МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«Гомельский государственный технический университет имени П.О.Сухого»

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

направление специальности 1-40 05 01-01 Информационные системы и технологии (в проектировании и производства)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА** к курсовому проекту

по дисциплине «Объектно-ориентированное программирование»

на тему: «*Windows form* приложение для системы распространения билетов в театре»

Исполнитель: студент группы ИТП-21

Бондарев Е.Ю.

Руководитель: преподаватель Курочка К.С.

Дата проверки: *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* Дата допуска к защите: *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* Дата защиты: *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

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

Подписи членов комиссии

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

Гомель 2023

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc130337682)

[1 Технические подходы к реализации приложения для системы продажи билетов 5](#_Toc130337683)

[1.1 Обзор аналогов 5](#_Toc130337684)

[1.2 Объектно-ориентированное программирование 6](#_Toc130337685)

[1.3 Принципы *SOLID* 7](#_Toc130337686)

[1.4 Модели хранения данных 8](#_Toc130337687)

[1.5 *Windows Forms* технология 9](#_Toc130337688)

[1.6 Язык разметки документов *XML* 9](#_Toc130337689)

# ВВЕДЕНИЕ

Система продажи билетов имеет важное значение для многих организаций, включая кинотеатры, театры, музеи, концертные залы и другие. Она облегчает процесс покупки билетов для клиентов и обеспечивает безопасный и удобный способ продажи билетов для организаторов мероприятий.

Эта система также может быть использована для контроля доступа на мероприятия, что обеспечивает безопасность и предотвращает мошенничество. Кроме того, система продажи билетов может предоставлять информацию о доступности мест, ценах на билеты, расписании и других деталях мероприятия. Это помогает клиентам выбрать подходящее время и место для посещения мероприятия и обеспечивает гладкое и эффективное проведение самого мероприятия.

Система продажи билетов может также предоставлять дополнительные услуги, такие как онлайн-бронирование мест и выбор места, возможность оплаты билетов с помощью различных способов, например, кредитных карт и мобильных приложений. Это удобно для клиентов и помогает повысить уровень сервиса.

Существует множество различных систем продажи билетов, включая онлайн-платформы, автоматизированные кассы и системы, управляемые операторами. Каждая система имеет свои преимущества и недостатки, и организации выбирают систему, которая наиболее соответствует их потребностям и бюджету.

Одним из главных преимуществ систем продажи билетов является их доступность для клиентов. Клиенты могут легко купить билеты, не выходя из дома, используя онлайн-платформы или мобильные приложения. Это особенно удобно для тех, кто не имеет времени или возможности посетить кассу в физическом месте продажи билетов.

Благодаря системам продажи билетов организации могут лучше контролировать количество проданных билетов и заранее знать количество зрителей на мероприятии.

Более того, система продажи билетов может помочь в сборе данных и аналитике. Организаторы мероприятий могут использовать данные о продажах билетов для анализа поведения клиентов и популярности определенных мероприятий. Это может помочь им в принятии решений относительно маркетинговых стратегий и улучшения опыта посетителей в будущем.

Некоторые системы продажи билетов также могут предоставлять возможность организаторам мероприятий продавать дополнительные товары и услуги, такие как сувениры или билеты на другие мероприятия, что может увеличить доходы организации.

В конечном итоге, система продажи билетов является важным элементом в организации мероприятий, которая обеспечивает удобство и безопасность для клиентов и организаторов. С помощью системы продажи билетов клиенты могут легко выбирать и оплачивать желаемые места на мероприятии, а организаторы могут контролировать доступ к мероприятию и отслеживать продажи. Система продажи билетов также управляет финансами организации, позволяя ей получать доход от проданных билетов и распределять его среди всех заинтересованных сторон. Кроме того, система продажи билетов помогает планировать будущие мероприятия, анализируя данные о посещаемости, предпочтениях и отзывах клиентов. Она может повысить эффективность организации мероприятий и улучшить опыт посетителей.

