**СОДЕРЖАНИЕ**

[Введение 2](#_Toc50226426)

[1. Организационная структура предприятия, оказываемые услуги 3](#_Toc50226427)

[1.1 История 3](#_Toc50226428)

[1.2 О производстве 5](#_Toc50226429)

[1.3 Оказываемые услуги 6](#_Toc50226430)

[1.4 Достижения 7](#_Toc50226431)

[1.5 Организационная структура предприятия 7](#_Toc50226432)

[1.6 Системы менеджмента 7](#_Toc50226433)

[1.7 Охрана окружающей среды 8](#_Toc50226434)

[1.8 Охрана труда 8](#_Toc50226435)

[2. Индивидуальное задание 9](#_Toc50226436)

[2.1 Структура программного продукта 9](#_Toc50226437)

[2.1.1 Логическая модель 9](#_Toc50226438)

[2.1.2 Физическая модель 12](#_Toc50226439)

[2.2 Описание программного обеспечения «IntelliJ IDEA» и технологии его эксплуатации на предприятии 14](#_Toc50226440)

[2.2.1 Описание программного обеспечения 14](#_Toc50226441)

[2.2.2 Технологии его эксплуатации на предприятии 15](#_Toc50226442)

[2.3 Язык 21](#_Toc50226443)

[2.4 СУБД и БД 23](#_Toc50226444)

[2.5 Демонстрация используемого продукта 28](#_Toc50226445)

[Заключение 30](#_Toc50226446)

# Введение

Производственная практика — первое знакомство студента с производственным процессом на практике. Практика позволяет закрепить полученные теоретические знания в ходе учебного процесса, а также получить новые практические знания и умения, которые в будущем понадобятся в профессиональной деятельности.

Условия производственной практики максимально приближены к будущей профессиональной деятельности студентов, что тоже оказывает большое влияние на формирование профессиональных навыков.

В качестве места прохождения практики мною была выбрана компания ОАО «Свiтанак». Набираться опыта и получать практические умения я буду в отделе автоматизированных систем управлением предприятия.

Целями производственной практики на предприятии ОАО «Свiтанак» является углубление теоретических знаний, а также получение практических навыков и ознакомление с производственным процессом компании.

# Организационная структура предприятия, оказываемые услуги

ОАО «Свiтанак» г. Жодино является одним из ведущих предприятий концерна Беллегпром и успешно работает на рынке трикотажа более 40 лет.

Продукцией предприятия являются трикотажные изделия из хлопка и смесей хлопка с полиэстером, эластаном, вискозой, льном. Для производства изделий компания использует только экологически чистые материалы с высокими гигиеническими показателями и прекрасными потребительскими свойствами. Компания производит одежду для взрослых и детей:

* белье;
* легкий верхний трикотаж для отдыха;
* домашний трикотаж;
* одежда для спорта.

Главная цель компании - производство качественной современной одежды. Продукцию ОАО «Свiтанак» за отличное качество и приемлемую цену уже оценили в России, а также в других странах СНГ, Германии, Италии, Швейцарии, странах Балтии.

Принципы работы:

* постоянное отслеживание новых тенденций в моде;
* совершенствование технологий и использование современного оборудования;
* соблюдение интересов клиентов;
* контроль качества изготавливаемой продукции.

«Свiтанак» продолжает укреплять свои позиции на рынке, повышая производительность и качество продукции. Каждый новый шаг деятельности предприятия направлен на то, чтобы стать лучше, а неотъемлемая философия коллектива «Свiтанка» - творчество, ответственность и энтузиазм работников.

## История

История предприятия начиналась в 1975 году, когда была образована Жодинская швейно-трикотажная фабрика — филиал Минского производственного объединения «Прогресс».

Государственный Знак качества, присвоенный продукции со второго полугодия 1977 года, служил надежной гарантией для более 200 торговых организаций из Москвы, Ленинграда, Средней Азии, Прибалтики.

Но, несмотря на достигнутые успехи, было ясно, что предприятие способно на гораздо большее. Развитие производства тормозила зависимость от привозного материала. Поэтому в 1981 году было начато строительство фабрики трикотажного полотна. Первый комплекс по производству трикотажного полотна был введен в действие в 1985 году, а уже в 1986 году был сдан в эксплуатацию второй комплекс — красильно-отделочное производство. В связи с этим в 1986 году Жодинская швейно-трикотажная фабрика была преобразована в самостоятельный субъект хозяйствования — «Жодинское производственное швейно-трикотажное объединение».

В начале 90-х гг. в новых экономических условиях требовались большая гибкость и способность быстрее реагировать на изменение конъюнктуры рынка.

Поэтому в 1991 году Производственное швейно-трикотажное объединение было преобразовано в Жодинскую промышленно-торговую фирму "Свiтанак". Логическим продолжением начатых в 1991 году перемен стало акционирование предприятия в 1994 году и его преобразование в ОАО «Свiтанак».

Сегодня ОАО «Свiтанак» — признанный лидер Беларуси по производству трикотажного белья и верхнего трикотажа для детей и взрослых.

Площадь предприятия составляет 10,7 га. Предприятие имеет в своем составе вязальный, красильный, подготовительно-закройный и швейный цеха, в которых работают более 3 тысяч работников.

Ассортимент изделий постоянно обновляется: внедряются новые модели, артикулы трикотажных полотен, разрабатываются колориты для набивных рисунков, новые рецептуры для крашения.

Ежемесячно выпускается около 500 моделей одежды.

Все коллекции одежды от «Свiтанка» — результат кропотливой работы дизайнерского центра предприятия. Модельеры предприятия — молодые, талантливые, неординарные люди, которые постоянно находятся в творческом поиске, разрабатывают и внедряют в производство новые модели одежды.

Основными потребителями продукции предприятия являются Республика Беларусь (около 55% выпускаемой продукции) и Российская Федерация (около 40%).

В Республики Беларусь свою продукцию акционерное общество реализует через собственные фирменные магазины и розничную торговую сеть. В Россию ОАО «Свiтанак» поставляет свой трикотаж через собственные торговые дома, а также через крупных дилеров. Кроме этого, продукция предприятия поставляется на рынки Италии, Германии, Нидерландов, стран СНГ и Балтии.

## О производстве

На предприятии осуществляется полный технологический цикл производства: от изготовления полотна до выпуска готовых изделий с печатями, вышивками и аппликациями. Чтобы быть на высоте мировых стандартов качества, предприятие непрерывно развивается, совершенствует технологии. Только за последнее десятилетие обновлена значительная часть швейного, красильно-отделочного и вязального оборудования. Установлены машины и устройства ведущих производителей из Германии, Швейцарии, Италии и других европейских стран, которые дают возможность «Свiтанку» выпускать более 12 миллионов изделий в год.

