**Проектно-конструкторская часть**

1. **Разработка структуры приложения.**

Приложение будет состоять из двух форм основного функционала: формы работника теплицы и формы администратора системы, а также дополнительных форм для ввода новой информации в систему. Работник теплицы может осуществлять контроль над текущими параметрами теплицы, такими как: температура воздуха, освещенность, влажность воздуха и влажность почвы. Помимо этого, ему доступен перечень растений, культивируемых в его теплице месте с их стадиями развития, а также технологические карты каждого растения, если таковые имеются. Взаимодействуя со списком растений, работник может не только изменить стадию развития того или иного растения, но и представить системе отчет о собранном урожае. Более того, работник имеет возможность добавлять в систему информацию о новых высаживаемых растениях. Все эти процессы выполняются с помощью взаимодействия со служебными формами, результат записывается в базу данных для анализа производительности теплицы.

Помимо отображения текущих и желаемых параметров окружающей среды на форме, система будет в автоматическом режиме формировать отчет об ошибке, если какой-либо из параметров выходит за пределы допустимых значений для того или иного растения. Хранение отчетов о таких ошибках реализовано в базе данных с целью последующей обработки и формирования статистических данных о корреляции условий выращивания и собираемом урожае, а также о нормальном функционировании систем теплицы в целом.

В процессе взаимодействия с формой, работник теплицы может ознакомиться с текущими задачами по поддержанию стабильности работы теплицы. Среди таких задач могут присутствовать как регулярные, хранящиеся в системе, так и назначенные конкретно для каждой из теплиц в случае возникновения непредвиденных ситуации или изменений в работе системы. Работник может отметить задачу как выполненную с занесением результата работ в базу данных для последующего ознакомления администратора.

Второй основной функционирующий компонент системы – это форма администратора комплекса, где он может ознакомиться с количеством собранного урожая за отчетный период, а также с перечнем происходивших в системе ошибок. Взаимодействие с этой формой позволит ему также обновлять, изменять и формировать новые задачи, что служит поддерживанию долговечности и целостности системы во времени. Это осуществляется с помощью второстепенных форм для ввода в систему новой информации. Также администратор способен добавлять в систему информацию о новых культурах для выращивания, а также записывать для них технологические карты.

Таким образом, основными потребителями информации, поступающей из базы данных, являются работники теплиц и администраторы комплекса. Они воспринимают информацию о выращиваемых растениях, текущих задачах, урожайности и произошедших ошибках, и в то же время являются источником информации для базы данных, так как владеют наиболее актуальными данными.

Система разработана в среде программирования MS Visual Studio 2015, в качестве системы управления базами данных используется PostgreSQL. Примерная последовательность разработки системы:

* Формирование физической схемы данных и создание надлежащей структуры базы данных;
* Реализация тестового наполнения базы данных;
* Разработка прототипа интерфейса формы работника теплицы;
* Реализация подключения интерфейса к СУБД с помощью введения уровня бизнес-логики приложения по паттерну MVVM;
* Разработка алгоритмов чтения данных из СУБД;
* Внесение необходимых правок, разработка дополнительных форм для ввода информации;
* Реализация механизма входа в приложение для различных типов работников;
* Разработка формы администратора и всех вспомогательных форм, обеспечивающих ее работу.
* Тестирование и отладка приложения.

1. **Разработка алгоритмов обработки информации**.

В систему входит алгоритм проверки текущего состояния параметра среды и его соответствия параметрам произрастающих в теплице растений. Хранение информации о необходимых параметрах реализовано в виде технологических карт, каждая из которых хранит собственно название параметра, его оптимальное значение, допустимые отклонения от оптимального значения, при соблюдении которых условия выращивания близки к идеальным, а также предельные отклонения, выход за границы которых грозит тяжелыми последствиями.

Во время поступления информации с датчиков значения контролируемых параметров изменяются, что влечет за собой вызов события об изменении значения соответствующего свойства зависимости. В методе, подписанном на это событие, новое значение параметра проверяется на соответствие оптимальным параметрам всех растений в теплице. Результат проверки записывается в базу данных. При этом необходимо помнить о том, что большинство показателей (исключая температуру), вычисляются в процентах. Поэтому необходимо следить, чтобы не возникало ситуации, когда значение параметра выходит за пределы 100%, что говорит прежде всего о неисправности датчика.