# **1 ТЕХНИЧЕСКИЕ ПОДХОДЫ К РЕАЛИЗАЦИИ ПРИЛОЖЕНИЯ ДЛЯ СИСТЕМЫ ПРОДАЖИ БИЛЕТОВ**

## **Обзор аналогов**

В настоящее время на рынке существует огромное количество различных систем продажи билетов, предоставляемых компаниями-поставщиками. Каждая из этих систем обладает своими уникальными особенностями и функциональностью, которые могут быть востребованы для различных типов мероприятий и организаторов.

Некоторые системы продажи билетов могут предоставлять возможность онлайн-бронирования мест, что особенно удобно для крупных мероприятий, где необходимо забронировать много мест для групп посетителей. Другие системы могут предоставлять возможность создания персонализированных билетов с уникальными *QR*-кодами для более безопасного контроля доступа на мероприятие.

Для театров, как и для любых других мероприятий, существует множество систем продажи билетов, которые предоставляют различные инструменты для управления продажами и улучшения опыта посетителей. Одной из таких систем является *Ticketmaster*. Это крупнейшая система продажи билетов в мире, которая позволяет театрам создавать схему зала и бронировать места, а также отправлять электронные билеты на почту или мобильный телефон посетителя. Кроме того, *Ticketmaster* предлагает инструменты для управления продажами, маркетинга и аналитики, что может быть полезно для организаторов театральных мероприятий.

Другой системой продажи билетов, которая может быть полезна для театров, является *Eventbrite*. Эта система позволяет театрам создавать персонализированные страницы мероприятий с подробной информацией, фотографиями и видео. Театры могут использовать *Eventbrite* для продажи билетов на свои спектакли и события, а также для сбора пожертвований и регистрации гостей.

Еще одной системой продажи билетов, которая может быть полезна для театров, является *Brown Paper Tickets*. Эта система специализируется на мероприятиях с небольшим бюджетом, включая театральные спектакли. *Brown Paper Tickets* предоставляет возможность продажи билетов онлайн и в реальном времени, а также отправляет билеты на почту или мобильный телефон посетителя.

Наконец, система *Spektrix* также может быть полезна для театров. Она предлагает инструменты для управления продажами, маркетинга и аналитики, а также позволяет театрам создавать персонализированные билеты с уникальными *QR*-кодами, которые могут использоваться для контроля доступа на мероприятие.

В целом, выбор системы продажи билетов для театров зависит от конкретных потребностей и желаемых функций. Каждая из упомянутых выше платформ предлагает уникальные возможности для управления продажами и улучшения опыта посетителей.

## **1.2 Объектно-ориентированное программирование**

ООП (объектно-ориентированное программирование) *–* это подход к программированию, в котором программа разбивается на объекты, каждый из которых имеет свои свойства и методы. В ООП данные и функции, работающие с этими данными, объединяются в объекты, которые обладают определенным поведением.

Основные принципы ООП включают в себя наследование, инкапсуляцию и полиморфизм. Наследование позволяет создавать новые классы на основе уже существующих, а инкапсуляция *–* скрыть детали реализации объекта от пользователя и предоставить только интерфейс для работы с ним. Полиморфизм позволяет использовать один и тот же интерфейс для работы с разными объектами, что делает код более гибким и удобным для использования.

ООП позволяет писать более читаемый и понятный код, что упрощает его сопровождение и развитие. В ООП используются классы, объекты, наследование и другие конструкции, которые позволяют разбивать код на логические блоки и обеспечивают лучшую организацию и структурирование кода.

ООП широко используется в современном программировании и применяется в различных областях, от создания веб-сайтов до разработки игр и мобильных приложений. Некоторые языки программирования, такие как *Java*, *Python* и *C*++, предназначены для ООП и предоставляют мощные инструменты для создания объектно-ориентированных программ.

ООП также позволяет использовать наследование для повторного использования кода и уменьшения дублирования. Кроме того, объектно-ориентированное программирование позволяет разделять ответственность между различными объектами, что упрощает разработку сложных систем.

Другими принципами ООП являются абстракция, полезная для описания общих понятий и выделения ключевых характеристик, и композиция, позволяющая объединять объекты в более крупные структуры.