Краткая характеристика:

* Производственная мощность – 1 500 000 шт. в месяц.
* Количество швей – 500 человек.
* Время для производства 1 заказа – до 1 месяца (в зависимости от сложности).
* Фабрика имеет возможность обрабатывать и конструировать модели, разрабатывать лекала изделий.
* Вязание трикотажного полотна – 320 тонн в месяц.
* Крашение полотна – 320 тонн в месяц.
* Печатание полотна на печатной машине ротационного типа – 127 тонн в месяц.

Все производственные мощности позволяют принимать заказы на: вязание трикотажных полотен (от 130 до 400г/м2); отварка, беление, крашение трикотажных полотен; отделка; печать рисунка; настилание и раскрой полотна; вышивка; пошив. Вязание и отделка поперечнополосатых полотен неограниченного раппорта. Ворсование футерованных полотен. Нанесение штучной печати на детали кроя. Вышивка на деталях кроя. Компания «Свiтанак» открыта для сотрудничества на всех направлениях!

## Оказываемые услуги

ОАО «Свiтанак» г. Жодино предлагает оказание следующих услуг:

* Вязание полотен различных типов переплетений из хлопчатобумажной, вискозной, хлопкополиэфирной пряжи, а так же в сочетании с полиэфирными нитями, нитью спандекс:
* кулирного полотна на вязальных машинах 18, 20, 22, 24 и 28 классов;
* плюшевых полотен, плюшевых полотен «под бархат» на вязальных машинах 20 класса;
* интерлочных полотен на вязальных машинах 20,28 классов;
* ластичных полотен на вязальных машинах 15,18 классов;
* футерованного полотна на вязальных машинах 20 класса.
* Беление, крашение трикотажных полотен, объем крашения – от 50 кг полотна и более. Крашение осуществляется активными, дисперсными красителями.
* Отделка различных трикотажных полотен как в разрезном (ширина до 200 см) так и в круглом виде (ширина от 50 до 100 см). Возможна стабилизация полотна на сушильно-ширильных машинах, ворсование полотна, стрижка полотна «под бархат», аппретирование полотна, каландрирование полотна, ворсование. По завершению отделки полотна производится разбраковка полотна, намотка в рулон (разрезное полотно), упаковка полотна.
* Печать рисунка на трикотажных полотнах на ротационной печатной машине Образцы набивок полотна фотографии: скачать, с максимальной шириной печати 185 см, максимальное количество цветов – 7, с последующей фиксацией красителя на полотне. Печать производиться пигментными красителями. В случае необходимости производится подготовка полотна под печать. Возможна печать рисунка с использованием имеющихся на предприятии шаблонов, так и печать рисунка с использованием предварительно изготовленных шаблонов по рисунку заказчика.
* Настилание и раскрой полотна (как разрезного так и в круглом виде) на автоматизированном настилочно-раскройном комплексе. Максимальная ширина настилаемого и раскраиваемого полотна в разрезном виде – до 180 см, в круглом - до 100 см. Раскрой полотна на ленту (бейку).
* Печать рисунка на деталях кроя и готовых изделиях шелкографическим способом на автоматических печатных машинах по рисунку заказчика, пигментными красителями, количество цветов – до 8, размер печати – до 50 х 40 см.;
* Выполнение вышивки на деталях кроя по дизайну заказчика, количество цветов вышивки – до 12, максимальный размер вышивки – 45 х 52 см. Возможно выполнение вышивки с аппликациями; с использованием пайеток.
* Пошив трикотажных изделий детского, мужского и женского ассортимента из различного типа трикотажных полотен. При пошиве возможна декоративная отделка изделия (нанесение страз, термоапликаций, выполнение строчек «кроше», «рюша»; а так же возможна установка блочек, кнопок; дублирование деталей кроя.

## Достижения

Свидетельством высокого качества продукции являются многочисленные награды на различных выставках и конкурсах.

ОАО «Світанак» - лауреат премии правительства Республики Беларусь за достижения в области качества, призер многочисленных конкурсов и фестивалей, в числе которых «Лучшие товары Республики Беларусь», «Лучшие товары Республики Беларусь на рынке Российской Федерации», международный фестиваль-конкурс «Выбор года».

«Світанак» признан одним из самых известных брендов Беларуси по итогам конкурса «Бренд года».

## Организационная структура предприятия

## Системы менеджмента

Продукция ОАО «Свiтанак» - это продукция высокого качества, созданная из экологически чистых материалов с высокими гигиеническими показателями и прекрасными потребительскими свойствами.

Качество трикотажных изделий "Світанка" подтверждает сертификация предприятия по международной системе ISO 9001, а также Премия правительства Республики Беларусь за достижения в области качества.

На предприятии используются химикаты и красители лидеров мировой химической промышленности. Предприятие большое внимание уделяет экологической безопасности своей продукции, снижению и предотвращению негативного воздействия на окружающую среду при ее производстве.

## Охрана окружающей среды

Организация уделяет большое внимание снижению и предотвращению негативного воздействия на окружающую среду при производстве продукции.

Основой природоохранной деятельности ОАО «Свiтанак» является Экологическая политика. Экологическая политика содержит обязательства, принимаемые на себя руководством организации по снижению вредного воздействия на окружающую среду. ОАО «Свiтанак» имеет экологический сертификат соответствия требованиям СТБ ISO 14001-2005.

## Охрана труда

Большое внимание на предприятии уделяется охране труда. Реализация прав работников на здоровые и безопасные условия труда является одной из важнейших задач в данной области. Управление охраной труда является составной частью общей системы управления предприятием.

Для решения задач управления охраной труда на предприятии разработана Политика в области охраны труда. Политика соответствует характеру и масштабам производственной деятельности, продукции и услуг организации. Для выполнения обязательств, установленных в Политике в области охраны труда в организации установлены документально оформленные цели и разработана Программа по их достижению.

Организации выдан сертификат соответствия, который удостоверяет, что система управления охраной труда при проектировании, разработке и производстве трикотажных изделий соответствует требованиям СТБ 18001-2009.

# Индивидуальное задание

Программный продукт «lsFusion» является основным для управления административными процессами в рамках организации. Он представляет собой кроссплатформенное клиент-серверное приложение, разработанное на языке *lsFusion*, используемое в качестве системы управления базами данных PostgreSQL. Приложение упрощает деятельность отделов маркетинга, продаж, розничной торговли, финансов, закупок, склада и многих других.

## 2.1 Структура программного продукта

Парадигму можно условно разделить на две части: логическую модель и физическую модель.

### Логическая модель

Логическая модель описывает непосредственно функционал системы, то есть то, что система умеет делать, какие данные принимает на вход, и какие отдает на выход.

