2 Разработка программного продукта

2.1 Разработка архитектуры программной системы

Структура программы представлена на рисунке 2.1.

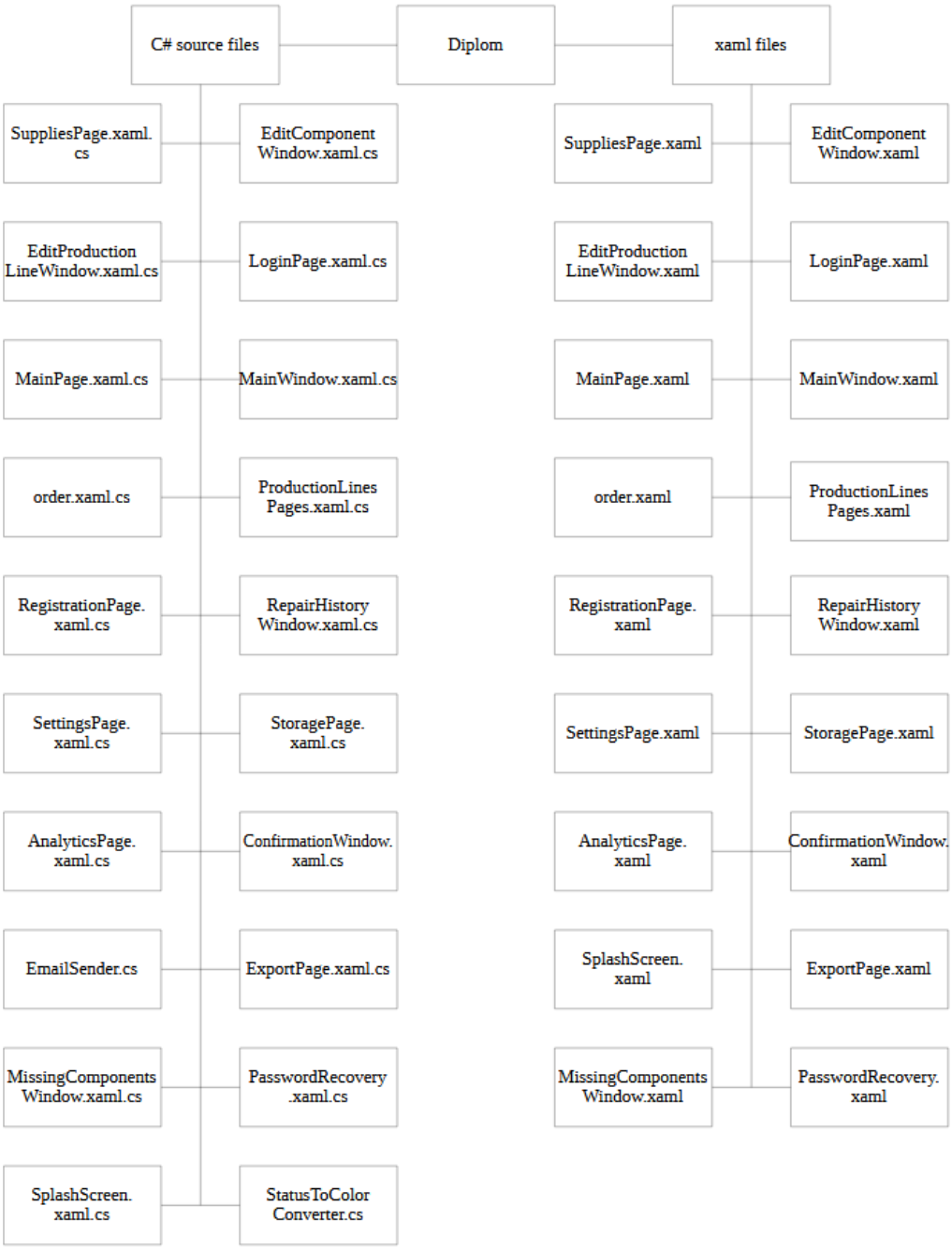


Рисунок 2.1 – Структура программы

Программный комплекс CompWare состоит из следующих ключевых модулей:

- модуль авторизации и регистрации (Управление доступом пользователей);
- модуль главной панели (Центральный узел управления);
- модуль управления складом (Контроль остатков товаров);
- модуль обработки поставок (Управление цепочками поставок);
- модуль управления заказами (Создание заказов);
- модуль производственных линий (Контроль производственных процессов);
- модуль аналитики (Анализ производственных показателей);
- модуль экспорта данных (Выгрузка информации);
- модуль системных настроек (Настройки аккаунта пользователя).

На рисунке 2.2 представлена графическая схема взаимодействия программных модулей.

