**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

Специальность 1-40 05 01-01 Информационные системы и технологии (в проектировании и производстве)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

по дисциплине «Объектно-ориентированное программирование»

на тему: «***WPF* ПРИЛОЖЕНИЕ ДЛЯ УЧЁТА ПОСТУПЛЕНИЯ ТОВАРОВ НА СКЛАДЫ ПРЕДПРИЯТИЯ**»

Исполнитель: студент гр. ИТП-22

Расшивалов Н.И.

Руководитель: доцент

Курочка К.С.

Дата проверки: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата допуска к защите: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подписи членов комиссии

по защите курсового проекта: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Гомель 2021

Установа адукацыі «Гомельскі дзяржаўны   
тэхнічны ўніверсітэт імя П.В. Сухога»

Факультэт аўтаматызаваных і інфармацыйных сістэм

Кафедра \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**РЭЦЭНЗІЯ**

**на курсавы праект (работу)**

па дысцыпліне \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ,

выканана студэнтам \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

групы \_\_\_\_\_\_\_\_\_\_

I. Пералік заўваг па тэксту курсавога праекта (работы)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

II. Агульная характарыстыка работы

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

СОДЕРЖАНИЕ

[Введение 2](#_Toc65968606)

[1 Обзор программго-технических средств для решения поставленной задачи 4](#_Toc65968607)

[1.1 Особенности языка программирования *С#* 4](#_Toc65968608)

[1.2 Особенности среды разработки 6](#_Toc65968609)

[1.3 Особенности объектно-реляционной СУБД 7](#_Toc65968610)

# **ВВЕДЕНИЕ**

Учет и совершенствование складских операций – залог качественной работы любого предприятия. Налаженная и правильная работа склада определяет, насколько рационально компания работает, как эффективно используются все необходимые производственные ресурсы. Улучшенные системы и автоматизированный бухгалтерский учет складских операций позволяют добиться лучшей рентабельности и производительности, вследствие чего, готовый продукт становится качественнее.

Складской учёт – один из основных компонентов как бухгалтерского, так и управленческого учёта. Независимо от того, какую финансово-хозяйственную деятельность осуществляет тот или иной субъект хозяйствования, ему не обойтись без складского учёта. Даже те предприятия, которые не ведут торговую либо производственную деятельность, имеют на балансе какие-либо активы, которые учитываются на складе (канцтовары, оргтехника, мебель, запасные части и т.д.). Поэтому грамотная постановка складского учёта является одним из основных условий успешной деятельности предприятия.

Компьютерный учет имеет свои особенности и радикально отличается от обычного (бумажного). Автоматизация системы ведения учета, хозяйства позволяет повысить эффективность работы, достичь возможностей, недоступных ранее при использовании «ручного» метода обработки документации. Желательно автоматизировать весь процесс деятельности предприятия, в этом случае, эффективность автоматизации будет более заметной.

Простейшим примером компьютерного учета служит приложение *microsoft* *exel*. Но эффективность его использования крайне мала в наше время, поскольку обработка информации занимает длительное время, отсутствие бизнес логики и целостности данных, сложности в работе нескольких пользователей, невозможность хранить большие объемы информации и ограниченный функционал.

Актуальность работы заключается в увеличении объема информации поступлении товаров на склады и их отгрузке, быстрый и точный подсчет данных.

Цель курсовой работы – создание программного комплекса для упрощения учета поступления товаров на склады предприятия.

Задачи курсовой работы заключаются в следующем – создание программного комплекса на языке программирования *С#* и СУБД *PostgreSQL* для учета поступления товаров на склады предприятия, а также написание блок схем и тестов для этого комплекса.

1. **ОБЗОР ПРОГРАММГО-ТЕХНИЧЕСКИХ СРЕДСТВ ДЛЯ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ**

## **Особенности языка программирования *С#***

*C#*– язык программирования, сочетающий объектно-ориентированные и аспектно-ориентированные концепции. Разработан в 1998–2001 годах группой инженеров под руководством Андерсa Хейлсбергa в компании *Microsoft* как основной язык разработки приложений для платформы *Microsoft* .*NET*. Компилятор с *C#* входит в стандартную установку самой .*NET*, поэтому программы на нём можно создавать и компилировать даже без инструментальных средств вроде *Visual* *Studio*.

*C*# относится к семье языков с *C*-подобным синтаксисом, из них его синтаксис наиболее близок к *С++* и *Java*. Язык имеет строгую статическую типизацию, поддерживает полиморфизм, перегрузку операторов, указатели на функции-члены классов, атрибуты, события, свойства, исключения, комментарии в формате *XML*. Переняв многое от своих предшественников – языков *С++,* *Java*, *Delphi*, Модула и *Smalltalk* ­­­­­– *С#*, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем: так, C# не поддерживает множественное наследование классов (в отличие от C++).