Логическая модель традиционно делится на две части:

* Логика предметной области, которая определяет, какую информацию хранит система, и что с этой информацией можно делать. Эту логику также часто называют бизнес-логикой (рисунок 2.1).
* Логика представлений, которая включает в себя все то, что связано с отображением информации субъектам, использующим информационную систему – пользователям и другим информационным системам (рисунок 2.2).

**Логика предметной области.**

Свойства являются базовым понятием логики предметной области и отвечают за хранение и вычисление данных.

Важным отличием свойств является то, что они никак не изменяют данные в системе - для этого в платформе предназначены действия.

Действия отвечают на вопрос что делать, но не отвечают на вопрос когда это делать. Для определения таких моментов в платформе существуют события.

Как правило, логика предметной области разрешает не любые изменения данных, а только изменения с учетом некоторых правил. Для задания этих правил в платформе используются ограничения.

Соответственно, сводная таблица по всем элементам логики предметной области выглядит следующим образом:

|  |  |  |
| --- | --- | --- |
| **Элементы системы** | **Отвечают на вопрос** | **Характер** |
| Свойства | Что хранить и вычислять | статический / декларативный |
| Действия | Что делать | динамический / императивный |
| События | Когда делать | динамический / императивный |
| Ограничения | Что можно делать | статический / декларативный |

Таблица 2.1 – Описание элементов системы.

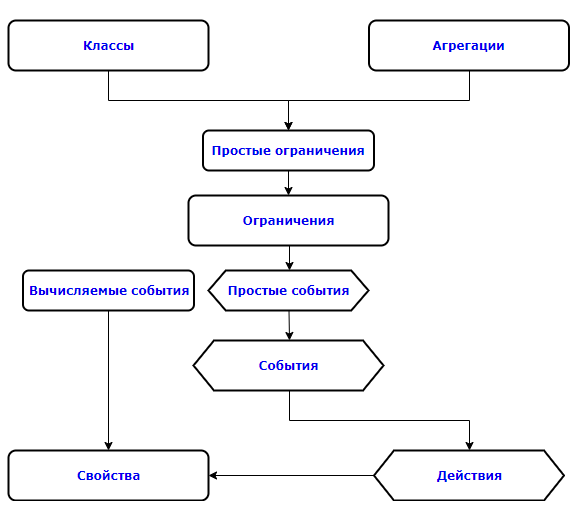


Рисунок 2.1 – Схема логики предметной области

**Логика представлений.**

Ключевым элементом логики представлений является форма. Именно при помощи форм в платформе организуется ввод и вывод информации в систему (из системы).

В самом простом варианте пользовательского интерфейса пользователю открываются несколько фиксированных форм, с которыми он будет работать, переключаясь между вкладками. Однако если форм, с которыми работает пользователь, много, открывать их все сразу не очень удобно. В таких случаях для управления работой с формами можно использовать так называемый навигатор. В этом случае, при старте клиента пользователю показывается только сам навигатор (никакие формы не открываются), и пользователь может открывать формы самостоятельно по мере их необходимости. Более того, в навигаторе пользователь может не только открывать формы в интерактивном представлении, но и вообще выполнять любые действия.

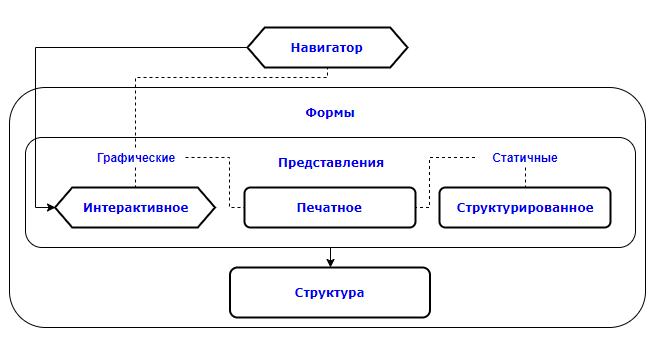


Рисунок 2.2 – Схема логики представлений.

### Физическая модель

Физическая модель состоит из трех частей:

* Разработка. Включает в себя все то, что связано с процессом разработки.
* Выполнение. Включает в себя все то, что связано с повышением производительности системы.
* Управление. Включает в себя все то, что связано с операциями над уже работающей системой.

Схема физической модели отображена на рисунке 2.3.



Рисунок 2.3 – Схема физической модели.

**Разработка.**

Модель разработки включает в себя следующие понятия:

* Идентификация элементов. Описывает механизмы, при помощи которых в платформе элементы системы могут обращаться друг к другу.
* Модульность. Позволяет упростить задачу проектирования ИС и распределить процесс разработки ИС между группами разработчиков.
* Метапрограммирование. Предоставляет максимальную гибкость в вопросах повторного использования кода.
* Миграция. Позволяет осуществлять контроль за миграцией данных при изменении логики приложения.
* Интернационализация. Позволяет адаптироваться к языковым и культурным особенностям различных стран и регионов.
* Интеграция. Описывает механизмы, при помощи которых система может обращаться к сторонним системам и наоборот.

**Выполнение.**

Если система небольшая и данных в ней немного, как правило, она работает достаточно эффективно без каких-либо дополнительных оптимизаций. Если же логика становится достаточно сложной, а количество данных значительно возрастает, часто имеет смысл подсказать платформе как лучше хранить и обрабатывать все эти данные.

В платформе есть два основных механизма работы с данными : свойства и действия. Первый отвечает за хранение и вычисление данных, второй - за перевод системы из одного состояние в другое. И если работа действий поддается оптимизации достаточно ограниченно (в том числе из-за эффекта последействия), то для свойств существует целый набор возможностей, которые позволяют как уменьшить время отклика конкретных операций, так и увеличить общую производительность системы:

* Материализации. Если чтений свойства достаточно много (значительно больше чем изменений), можно существенно улучшить производительность операций, использующих такое свойство, материализовав его.
* Индексы. Если свойство часто участвует в вычислениях других свойств, как правило, имеет смысл построить индекс по этому свойству.
* Таблицы. Если для одного и того же набора объектов часто читаются / изменяются одновременно одни и те же свойства, хранить каждое такое свойство отдельно может быть достаточно неэффективно. Поэтому в платформе хранение свойств можно "группировать" в так называемые таблицы.

