**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**

Для эффективной работы любого из предприятий важен учет и модернизация операций, производимых на складе. Автоматизация складских операций позволяет добиться лучшей производительности. Правильная и отлаженная работа склада определяет, насколько эффективно работает предприятие, как рационально используются все необходимые ресурсы производства. Благодаря этому, готовый продукт становится намного качественнее.

Учёт товаров на складе является одним из главных компонентов в ведении торговой деятельности предприятия и не только. Компании, которые не осуществляют торговой или производственной деятельности, имеют на складе какие-либо активы. Их так же нужно учитывать.

Компьютерный учет имеет свои особенности и преимущества перед бумажным. Автоматизация системы ведения учета, позволяет повысить эффективность работы, достичь возможностей, недоступных ранее при использовании «ручного» метода обработки документации.

Простейшим примером компьютерного учета служит приложение *Microsoft Excel.* 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#***

Язык *C#* – это современный объектно-ориентированный язык программирования общего назначения, разработанный *Microsoft* по инициативе Андреса Хейлсберга.

Язык программирования *C#* очень простой для изучения. Он полностью основан на языках *C* и *C++.*

Однако синтаксис языка *C#* очень выразителен, но при этом он прост и легок в освоении.

*C#* упрощает многие из сложностей *C++* и предоставляет мощные функции, такие как типы значений, допускающие значение *null*, делегаты, перечисления, лямбда-выражения и прямой доступ к памяти, которых нет, например, в *Java*.

Раньше *Microsoft* создавала этот язык только для приложений *Windows*, но после этого, этот язык начал использоваться для консоли, *Android* и *iOS*, кроме того, *C#* стал использоваться с программным обеспечением для машинного обучения.

Основное преимущество языка *C#* в том, что приложение, написанное на *C#*, может быть развернуто в любой операционной системе, например, *Android* или *iOS*, *Windows* или облачной платформе. Это называется кроссплатформенностью.

В языке *C#* есть много полезных функций, которые делают его более удобным и уникальным по сравнению с другими языками.

Язык очень быстрый, его компиляция и выполнение слишком малы.

*C#* – простой язык. Это дает структурированный подход к разбивке проблемы на части. Кроме того, он имеет богатый набор библиотечных функций и типов данных.

Язык *C#* – это объектно-ориентированный язык программирования. Это так же упрощает разработку и сопровождение по сравнению с процедурно-ориентированным языком программирования.

Кроме того, программирование на *C#* поддерживает инкапсуляцию данных, наследование, полиморфизм, интерфейсы.

*C#* – это безопасный код, который может обращаться только к области памяти и имеет разрешение на выполнение. Следовательно, это повышает безопасность программы.

В языке *C#* невозможно выполнять небезопасные преобразования, такие как преобразование *double* в логическое значение. Его типы значений (примитивные типы) инициализируются нулями, а ссылочные типы (объекты и классы) автоматически инициализируются компилятором значением *null*.

Совместимость – это процесс, который позволяет программам на *C#* делать практически все, что может делать собственное приложение *C++*. Языковая совместимость – это способность кода взаимодействовать с кодом, написанным с использованием другого языка программирования. Это может помочь максимально увеличить повторное использование кода и повысить эффективность процесса разработки.

Язык *C#* обеспечивает поддержку использования *COM*-объектов, независимо от того, на каком языке они были созданы. Однако он также поддерживает специальную функцию, которая позволяет программе вызывать любой собственный *API*.

*C#* – это компьютеризированный масштабируемый язык программирования с возможностью обновления.

В программировании на *C#* установлена ​​очень эффективная система, которая автоматически собирает и стирает мусор, присутствующий в системе. Языкочень эффективен в управлении системой, потому что он не создает беспорядка в системе, и система не зависает во время выполнения.

Приложение, написанное на .*NET*, будет иметь лучшую интеграцию и интерпретируемость по сравнению с другими *NET*-технологиями. Программирование на *C#* выполняется в среде *CLR*, что упрощает интеграцию с компонентами, написанными на других языках.

Стоимость обслуживания меньше и безопаснее в использовании по сравнению с другими языками. На языке *C#* можно разрабатывать собственные приложения для *iOS*, *Android* и *Windows* *Phone* с помощью платформы *Xamarin*.

Вот некоторые преимущества языка *C#* над языком *Java*, которые делают его более полезным.

Поскольку язык *C#* является языком .*NET*, он поддерживает взаимодействие языков. Он может получить доступ к коду, написанному на любом языке, совместимом с .*NET*, а также наследовать классы, написанные на этих языках. Но на языке *Java* это невозможно.

Переносимый исполняемый файл языка *C#* может содержать любое количество классов, в то время как файл .*class* на языке *Java* содержит только один класс.

Язык *C#* содержит такие функции, как «Свойства» и «Индексаторы», хотя они недоступны в языке *Java*.

*C#* поддерживает перегрузку оператора, структуры и директивы препроцессора, тогда как язык *Java* не имеет их.

В программировании на C # пространства имен не связаны с каталогами, в то время как пакеты в программировании на *Java* напрямую связаны с именами каталогов.

Код *C#* при компиляции генерирует файл «.*exe*» или «.*dll*», который также называется переносимым исполняемым файлом. Эти файлы содержат код *Microsoft* *Intermediate* *Language* (*MSIL*). В то время как код *Java* при компиляции генерирует файл .*class*, который содержит байт-код.

В программировании на *C#* методы не являются виртуальными по умолчанию, тогда как в программировании на *Java* методы являются виртуальными по умолчанию, что снижает производительность.

*C#* полностью основан на платформе *Microsoft* .*Net*, поэтому этот язык не является гибким.

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, авто-дополнения 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 |