Важным аспектом ООП является использование классов для создания объектов. Класс определяет свойства и методы объекта, а каждый созданный объект имеет свои собственные значения свойств, но использует одни и те же методы класса.

Некоторые известные примеры объектно-ориентированных языков программирования включают Java, Python, C++, Ruby и PHP. В целом, ООП представляет собой мощный инструмент для разработки программного обеспечения, который может значительно упростить процесс разработки и поддержки кода.

Другим важным принципом ООП является инкапсуляция, которая обеспечивает скрытие внутренней реализации объекта и предоставляет интерфейс для работы с ним. Это позволяет изменять внутреннюю реализацию объекта без влияния на другие части программы, использующие этот объект.

ООП также обеспечивает возможность полиморфизма, который позволяет объектам разных классов иметь одинаковые методы и использоваться в качестве аргументов функций без необходимости знать их конкретный тип.

Еще одним важным аспектом ООП является использование наследования интерфейсов, который позволяет создавать новые классы на основе уже существующих интерфейсов. Это позволяет увеличивать гибкость и расширяемость программы.

Наконец, ООП поддерживает принцип композиции объектов, который позволяет создавать сложные объекты из более простых, используя их в качестве компонентов. Это позволяет создавать гибкие и масштабируемые системы.

Общие преимущества ООП включают повторное использование кода, уменьшение дублирования, упрощение разработки и поддержки кода, улучшение гибкости и расширяемости программы, а также увеличение понятности и читаемости кода.

## **1.3 Принципы *SOLID***

*SOLID* – это аббревиатура, которая означает пять принципов объектно-ориентированного программирования, разработанных Робертом Мартином (*Robert C. Martin*). Эти принципы помогают разработчикам создавать гибкие, расширяемые и легко поддерживаемые приложения.

Вот краткое описание каждого из принципов *SOLID*:

1. Принцип единственной ответственности (*Single Responsibility Principle, SRP*) - класс должен иметь только одну причину для изменения. Каждый класс должен быть ответственен только за одну часть функциональности программы. Это позволяет легко поддерживать и тестировать код.
2. Принцип открытости/закрытости (*Open/Closed Principle, OCP*) - классы должны быть открыты для расширения, но закрыты для изменения. Это означает, что вы должны использовать наследование или интерфейсы для добавления новой функциональности, а не изменять существующий код.
3. Принцип подстановки Барбары Лисков (*Liskov Substitution Principle, LSP*) - объекты должны быть заменяемы друг за друга без изменения поведения программы. Это означает, что вы должны использовать наследование, только если это абсолютно необходимо, и если классы-наследники действительно могут заменять своих родителей.
4. Принцип разделения интерфейса (*Interface Segregation Principle, ISP*) - клиенты не должны зависеть от интерфейсов, которые они не используют. Лучше создавать несколько узких интерфейсов, чем один большой. Это уменьшает связанность между классами и делает код более гибким.
5. Принцип инверсии зависимостей (*Dependency Inversion Principle, DIP*) - модули верхнего уровня не должны зависеть от модулей нижнего уровня. Оба типа модулей должны зависеть от абстракций. Это означает, что вы должны программировать на уровне абстракций, а не конкретных реализаций.
6. Применение принципов *SOLID* помогает создавать более чистый, гибкий и легко поддерживаемый код, который легче расширять и модифицировать в будущем.

## **1.4 Модели хранения данных**

Существует ряд различных методов сохранения информации, каждый из которых специализирован для решения определенных задач и имеет свои сильные и слабые стороны.

Например, реляционные базы данных широко используются для хранения структурированных данных, таких как финансовые отчеты, данные клиентов и т.д. Однако они могут быть неэффективными для работы с неструктурированными данными, такими как изображения или видео.

Для хранения неструктурированных данных могут использоваться *NoSQL* базы данных, такие как *MongoDB* или *Cassandra*. Они хорошо подходят для хранения больших объемов данных, не имеющих строгой структуры, таких как данные социальных медиа, информация об устройствах Интернета вещей (*IoT*) и т.д.