**Управление.**

Модуль управления включает в себя следующие понятия:

* Системные параметры. Содержит описание существующих в системе параметров, тем или иным образом влияющих на ее запуск или работу.
* Пользовательский интерфейс. Содержит описание возможностей настройки графического интерфейса пользователем.
* Интерпретатор. Описывает механизмы, при помощи, которых администратор может выполнять код на языке lsFusion в уже работающей системе (то есть после ее старта).
* Политика безопасности. Содержит описание настроек доступа к элементам системы в зависимости от работающего в системе пользователя.
* Планировщик. Описывает механизмы, позволяющие выполнять заданные действия по расписанию.
* Журналы и логи. Содержит описание журналов и логов, позволяющих получить информацию о различных событиях, происходивших в системе.
* Монитор процессов. Описывает механизмы, позволяющие получить информацию о текущем состоянии системы (ее производительности на текущий момент времени).
* Профайлер. Описывает механизмы, позволяющие проанализировать производительность системы в течении некоторого промежутка времени.
* Чат. Описывает механизмы общения с пользователями внутри системы.

## Описание программного обеспечения «IntelliJ IDEA» и технологии его эксплуатации на предприятии

### Описание программного обеспечения

IntelliJ IDEA — интегрированная среда разработки программного обеспечения для многих языков программирования, в частности Java, JavaScript, Python, разработанная компанией JetBrains.

Первая версия появилась в январе 2001 года и быстро приобрела популярность как первая среда для Java с широким набором интегрированных инструментов для рефакторинга, которые позволяли программистам быстро реорганизовывать исходные тексты программ. Дизайн среды ориентирован на продуктивность работы программистов, позволяя сконцентрироваться на функциональных задачах, в то время как IntelliJ IDEA берёт на себя выполнение рутинных операций.

Начиная с шестой версии продукта IntelliJ IDEA предоставляет интегрированный инструментарий для разработки графического пользовательского интерфейса. Среди прочих возможностей, среда хорошо совместима со многими популярными свободными инструментами разработчиков, такими как CVS, Subversion, Apache Ant, Maven и JUnit. В феврале 2007 года разработчики IntelliJ анонсировали раннюю версию плагина для поддержки программирования на языке Ruby.

Среда доступна в двух редакциях: Community Edition и Ultimate Edition. Community Edition является полностью свободной версией, доступной под лицензией Apache 2.0, в ней реализована полная поддержка Java SE, Kotlin, Groovy, Scala, а также интеграция с наиболее популярными системами управления версиями. В редакции Ultimate Edition, доступной под коммерческой лицензией, реализована поддержка Java EE, UML-диаграмм, подсчёт покрытия кода, а также поддержка других систем управления версиями, языков и фреймворков.

Поддерживаемые языки:

* Java;
* JavaScript;
* CoffeeScript;
* HTML/XHTML/HAML;
* CSS/SASS/LESS;
* XML/XSL/XPath;
* YAML;
* ActionScript/MXML;
* Python;
* Ruby;
* Haxe;
* Groovy;
* Scala;
* SQL;
* PHP;
* Kotlin;
* Clojure;
* Си;
* C++;
* Go.

### Технологии его эксплуатации на предприятии

Рассмотрим технологию эксплуатации на примере создания нового lsFusion проекта. Для начала необходимо запустить программу IntelliJ IDEA.

Выбираем пункт Create New Project при старте IDEA. Или, если IDEA уже запущена, выбираем в меню пункт File -> New -> Project (рисунок 2.4 и 2.5).

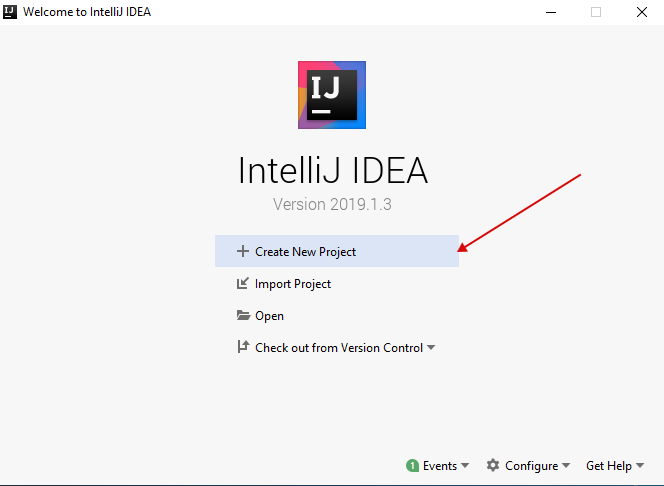


Рисунок 2.4 – Главное окно IDE

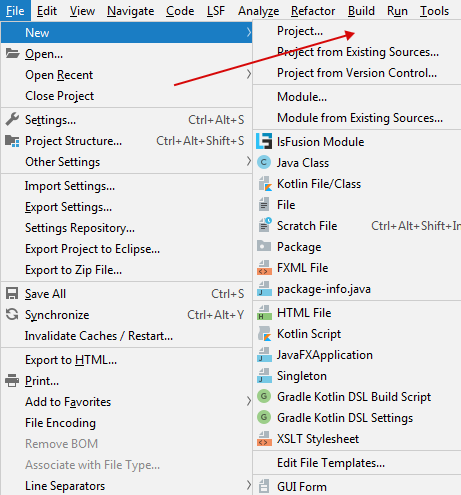


Рисунок 2.5 – Создание нового проекта

Выбираем тип проекта lsFusion. Проверяем, что задан JDK и выбрана ссылка на библиотеку с сервером lsFusion (при автоматической установке эта ссылка устанавливается автоматически, если ее все же нет - скачиваем Download или выбираем уже скачанную библиотеку Create), нажимаем Next (рисунок 2.6).

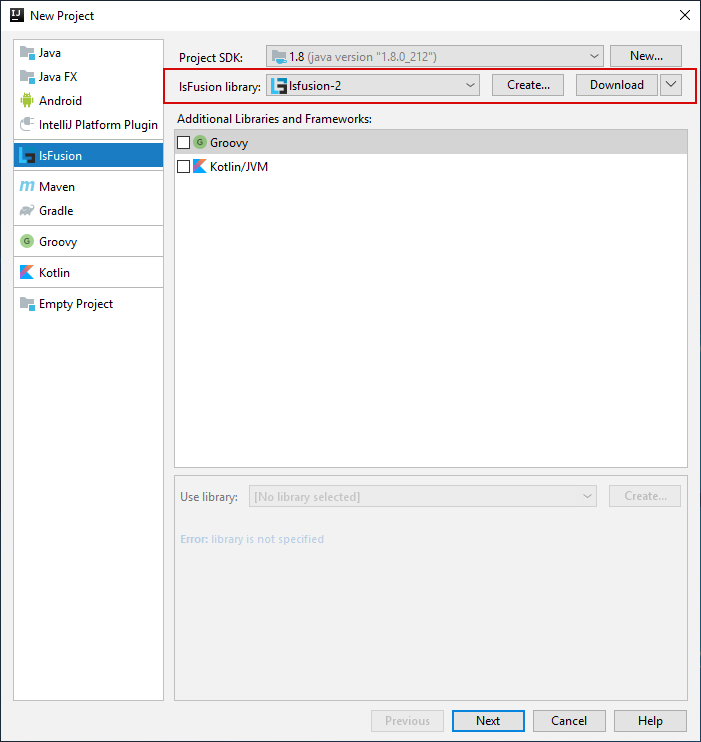


Рисунок 2.6 – Выбор библиотеки

Заполняем имя проекта и каталог, в котором будут находиться исходные файлы. При необходимости изменяем параметры подключения к базе данных и сервера приложений. Нажимаем Finish (рисунок 2.7).