*C#* разрабатывался как язык программирования прикладного уровня для *CLR* и, как таковой, зависит, прежде всего, от возможностей самой *CLR*. Это касается, прежде всего, системы типов *C#*, которая отражает *FCL*. Присутствие или отсутствие тех или иных выразительных особенностей языка диктуется тем, может ли конкретная языковая особенность быть транслирована в соответствующие конструкции *CLR*. Так, с развитием *CLR* от версии 1.1 к 2.0 значительно обогатился и сам C#. Однако эта закономерность была нарушена с выходом *C#* 3.0, представляющим собой расширения языка, не опирающиеся на расширения платформы .*NET*. *CLR* предоставляет *C#*, как и всем другим .NET-ориентированным языкам, многие возможности, которых лишены «классические» языки программирования. Например, сборка мусора не реализована в самом *C#*, а производится *CLR* для программ, написанных на *C#* точно так же, как это делается для программ на *VB*.*NET*, *J#* и др.

Вот лишь несколько функций языка *C#*, которые позволяют создавать надежные и устойчивые приложения. Сборка мусора – автоматически освобождает память, занятую недоступными неиспользуемыми объектами. допускающие значения null, обеспечивают защиту от переменных, которые не ссылаются на выделенные объекты. Обработка исключений предоставляет структурированный и расширяемый подход к обнаружению ошибок и восстановлению после них. Лямбда-выражения поддерживают приемы функционального программирования. Синтаксис LINQ создает общий шаблон для работы с данными из любого источника. Поддержка языков для асинхронных операций предоставляет синтаксис для создания распределенных систем. В *C#* действует – единая система типов. Все типы *C#*, включая типы-примитивы, такие как *int* и *double*, наследуют от одного корневого типа *object*. Все типы используют общий набор операций, а значения любого типа можно хранить, передавать и обрабатывать схожим образом. Более того, *C#* поддерживает как определяемые пользователями [ссылочные типы](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/reference-types), так и [типы значений](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/value-types). *C#* позволяет динамически выделять объекты и хранить упрощенные структуры в стеке. *C#* поддерживает универсальные методы и типы, обеспечивающие повышенную безопасность типов и производительность. *C#* предоставляет итераторы, которые позволяют разработчикам классов коллекций определять пользовательские варианты поведения для клиентского кода.

В *C#* особое внимание уделяется **управлению версиями** для обеспечения совместимости программ и библиотек при их изменении. Вопросы управления версиями существенно повлияли на такие аспекты разработки *C#*, как раздельные модификаторы *virtual* и *override*, правила разрешения перегрузки методов и поддержка явного объявления членов интерфейса.

## **Особенности среды разработки**

***Visual Studio Code*** – это сервис, который позиционируется как «легкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений.

***Visual Studio Code*** позволяет разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии *Windows* *Forms*, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ.

В редакторе присутствуют встроенный отладчик, инструменты для работы с *Git* и средства рефакторинга, навигации по коду, автодополнения типовых конструкций и контекстной подсказки.

Продукт поддерживает разработку для платформ *ASP*.*NET* и *Node*.*js*, и считается легковесным решение, которое позволяет обойтись без полной интегрированной среды разработки.

Большим плюсом редактора является поддержка большого количества языков, таких как *C++*, *C#*, *Python*, *PHP*, *JavaScript* и других.

Возможности ***Visual Studio Code:***

* встроенные инструменты интеграции с *GitHub*, *GIT*, а также *Visual* *Studio* *Team* *Services* для быстрого тестирования, сборки, упаковки и развертывания разных типов приложений;
* удобство работы с *Unity*-проектами;
* публикация созданных приложений в *Microsoft* *Azure* через сервис *Visual* *Studio* *Team* *Services*;
* поддержка практически всех языков программирования;
* написание кода для конкретной задачи с его последующей интеграцией в проект (с надстройкой или напрямую);
* обширная библиотека шаблонов, готовых фрагментов кода и сниппетов с возможностью добавления своих элементов;
* одновременная работа с несколькими проектами (в нескольких окнах);
* интерфейс можно разделить на две панели для сравнения кода;
* функция отладки.

Преимущества ***Visual* *Studio* *Code*:**

* множество настроек (как всей программы, так и интерфейса);
* расширяемая библиотека дополнений и готовых решений;
* мультифункциональность (редактор поддерживает почти все языки, используемые для создания приложений);
* простота и гибкость.

## **Особенности объектно-реляционной СУБД**

*PostgreSQL* – объектно-реляционная система управления базами данных

*PostgreSQL* – это популярная свободная объектно-реляционная система управления базами данных. *PostgreSQL* базируется на языке *SQL* и поддерживает многочисленные возможности.

**Преимущества *PostgreSQL*:**

* поддержка БД неограниченного размера;
* мощные и надёжные механизмы транзакций и репликации;
* расширяемая система встроенных языков программирования и поддержка загрузки *C*-совместимых модулей;
* наследование;
* легкая расширяемость.