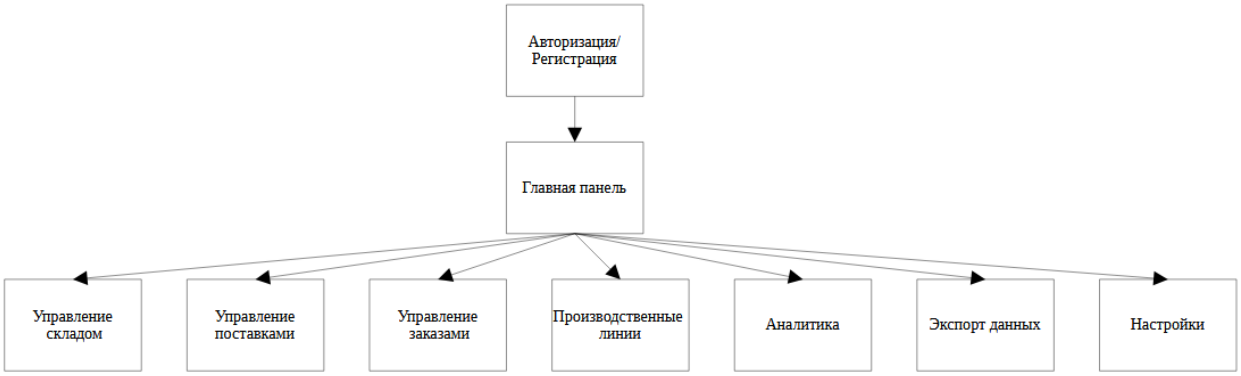


Рисунок 2.2 – Графическая схема взаимодействия программных модулей

2.2 Разработка структуры данных

На рисунке 2.3 представлена физическая схема данных.

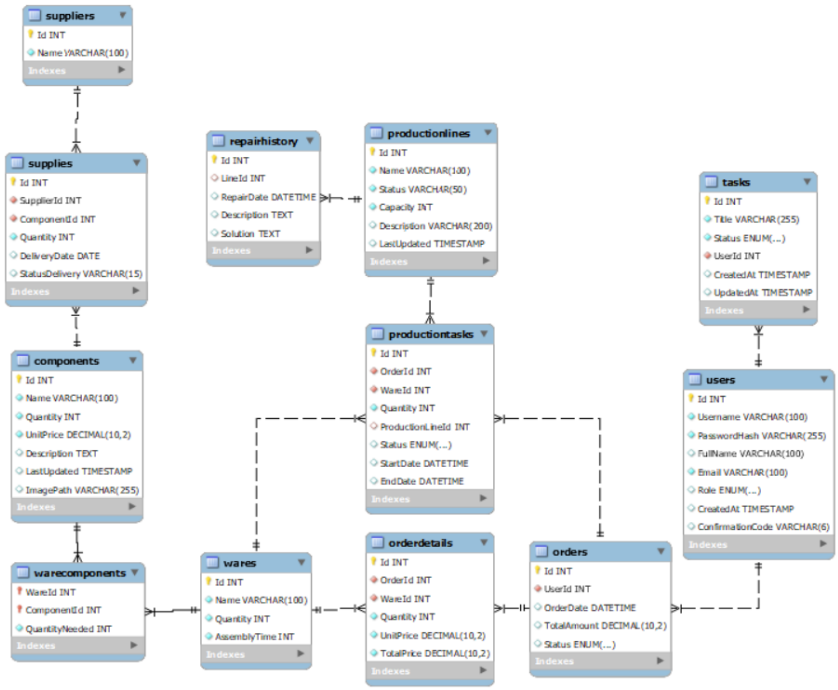


Рисунок 2.3 – Физическая схема данных

На таблицах 2.1 – 2.12 описаны сущности базы данных, их атрибуты и используемые типы данных с кратким описанием.

Таблица 2.1 – описание сущности «Пользователи»

Название	Описание	Размер	Тип
id	Идентификационный номер пользователя	Целочисленный	Численный
Username	Имя пользователя	100	Текстовый
PasswordHash	Хеш пароля	255	Текстовый
FullName	Полное имя	100	Текстовый
Email	Электронная почта	100	Текстовый
Role	Роль(Admin/User)	2 значения	Перечисление
CreatedAt	Дата создания записи	Краткий	Дата и время
ConfirmationCode	Код подтверждения	6	Текстовый

Таблица 2.2 – описание сущности «Компоненты»

Название	Описание	Размер	Тип
id	Идентификационный номер компонента	Целочисленный	Численный
Name	Название компонента	100	Текстовый
Quantity	Количество	Целочисленный	Числовой

UnitPrice	Цена за единицу	(10, 2)	Десятичный
Description	Описание компонента	-	Текстовый
LastUpdated	Дата обновления	Краткий	Дата и время

Таблица 2.3 – описание сущности «Поставщики»