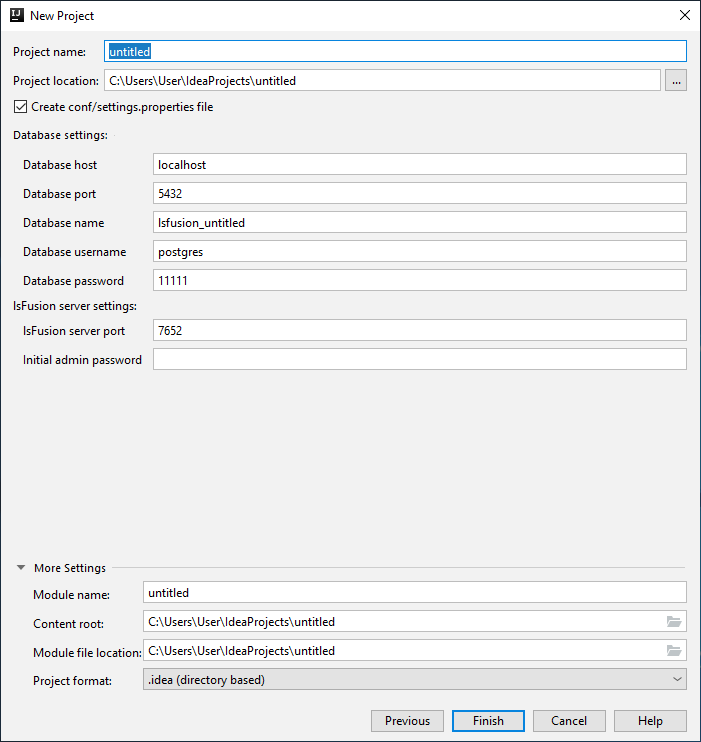


Рисунок 2.7 – Заполнение параметров проекта

Перейдем к разработке приложения.

Весь исходный код на языке lsFusion, по умолчанию, находится в папке src/main/lsfusion.

Добавить новый модуль можно нажав правой кнопкой мыши на соответствующей папкой и выбрав пункт меню New / lsFusion (рисунок 2.8):

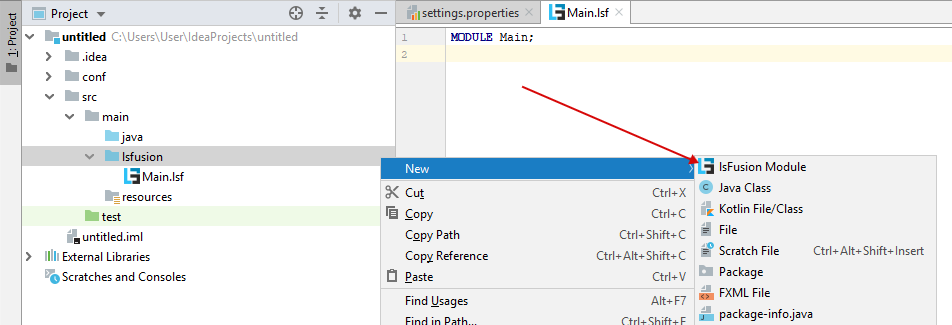


Рисунок 2.8 – Добавление нового модуля

Аналогичным образом, при необходимости, можно создавать вложенные папки (пункт Package), группируя тем самым логически связанные модули в директории.

Перейдем к запуску сервера.

По умолчанию, при создании нового проекта создается конфигурация запуска сервера. Соответственно запустить ее, можно выбрав пункт меню Run -> Run 'Run lsFusion server', или нажав на значок рядом с надписью Run lsFusion server в правом верхнем углу. При удачном старте самой последней строкой в логе должна быть Server has successfully started (рисунок 2.9).

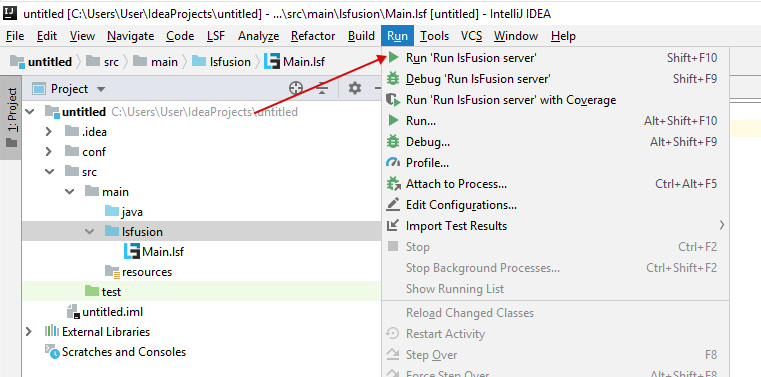


Рисунок 2.9 – Запуск сервера

После того, как сервер успешно запущен, к нему можно подключаться при помощи веб или десктоп клиента через ярлык на рабочем столе.

Прейдем к созданию конфигурации запуска сервера.

Если платформа подключается к уже существующему проекту (например через Maven к maven проекту), может потребоваться создать конфигурацию запуска сервера приложений вручную. Делается это следующим образом:

Выбираем пункт меню Edit configurations (рисунок 2.10):

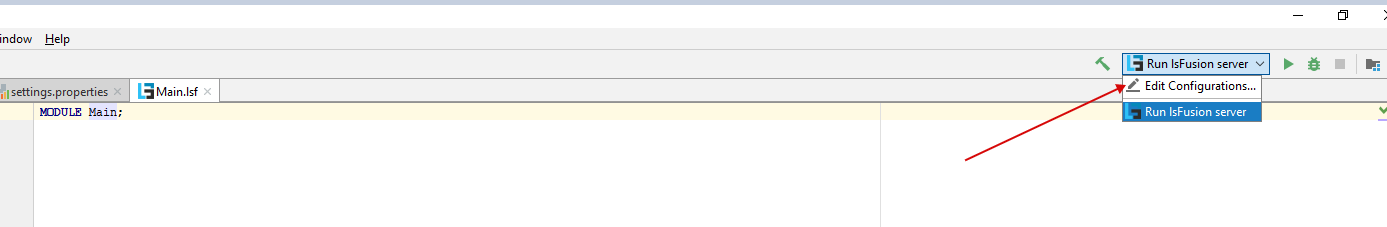


Рисунок 2.10 – Выбор пункта меню

В открывшемся окне, жмем + в левом верхнем углу и в выпавшем списке выбираем lsFusion Server (рисунок 2.11).

