public DbSet<PartiyaEggsRodClass> PartiyaEggsRodClass { get; set; }

public DbSet<PartiyaMolodnyaka> PartiyaMolodnyaka { get; set; }

public DbSet<PartiyaTovarnEggs> PartiyaTovarnEggs { get; set; }

public DbSet<PartiyaVzrosloyChicken> PartiyaVzrosloyChicken { get; set; }

public DbSet<PartiyaEggs> PartiyaEggs { get; set; }

public DbSet<Ptichnic> Ptichnic { get; set; }

public DbSet<Reproductor> Reproductor { get; set; }

public DbSet<UPK> UPK { get; set; }

public DbSet<LogPas> LogPas { get; set; } // Логины и пароли для входа в аккаунт

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

//modelBuilder.Entity<Broyler>().HasIndex(s => s.TypeChiсken).IsUnique(); //Работает и делает поля уникальными

// Значение по умолчанию, можно сделать и вызов функции но я хз как (getdate())

modelBuilder.Entity<Incubator>().Property(s => s.FreeOrNotFree).HasDefaultValue(true);

modelBuilder.Entity<PartiyaEggsRodClass>().Property(s => s.FreeOrNotFree).HasDefaultValue(true);

modelBuilder.Entity<Reproductor>().Property(s => s.KolvoB).HasDefaultValue(0);

modelBuilder.Entity<Reproductor>().Property(s => s.KolvoN).HasDefaultValue(0);

modelBuilder.Entity<Reproductor>().Property(s => s.maxB).HasDefaultValue(200);

modelBuilder.Entity<Reproductor>().Property(s => s.maxN).HasDefaultValue(1000);

modelBuilder.Entity<PartiyaMolodnyaka>().Property(s => s.FreeOrNotFree).HasDefaultValue(true);

modelBuilder.Entity<Ptichnic>().Property(s => s.FreeOrNotFree).HasDefaultValue(true);

modelBuilder.Entity<PartiyaVzrosloyChicken>().Property(s => s.FreeOrNotFree).HasDefaultValue(true);

modelBuilder.Entity<UPK>().Property(s => s.FreeOrNotFree).HasDefaultValue(true);

modelBuilder.Entity<ComplexProizvodstvaEggs>().Property(s => s.FreeOrNotFree).HasDefaultValue(true);

modelBuilder.Entity<ComplexProizvodstvaEggs>().Property(s => s.Pfeed).HasDefaultValue(false);

modelBuilder.Entity<ComplexProizvodstvaEggs>().Property(s => s.Pwater).HasDefaultValue(false);

modelBuilder.Entity<ComplexProizvodstvaEggs>().Property(s => s.Cikl).HasDefaultValue(0);

modelBuilder.Entity<ComplexProizvodstvaEggs>().Property(s => s.CiklMax).HasDefaultValue(10);

modelBuilder.Entity<PartiyaEggs>().Property(s => s.FreeOrNotFree).HasDefaultValue(true);

modelBuilder.Entity<OtbrakovkaEggs>().Property(s => s.FreeOrNotFree).HasDefaultValue(true);

}

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseNpgsql("Host=localhost;Port=5432;Database=Pticefabrica;Username=postgres;Password=1111");

}

}

}

**Приложение Д**

(обязательное)

**Классы генерирующие отчет**

using Npgsql;

using System;

using System.Data;

using System.Windows.Forms;

using WinFormsLibrary1;

using WinFormsLibrary1.Entity;

using ApplicationContext = WinFormsLibrary1.ApplicationContext;

using System.Collections.Generic;

using System.Linq;

namespace WinFormsLibrary1

{

public class Reporter

{

/\*\*/

public bool GetReport(int year, int kvartal)

{

DateTime startdate = DateTime.Now;

DateTime enddate = DateTime.Now;

if (kvartal == 1)

{

startdate = new DateTime(year, 1, 1);

enddate = new DateTime(year, 4, 1);

}

if (kvartal == 2)

{

startdate = new DateTime(year, 4, 1);

enddate = new DateTime(year, 7, 1);

}

if (kvartal == 3)

{

startdate = new DateTime(year, 7, 1);

enddate = new DateTime(year, 10, 1);

}

if (kvartal == 4)

{

startdate = new DateTime(year, 10, 1);

enddate = new DateTime(year + 1, 1, 1);

}

//string[][] rep = new string[][];

ApplicationContext context = new ApplicationContext();

List<Melanj> b = context.Melanj.Where(h => h.DateRosliva >= startdate && h.DateRosliva < enddate).ToList();

//string[][] rep = new string[b.Count][];

List<List<String>> report = new List<List<String>>();

foreach (var item in b)

{

List<String> vs = new List<String>();

vs.Add(item.Ntari.ToString());

vs.Add(item.TypeMelanga.ToString());

vs.Add(item.DateRosliva.ToString());

report.Add(vs);

}

List<string> namesColumn = new List<string> { "Номер", "Тип", "Дата разлива"};

new GenerationExel().Generate(report, $"Melanj\_Q{kvartal}", $"{year}", "Melanj", namesColumn);

namesColumn.Clear();

report.Clear();

List<Fabrikat> fab = context.Fabrikat.Where(h => h.DateUp >= startdate && h.DateUp < enddate).ToList();

foreach (var item in fab)

{

List<String> vs = new List<String>();

vs.Add(item.ID.ToString());

vs.Add(item.DateUp.ToString());

report.Add(vs);