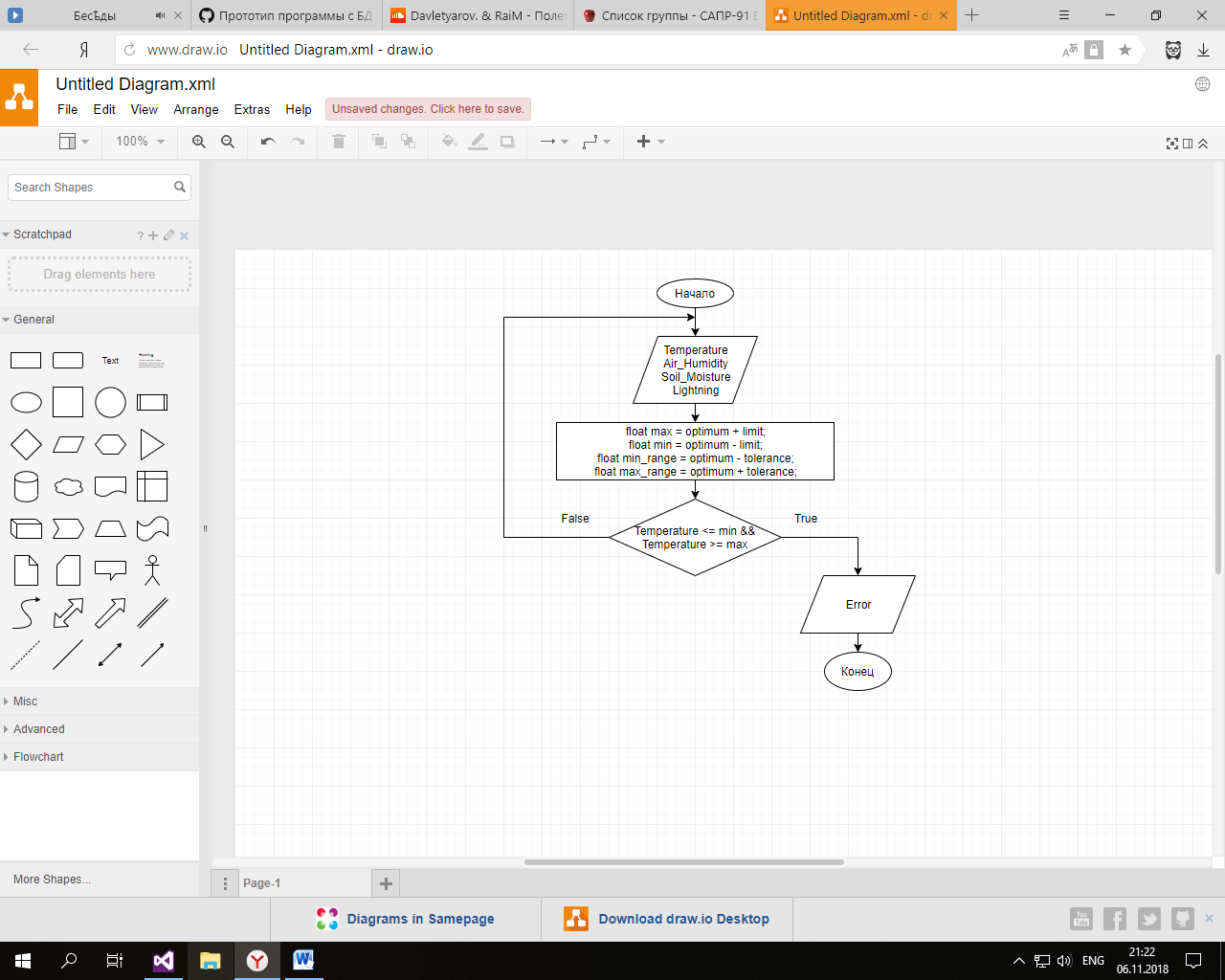


Рисунок 1 – Графическое преставление алгоритма обработки информации, поступающей с датчиков