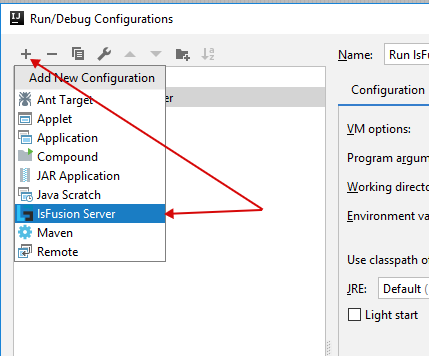


Рисунок 2.11 – Выбор сервера

Задаем имя конфигурации и модуль (если их несколько), для которого необходимо стартовать сервер приложений lsFusion. Также в этом окне, при необходимости, можно задать процессы сборки, которые надо выполнить перед запуском конфигурации, дополнительные параметры виртуальной машины Java, и т.п (рисунок 2.12).

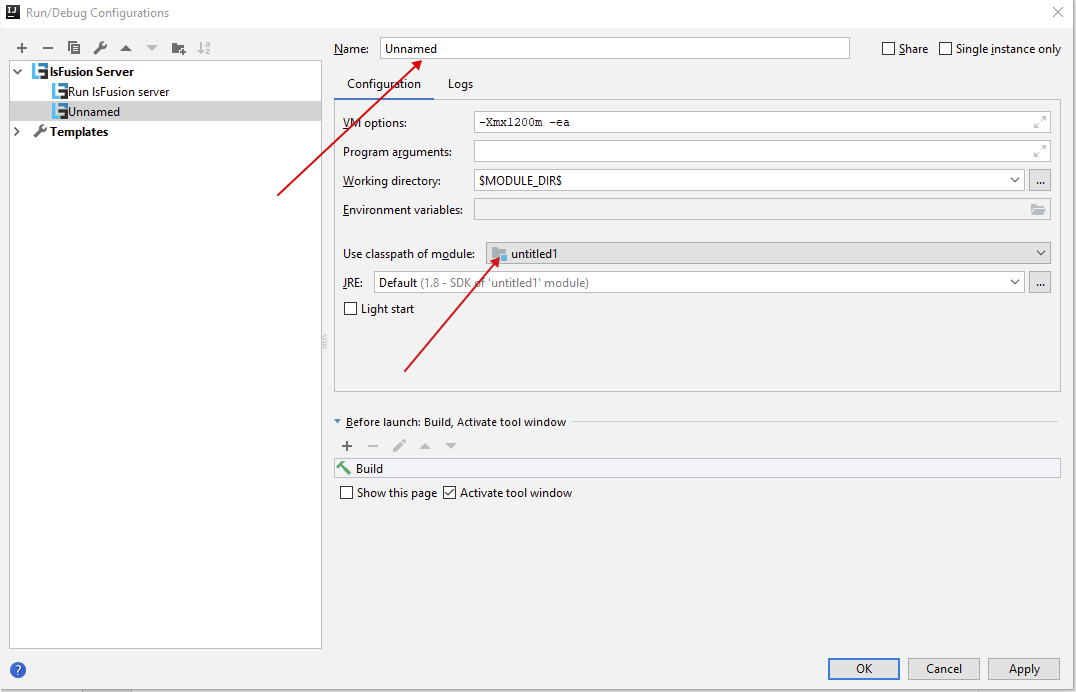


Рисунок 2.12 – Конфигурирование запуска

## Язык

lsFusion - это язык программирования, на котором описываются модули lsFusion.

Модуль - это некоторая функционально законченная часть проекта. Модуль состоит из объявлений свойств, действий, событий, ограничений и других элементов системы.

Каждый модуль имеет имя, которое должно быть уникальным в пределах проекта.

Обычно модули для описания части своего функционала используют элементы из других модулей. Соответственно, если модуль B использует элементы из модуля A, то в модуле B необходимо указать, что он зависит от A. На основании этих зависимостей все модули в проекте выстраиваются в некотором порядке, в котором происходит их инициализация. Гарантируется, что если модуль B зависит от модуля A, то инициализация модуля A произойдет раньше, чем инициализация модуля B. Циклические зависимости между модулями в проекте не допускаются.

Если модуль C зависит от модуля B, а модуль B зависит от модуля A, то будем также считать, что модуль С зависит от модуля A.

Любой модуль всегда автоматически зависит от системного модуля System, вне зависимости от того указано это явно или нет.

В каждом модуле указывается пространство имен, внутри которого будут находиться имена всех созданных в этом модуле элементов системы. По умолчанию, модуль создает собственное пространство имен, имя которого равно имени модуля. Также для модуля можно задать список дополнительных пространств имен, которые будут иметь приоритет при поиске элементов системы.

Каждому модулю в платформе соответствует ровно один файл, начинающийся со специального заголовка.

Каждый модуль состоит из последовательности инструкций, разделенных на заголовок и тело модуля. Заголовок может включать в себя четыре специальные инструкции: MODULE, REQUIRE, PRIORITY и NAMESPACE, описывающие свойства модуля. После заголовка идет тело модуля, которое состоит из остальных инструкций, и описывает логику работы этого модуля.

Пример кода на языке lsFusion отображен на рисунке 2.13.

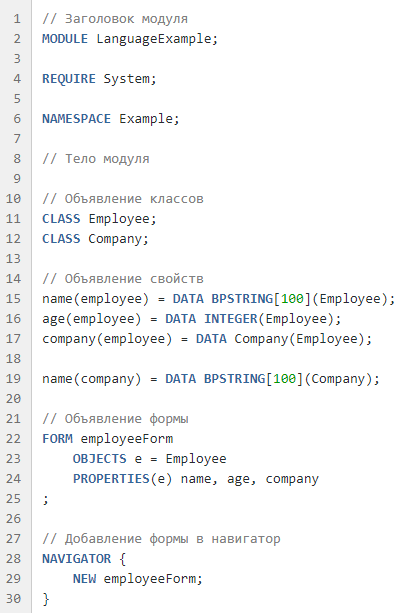


Рисунок 2.13 – Пример кода на языке lsFusion.

## СУБД и БД

**База данных (БД)** — это организованная структура, предназначенная для хранения информации. Обычно БД представляются в виде совокупности взаимосвязанных файлов или таблиц, предназначенных для решения конкретной задачи.

**Система управления базами данных (СУБД)** — совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

Основные функции СУБД:

* Управление данными во внешней памяти (на дисках);
* Управление данными в оперативной памяти с использованием дискового кэша;
* История изменений, резервное копирование и восстановление базы данных после сбоев;
* Поддержка языков БД (язык определения данных, язык манипулирования данными).