Еще один подход - использование облачных хранилищ, таких как *Amazon S3* или *Microsoft Azure*. Эти хранилища обеспечивают высокую отказоустойчивость и масштабируемость, что делает их идеальными для хранения крупных объемов данных.

Независимо от выбранной модели хранения данных, важно выбирать ее в соответствии с требованиями вашей конкретной задачи, а также учитывать ограничения бюджета и доступные ресурсы.

Кроме того, при выборе модели хранения данных необходимо учитывать требования к скорости чтения и записи данных, а также к безопасности и конфиденциальности. Например, если данные содержат конфиденциальную информацию, то необходимо выбирать модель хранения данных, которая обеспечивает надежную защиту от несанкционированного доступа.

Также стоит учитывать возможность интеграции выбранной модели хранения данных с другими системами и приложениями, с которыми ваша компания работает.

В целом, выбор модели хранения данных - это компромисс между различными факторами, такими как требования к хранению данных, доступные ресурсы и бюджет, безопасность, скорость доступа и масштабируемость. Важно выбрать модель, которая наилучшим образом соответствует вашим требованиям и поможет эффективно решать поставленные задачи.

## **1.5 *Windows Forms* технология**

*Windows Forms* – это платформа пользовательского интерфейса для создания классических приложений *Windows*. Она обеспечивает один из самых

эффетивных способов создания классических приложений с помощью визуального конструктора в *Visual Studio*. Такие функции, как размещение визуальных элементов управления путем перетаскивания, упрощают создание классических приложений

В *Windows Forms* можно разрабатывать графически сложные приложения, которые просто развертывать, обновлять, и с которыми удобно работать как в автономном режиме, так и в сети. Приложения *Windows Forms* могут получать доступ к локальному оборудованию и файловой системе компьютера, на котором работает приложение.

Приложение *Windows Forms* представляет собой событийно-ориентированное приложение, поддерживаемое *Microsoft .NET Framework*. В отличие от пакетных программ, большая часть времени тратится на ожидание от пользователя каких-либо действий, как, например, ввод текста в текстовое поле или клика мышкой по кнопке. При выполнении пользователем какого-либо действия с формой или одним из ее элементов управления создается событие. Приложение реагирует на эти события, как задано в коде, и обрабатывает события при их возникновении.

*Windows Forms* также поддерживает многопоточность, что позволяет создавать приложения, которые могут выполнять несколько задач одновременно. Она также предоставляет широкий спектр визуальных элементов управления, таких как кнопки, поля ввода, списки и многое другое, которые могут быть легко настроены и управляемы в коде.

*Windows Forms* была представлена в .*NET Framework* 1.0 и быстро стала одной из наиболее популярных платформ для разработки классических приложений *Windows*. С развитием технологий, таких как *WPF* и *UWP, Windows Forms* по-прежнему остается популярным выбором для создания классических приложений и остается поддерживаемой в последних версиях .*NET Framework.*

## **1.6 Язык разметки документов *XML***

*XML (Extensible Markup Language) –* это язык разметки документов, используемый для хранения и обмена данными. XML был разработан с целью предоставить стандартный формат для обмена информацией между различными системами и приложениями.

*XML* используется во многих областях, включая веб-приложения, базы данных, настройки программного обеспечения, научные исследования и многое другое. В отличие от *HTML, XML* не описывает, как должна выглядеть информация на экране, а скорее определяет ее структуру и семантику.

Файлы *XML* состоят из элементов, которые определяют структуру данных. Каждый элемент содержит открывающий и закрывающий теги, между которыми находится содержимое элемента. Некоторые элементы могут содержать атрибуты, которые задают дополнительные характеристики элемента.

*XML* является расширяемым языком, что означает, что его можно легко адаптировать под конкретные потребности проекта. Например, разработчики могут определить свои собственные элементы, атрибуты и правила, которые будут использоваться в файле *XML*. Это делает *XML* очень гибким и удобным для использования в различных областях.

XML также поддерживает возможность использования схемы (XSD), которая определяет структуру и типы данных, которые могут содержаться в XML-документе. С помощью схемы можно проверять корректность структуры и содержимого XML-документа, что делает его более надежным и удобным для обмена данными.