**Текущие ограничения *PostgreSQL*:**

* Нет ограничений на максимальный размер базы данных
* Нет ограничений на количество записей в таблице
* Нет ограничений на количество индексов в таблице
* Максимальный размер таблицы — 32 Тбайт
* Максимальный размер записи — 1,6 Тбайт
* Максимальный размер поля — 1 Гбайт
* Максимум полей в записи 250—1600 (в зависимости от типов полей)

**Функции** в *PostgreSQL* являются блоками кода, исполняемыми на сервере, а не на клиенте БД. Хотя они могут писаться на чистом *SQL*, реализация дополнительной логики, например, условных переходов и циклов, выходит за рамки собственно *SQL* и требует использования некоторых языковых расширений. Функции могут писаться с использованием различных языков программирования. *PostgreSQL* допускает использование функций, возвращающих набор записей, который далее можно использовать так же, как и результат выполнения обычного запроса. Функции могут выполняться как с правами их создателя, так и с правами текущего пользователя. Иногда функции отождествляются с хранимыми процедурами, однако между этими понятиями есть различие.

**Триггеры** в *PostgreSQL* определяются как функции, инициируемые *DML*-операциями. Например, операция *INSERT* может запускать триггер, проверяющий добавленную запись на соответствия определённым условиям. При написании функций для триггеров могут использоваться различные языки программирования. Триггеры ассоциируются с таблицами. Множественные триггеры выполняются в алфавитном порядке.

**Механизм правил** в *PostgreSQL* представляет собой механизм создания пользовательских обработчиков не только *DML*-операций, но и операции выборки. Основное отличие от механизма триггеров заключается в том, что правила срабатывают на этапе разбора запроса, до выбора оптимального плана выполнения и самого процесса выполнения. Правила позволяют переопределять поведение системы при выполнении *SQL*-операции к таблице.

**Индексы** в *PostgreSQL* следующих типов: *B*-дерево, хэш, *R*-дерево, *GiST*, *GIN*. При необходимости можно создавать новые типы индексов, хотя это далеко не тривиальный процесс.

**Многоверсионность** поддерживается в *PostgreSQL* – возможна одновременная модификация БД несколькими пользователями с помощью механизма *Multiversion* *Concurrency* *Control* (*MVCC*). Благодаря этому соблюдаются требования *ACID*, и практически отпадает нужда в блокировках чтения.

**Расширение** *PostgreSQL* для собственных нужд возможно практически в любом аспекте. Есть возможность добавлять собственные преобразования типов, типы данных, домены (пользовательские типы с изначально наложенными ограничениями), функции (включая агрегатные), индексы, операторы (включая переопределение уже существующих) и процедурные языки.

**Наследование** в *PostgreSQL* реализовано на уровне таблиц. Таблицы могут наследовать характеристики и наборы полей от других таблиц (родительских). При этом данные, добавленные в порождённую таблицу, автоматически будут участвовать (если это не указано отдельно) в запросах к родительской таблице.

В разработке простых сайтов *PostgreSQL* используется несколько реже, чем *[MySQL](https://web-creator.ru/articles/mysql)* / *MariaDB*, но всё же эта пара с заметным отрывом опережает по частоте использования остальные [системы управления базами данных](https://web-creator.ru/articles/about_databases). При этом в разработке сложных сайтов и веб-приложений *PostgreSQL* опережает по использованию *MySQL* и *MariaDB*. Большинство [фреймворков](https://web-creator.ru/articles/about_frameworks) (например, *[Ruby](https://web-creator.ru/articles/about_ruby_on_rails)**[on](https://web-creator.ru/articles/about_ruby_on_rails)**[Rails](https://web-creator.ru/articles/about_ruby_on_rails)*, *[Yii](https://web-creator.ru/articles/yii)*, *[Symfony](https://web-creator.ru/articles/symfony)*, *[Django](https://web-creator.ru/articles/django)*) поддерживают использование *PostgreSQL* в разработке.

У *PostgreSQL* множество возможностей. Созданный с использованием объектно-реляционной модели, он поддерживает сложные структуры и широкий спектр встроенных и определяемых пользователем типов данных. Он обеспечивает расширенную ёмкость данных и заслужил доверие бережным отношением к целостности данных.

Сравнение популярнейших СУБД Таблица 1.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СУБД | Тип | Операционные системы | Лицензия | Исходный код | Поддержка |
| Oracle Database | Мульти-модельная | Linux, Microsoft Windows, Oracle Solaris, IBM AIX, HP-UX | Коммерческая | Закрытый | Платная |
| MySQL | Реляционная | Linux, Microsoft Windows, Oracle Solaris, macOS, FreeBSD | GNU GPL и коммерческая | Открытый | Платная |
| SQL Server | Реляционная | Linux, Microsoft Windows | Коммерческая | Закрытый | Бесплатная |
| PostgreSQL | Объектно-реляционная | Linux, Microsoft Windows, Oracle Solaris, IBM AIX, macOS, HP-UX, QNX | Открытое ПО, разрешительная лицензия | Открытый | Платная |