Обычно современная СУБД содержит следующие компоненты:

* Ядро, которое отвечает за управление данными во внешней и оперативной памяти и журнализацию,
* Процессор языка базы данных, обеспечивающий оптимизацию запросов на извлечение и изменение данных и создание, как правило, машинно-независимого исполняемого внутреннего кода,
* Подсистему поддержки времени исполнения, которая интерпретирует программы манипуляции данными, создающие пользовательский интерфейс с СУБД, а также сервисные программы (внешние утилиты), обеспечивающие ряд дополнительных возможностей по обслуживанию информационной системы.

Существует 4 основных типа отношений между таблицами:

* Один к одному. В этом случае каждой записи одной таблицы соответствует только одна запись другой таблицы;
* Один ко многим. Это когда одной записи главной таблицы (master) соответствует несколько записей подчиненной таблицы (detail). То есть, каждой записи, которая есть первичным ключом одной таблицы, соответствует несколько записей связанной таблицы;
* Многие к одному. Это когда нескольким записям главной таблицы отвечает одна запись подчиненной таблицы;
* Многие ко многим. Это когда в обоих таблицах существует несколько взаимосвязанных записей.

При разработке данного программного продукта была использована СУБД PostgreSQL.

PostgreSQL — [свободная](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%9F%D0%9E) [объектно-реляционная](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D1%80%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94) [система управления базами данных](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) (СУБД).

Возможности данной СУБД:

* **Функции**

Функции являются блоками кода, исполняемыми на сервере, а не на клиенте БД. Хотя они могут писаться на чистом SQL, реализация дополнительной логики, например, [условных переходов](https://ru.wikipedia.org/wiki/%D0%A3%D1%81%D0%BB%D0%BE%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BF%D0%B5%D1%80%D0%B5%D1%85%D0%BE%D0%B4) и [циклов](https://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D0%BA%D0%BB_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), выходит за рамки SQL и требует использования некоторых языковых расширений. Функции могут писаться с использованием одного из следующих языков:

* Встроенный процедурный язык PL/pgSQL, во многом аналогич-ный языку PL/SQL, используемому в СУБД Oracle;
* Скриптовые языки —PL/Lua, PL/LOLCODE, PL/Perl, PL/PHP, PL/Python, PL/Ruby, PL/sh, PL/Tcl, PL/Scheme, PL/v8 (Javascript);
* Классические языки — C, C++, Java (через модуль PL/Java);
* Статистический язык R (через модуль PL/R).

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](https://ru.wikipedia.org/wiki/%D0%A5%D1%80%D0%B0%D0%BD%D0%B8%D0%BC%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%B0), однако между этими понятиями есть различие. С девятой версии возможно написание автономных блоков, которые позволяют выполнять код на процедурных языках без написания функций, непосредственно в клиенте.

* **Триггеры**

Триггеры 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.

* **Правила и представления**

Механизм правил представляет собой механизм создания пользовательских обработчиков не только [DML](https://ru.wikipedia.org/wiki/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. Хорошим примером является реализация механизма [представлений](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_(%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *views*): при создании представления создается правило, которое определяет, что вместо выполнения операции выборки к представлению система должна выполнять операцию выборки к базовой таблице/таблицам с учётом условий выборки, лежащих в основе определения представления. Для создания представлений, поддерживающих операции обновления, правила для операций вставки, изменения и удаления строк должны быть определены пользователем.

* **Индексы**

В PostgreSQL имеется поддержка [индексов](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B4%D0%B5%D0%BA%D1%81_(%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)) следующих типов: [B-дерево](https://ru.wikipedia.org/wiki/B-%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE), [хеш](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D0%B0), [GiST](https://ru.wikipedia.org/wiki/GiST), [GIN](https://ru.wikipedia.org/wiki/GIN), [BRIN](https://ru.wikipedia.org/wiki/BRIN), [Bloom](https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%BB%D1%8C%D1%82%D1%80_%D0%91%D0%BB%D1%83%D0%BC%D0%B0). При необходимости можно создавать новые типы индексов. Индексы в PostgreSQL обладают следующими свойствами:

* возможен просмотр индекса не только в прямом, но и в обратном порядке — создание отдельного индекса для работы конструкции ORDER BY ... DESC не нужно;
* возможно создание индекса над несколькими столбцами таблицы, в том числе над столбцами различных типов данных;
* индексы могут быть функциональными, то есть строиться не на базе набора значений некоего столбца/столбцов, а на базе набора значений функции от набора значений;
* индексы могут быть частичными, то есть строиться только по части таблицы (по некоторой её проекции); в некоторых случаях это помогает создавать намного более компактные индексы или достигать улучшения производительности за счёт использования разных типов индексов для разных (например, с точки зрения частоты обновления) частей таблицы;
* планировщик запросов может использовать несколько индексов одновременно для выполнения сложных запросов.
* **Многоверсионность**

PostgreSQL поддерживает одновременную модификацию БД несколькими пользователями с помощью механизма Multiversion Concurrency Control (MVCC). Благодаря этому соблюдаются требования ACID и практически отпадает нужда в блокировках чтения.

* **Множество типов данных**

PostgreSQL поддерживает большой набор встроенных типов данных:

* Численные типы
* Целые
* С фиксированной точкой
* С плавающей точкой
* Денежный тип (отличается специальным форматом вывода, а в остальном аналогичен числам с фиксированной точкой с двумя знаками после запятой)
* Символьные типы произвольной длины
* Двоичные типы (включая BLOB)
* Типы «дата/время» (полностью поддерживающие различные форматы, точность, форматы вывода, включая последние изменения в часовых поясах)
* Булев тип
* Перечисление
* Геометрические примитивы
* Сетевые типы
* IP и IPv6-адреса
* CIDR-формат
* MAC-адрес
* UUID-идентификатор
* XML-данные
* Массивы
* JSON
* Идентификаторы объектов БД
* **Псевдотипы**

Более того, пользователь может самостоятельно создавать новые требуемые ему типы и программировать для них механизмы индексирования с помощью GiST.

* **Расширение пользовательскими объектами**

PostgreSQL может быть расширен пользователем для собственных нужд 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)
* Процедурные языки
* **Наследование**

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 PostgreSQL 10 был добавлен механизм [партицирования](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) таблиц. [Партицирование](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) предназначено для разделения одной таблицы на несколько, так называемые партиции. [Партицирование](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) схоже с наследованием, но имеет более дружелюбный к пользователю синтаксис и более строгие ограничения, что позволяет выполнять дополнительные оптимизации при планировании запросов.

## Демонстрация используемого продукта

Для демонстрации будем использовать веб-версию программного продукта.