1. **Логическая схема базы данных**

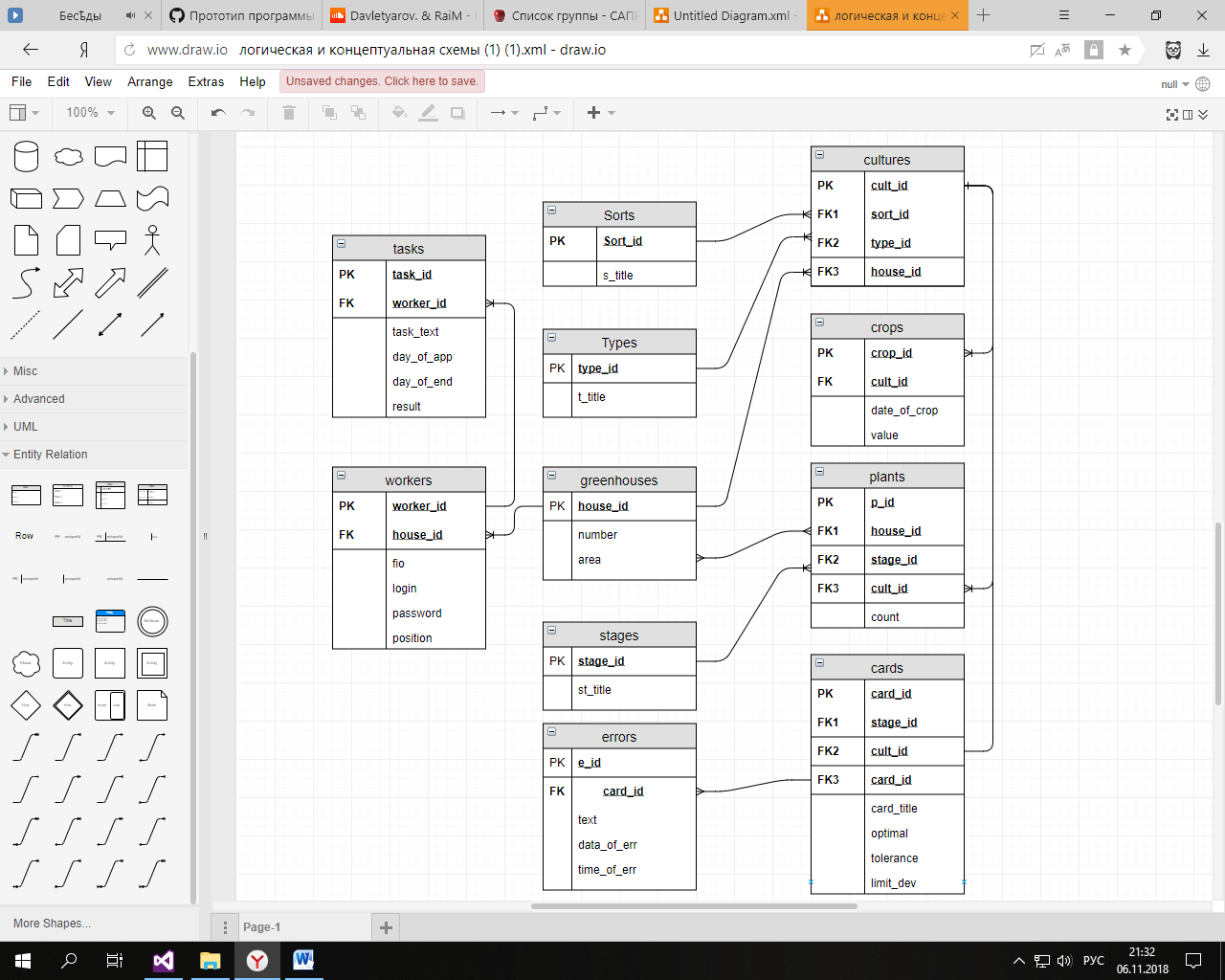


Рисунок 2 – Логическая схема базы данных

Логическая схема проектируемой системы состоит из 12 сущностей, среди которых:

* Sorts – сущность-справочник, хранящий значения сортов выращиваемых культур;
* Types – сущность-справочник, хранящий значения видов выращиваемых культур;
* Stages – сущность-справочник, хранящий перечень стадий, которые проходит растение в ходе своего выращивания;
* Greenhouses – сущность, хранящая информацию о теплицах, входящих в комплекс, их номер и площадь;
* Cultures – сущность, объединяющая значения вышеописанных справочников, чтобы получить информацию о культивируемой культуре и теплицах, в которых она выращивается;
* Plants – сущность, хранящая информацию о произрастающих в данный момент в теплице растений. Удаление отмерших растений из базы данных не происходит, на уровне интерфейса пользователя таким растениям присваивается максимальное значение стадии развития, после чего растение на форме не показывается;
* Cards – сущность, хранящая технологические карты для растений, включает название параметра, его оптимальное значение, допуск и предельное отклонение;
* Errors – сущность, хранящая отчеты о возникающих в системе ошибках при отклонении параметров от их оптимальных значений. Включает дату и время возникновения ошибки, текст, представляющий собой автоматически генерируемую строку на основе той технологической карты, несоответствие которой и было причиной возникновения ошибки;
* Tasks – сущность, хранящая задачи для работников. Включает информацию о дате появления задачи, непосредственно содержание задачи, дату выполнения задачи, а также результат выполнения задачи;
* Workers – сущность, хранящая информацию о работниках теплиц, их персональных данных, занимаемой должности, а также логине и пароля для входа в систему;
* Crops – сущность, хранящая информацию о событии сбора урожая конкретной культуры. Включает поля даты сбора урожая, а также его количество.

Таким образом, логическая схема базы данных показывает, что описываемое ей отношения находятся в третьей нормальной форме, так как ни один ключевой атрибут не находится в транзитивной функциональной зависимости от потенциального ключа.

1. **Разработка архитектуры приложения.**

Разработка приложения производится с помощью среды программирования MS Visual Studio 2015 на языке программирования C# и средства разработки интерфейса пользователя Windows Presentation Foundation. Хранение данных и управление ими производится с помощью системы управления базами данных PostgreSQL. Связь между уровнями бизнес-логики приложения реализована с помощью NuGet-пакета Npgsql, и предоставляемыми инструментами, такими как DataReader и DataAdapter. Приложение реализует паттерн проектирования MVVM и многослойную архитектуру приложения.

В основе WPF лежит векторная система визуализации, не зависящая от разрешения устройства вывода и созданная с учётом возможностей современного графического оборудования. WPF предоставляет средства для создания визуального интерфейса, включая язык XAML (eXtensible Application Markup Language), элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление.[MSDN]

XAML представляет собой язык декларативного описания интерфейса, основанный на XML. Также реализована модель разделения кода и дизайна, позволяющая разделить реализацию графического интерфейса и уровня бизнес-логики приложения. Кроме того, есть встроенная поддержка стилей элементов, а сами элементы легко разделить на элементы управления второго уровня, которые, в свою очередь, разделяются до уровня векторных фигур и свойств/действий.

Model-View-ViewModel (MVVM) — шаблон проектирования архитектуры приложения. Представлен в 2005 году Джоном Госсманом (John Gossman) как модификация шаблона Presentation Model. Шаблон MVVM делится на три части:

* *Модель* представляет собой логику работы с данными и описание фундаментальных данных, необходимых для работы приложения.
* *Представление* — графический интерфейс (окна, списки, кнопки и т. п.). Выступает подписчиком на событие изменения значений свойств или команд, предоставляемых Моделью Представления. В случае, если в Модели Представления изменилось какое-либо свойство, то она оповещает всех подписчиков об этом, и Представление, в свою очередь, запрашивает обновлённое значение свойства из Модели Представления. В случае, если пользователь воздействует на какой-либо элемент интерфейса, Представление вызывает соответствующую команду, предоставленную Моделью Представления.
* *Модель Представления* — с одной стороны, абстракция Представления, а с другой — обёртка данных из Модели, подлежащие связыванию. То есть, она содержит Модель, преобразованную к Представлению, а также команды, которыми может пользоваться Представление, чтобы влиять на Модель.

PostgreSQL - это объектно-реляционная система управления базами данных (СУБД), которая имеет традиционные возможности коммерческих СУБД с расширениями, которые есть в СУБД нового поколения. PostgreSQL - это свободное и полностью открытое программное обеспечение.