}

namesColumn.Add("Номер");

namesColumn.Add("Дата");

new GenerationExel().Generate(report, $"Fabrikat\_Q{kvartal}", $"{year}", "Fabrikat", namesColumn);

namesColumn.Clear();

report.Clear();

List<PartiyaTovarnEggs> tov = context.PartiyaTovarnEggs.Where(h => h.DateUp >= startdate && h.DateUp < enddate).ToList();

foreach (var item in tov)

{

List<String> vs = new List<String>();

vs.Add(item.ID.ToString());

vs.Add(item.Kolvo.ToString());

vs.Add(item.DateUp.ToString());

vs.Add(item.Categori.ToString());

report.Add(vs);

}

namesColumn.Add("Номер");

namesColumn.Add("Количество");

namesColumn.Add("Дата");

namesColumn.Add("Категория");

new GenerationExel().Generate(report, $"TovarnEggs\_Q{kvartal}", $"{year}", $"Товарные яйца", namesColumn);

return true;

}

}

}

using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

using OfficeOpenXml;

using OfficeOpenXml.Drawing.Chart;

using OfficeOpenXml.Style;

using WinFormsLibrary1.Entity;

namespace WinFormsLibrary1

{

public class GenerationExel

{

/\*\*/

public bool Generate(List<List<String>> report, string ways, string dir, string reportname, List<String> colnames)

{

ExcelPackage package = new ExcelPackage();

var sheet = package.Workbook.Worksheets

.Add("Report");

sheet.Cells["B2"].Value = "Company:";

sheet.Cells[2, 3].Value = "Птицефабрика 1";

sheet.Cells["B3"].Value = "Location:";

sheet.Cells["C3"].Value = "пгт Зимний, ул.Снежная, д.1";

sheet.Cells["B4"].Value = "Sector:";

sheet.Cells["C4"].Value = "Производственный";

sheet.Cells["B5"].Value = "Report type";

sheet.Cells["C5"].Value = reportname;

sheet.Cells[8, 2].Value = "Total Count:";

for(int i = 0; i < colnames.Count; i++)

{

sheet.Cells[9, 2 + i].Value = colnames[i];

}

var row = 10;

var column = 2;

var columns = 0;

foreach (var item in report)

{

for(int i = 0; i < item.Count ;i++)

{

sheet.Cells[row, column + i].Value = item[i];

}

row++;

columns = item.Count;

}

//sheet.Cells[8, 3].Formula = $"=COUNT(B{10}:B{row})";

sheet.Cells[8, 3].Value = report.Count;

//sheet.Cells[1, 1, row, column + 2].AutoFitColumns();

sheet.Column(2).Width = 20;

sheet.Column(3).Width = 20;

sheet.Column(4).Width = 20;

sheet.Column(5).Width = 20;

sheet.Cells[9, 4, 9 + report.Count, 4].Style.Numberformat.Format = "yyyy.mm.dd hh:mm:ss";

sheet.Cells[9, 2, 9 + report.Count, 2].Style.Numberformat.Format = "### ### ### ##0";

sheet.Column(2).Style.HorizontalAlignment = ExcelHorizontalAlignment.Left;

sheet.Column(3).Style.HorizontalAlignment = ExcelHorizontalAlignment.Left;

sheet.Column(4).Style.HorizontalAlignment = ExcelHorizontalAlignment.Left;

//sheet.Cells[9, 3, 9 + report.Count, 3].Style.HorizontalAlignment = ExcelHorizontalAlignment.Center;

sheet.Cells[8, 2, 8, 4].Style.Font.Bold = true;

sheet.Cells["B2:C4"].Style.Font.Bold = true;

sheet.Cells[9, 2, 9 + report.Count, 1 + columns].Style.Border.BorderAround(ExcelBorderStyle.Double);

sheet.Cells[9, 2, 9, 1 + columns].Style.Border.Bottom.Style = ExcelBorderStyle.Thin;

sheet.Protection.IsProtected = true;

string path = $@"Y:\{dir}";

string way = $@"Y:\{dir}\{ways}.xlsx";

DirectoryInfo dirInfo = new DirectoryInfo(path);

FileInfo fi = new FileInfo(way);

if (!dirInfo.Exists)

{

dirInfo.Create();

}

if (!fi.Exists)

{

package.SaveAs(fi);

return true;

}

else

return false;

}

}

}

Приложение Г

(справочное)

**Библиографический список**

1. Entity Framework [электронный ресурс] // обновлено 02.19.2020 URL: <https://docs.microsoft.com/en-us/aspnet/entity-framework/> (дата обращения 24.11.2021).
2. Entity Framework [электронный ресурс] // Википедия — свободной энциклопедия URL: <https://en.wikipedia.org/wiki/Entity_Framework/> (дата обращения 24.11.2021).
3. Руководство в Entity Framework [электронный ресурс] // Последнее обновление 31.10.2015 URL: <https://metanit.com/sharp/efcore/> (дата обращения 24.11.2021).
4. Управление схемой БД и миграции [электронный ресурс] // Последнее обновление 24.11.2021 URL: <https://metanit.com/sharp/efcore/2.2.php> (дата обращения 02.12.2021).
5. Отношения между классами и объектами [электронный ресурс] // Последнее обновление 31.10.2015 URL: <https://metanit.com/sharp/patterns/1.2.php/> (дата обращения 24.11.2021).