Запустив приложение, первое, что видит пользователь - это форма авторизации, с помощью которой ведется разграничение прав доступа к функциям с помощью распределения ролей между зарегистрированными пользователями.

На рисунке ниже отображена форма авторизации.

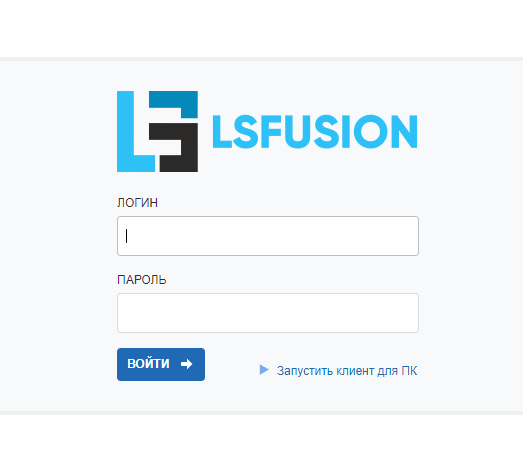


Рисунок 2.14 - Форма авторизации

Введя логин и пароль и успешно пройдя авторизацию, пользователь увидит список доступных ему разделов, в которых содержатся функции, доступные для роли данного пользователя. На рисунке 2.15 отражена рабочая область программы.

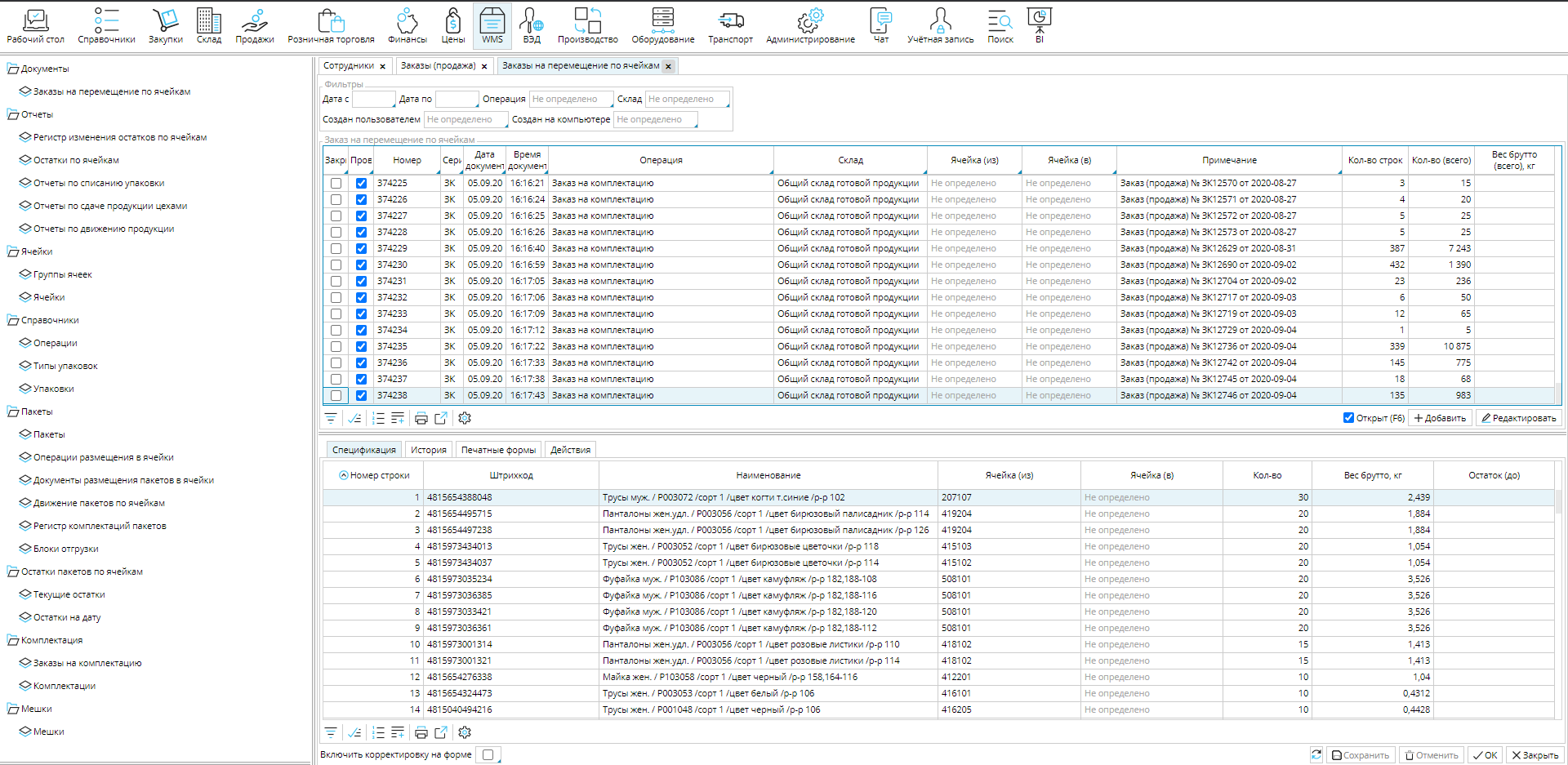


Рисунок 2.15 - Рабочая область программы

# Заключение

За время прохождения практики в ОАО «Свiтанак», я подкрепил полученные во время учебы теоретические знания практическими навыками. Познакомился с рабочими процессами в организации, структурой и принципами функционирования предприятия. Развил коммуникативные навыки. Получил опыт работы в коллективе.

Во время прохождения практики расширил свои знания в области информационных систем, компьютерных сетей.

В данной практике перед нами было представлено множество целей:

* ознакомление со структурой предприятия, организацией и управлением;
* изучить основные информационные потоки и документооборот на предприятии;
* углубить и закрепить знания, полученные при изучении специальных дисциплин.

Проходя практику выполнил все поставленные задачи. Индивидуальное задание, полученное на предприятии, было выполнено полностью и в срок.

В результате прохождения практики я научился работать с современными системами программного обеспечения, операционными системами, облачными системами, обслуживающими сервисными программами.

# Список использованной литературы

1. Эффективная Java – Джошуа Блох
2. Clean Code: A Handbook of Agile Software Craftsmanship – Robert C. Martin
3. Java. Библиотека профессионала – Кей С. Хорстманн
4. Head First. Паттерны проектирования – Эрик Фримен, Элизабет Робсон
5. Test Driven: TDD and Acceptance TDD for Java Developers – Lasse Koskela
6. [https://documentation.lsfusion.org/](https://documentation.lsfusion.org/pages/viewpage.action?pageId=18645035)
7. <https://ru.wikipedia.org/>
8. <https://www.postgresql.org/>
9. https://lsfusion.org/