Название	Описание	Размер	Тип
Id	Идентификационный номер поставщика	Целочисленный	Числовой
Name	Имя поставщика	100	Текстовый

Таблица 2.4 – описание сущности «Товары»

Название	Описание	Размер	Тип
Id	Идентификационный номер товара	Целочисленный	Числовой
Name	Название товара	100	Текстовый

Таблица 2.5 – описание сущности «Поставки»

Название	Описание	Размер	Тип
Id	Идентификационный номер поставки	Целочисленный	Числовой
SupplierId	Идентификационный номер поставщика	Целочисленный	Числовой
WareId	Идентификационный номер товара	Целочисленный	Числовой
ComponentId	Идентификационный номер компонента	Целочисленный	Числовой
Quantity	Количество	Целочисленный	Числовой
DeliveryDate	Дата поставки	Краткий	Дата
StatusDelivery	Статус поставки	15	Текстовый

Таблица 2.6 – описание сущности «Производственные линии»

Название	Описание	Размер	Тип
Id	Идентификационный номер производственной линии	Целочисленный	Числовой
Name	Название линии	100	Текстовый
Status	Статус линии	50	Текстовый
Capacity	Производственная мощность	Целочисленный	Числовой
Description	Описание линии	200	Текстовый
LastUpdated	Дата последнего обновления	Краткий	Дата и время

Таблица 2.7 – описание сущности «История починок»

Название	Описание	Размер	Тип
Id	Идентификационный номер починки	Целочисленный	Числовой
LineId	Ссылка на производственную линию	Целочисленный	Числовой
RepairDate	Дата ремонта	Краткий	Дата и время
Description	Описание ремонта	-	Текстовый

Solution	Решение проблемы	-	Текстовый

Таблица 2.8 – описание сущности «Заказы»

Название	Описание	Размер	Тип
Id	Идентификационный номер заказа	Целочисленный	Числовой
UserId	Ссылка на пользователя	Целочисленный	Числовой
OrderDate	Дата заказа	Краткий	Дата и время
TotalAmount	Общая сумма заказа	(10, 2)	Десятичный
Status	Статус заказов ('Pending', 'WaitingForComponents', 'InProduction','Completed', 'Cancelled')	5 значений	Перечисление

Таблица 2.9 – описание сущности «Детали заказов»

Название	Описание	Размер	Тип
Id	Идентификационный номер записи	Целочисленный	Числовой
OrderId	Ссылка на заказ	Целочисленный	Числовой
WareId	Ссылка на товар	Целочисленный	Числовой
Quantity	Количество	Целочисленный	Числовой
UnitPrice	Цена за единицу	(10, 2)	Десятичный
TotalPrice	Общая стоимость	(10, 2)	Десятичный

Таблица 2.10 – описание сущности «Производственные задачи»

Название	Описание	Размер	Тип
Id	Идентификационный номер записи	Целочисленный	Числовой
OrderId	Ссылка на заказ	Целочисленный	Числовой
WareId	Ссылка на товар	Целочисленный	Числовой
Quantity	Количество	Целочисленный	Числовой
ProductionLineId	Ссылка на производственную линию	Целочисленный	Числовой
Status	Статус заказов ('Queued', InProgress, 'Completed')	3 значения	Перечисление
StartDate	Начальная дата	Краткий	Дата и время
EndDate	Конечная дата	Краткий	Дата и время

Таблица 2.11 – описание сущности «Задачи»

Название	Описание	Размер	Тип
Id	Идентификационный номер записи	Целочисленный	Числовой
Title	Название	255	Текстовый
Status	Статус заказов ('ToDo', 'InProgress', 'Done')	3 значения	Перечисление
Quantity	Количество	Целочисленный	Числовой
UserId	Ссылка на пользователя	Целочисленный	Числовой
CreatedAt	Дата создания	Краткий	Дата и время
UpdatedAt	Дата обновления	Краткий	Дата и время

Таблица 2.12 – описание сущности «Компоненты товаров»

Название	**Описание**	**Размер**	**Тип**
WareId	Идентификационный номер товара	Целочисленный	Числовой
ComponentId	Идентификационный номер компонента	Целочисленный	Числовой
QuantityNeeded	Необходимое количество	Целочисленный	Числовой

2.3 Конструирование пользовательского интерфейса

При разработке пользовательского интерфейса системы CompWare были применены следующие принципы удобства использования:

1) Минимализм – интерфейс исключает избыточные элементы, фокусируясь на ключевых функциях.

2) Консистентность – все экраны системы выдержаны в едином стиле (одинаковые кнопки, шрифты, цветовая схема).

3) Интуитивность – расположение элементов соответствует логике работы.

Интерфейс системы использует следующие стилевые решения:

1) Цветовая палитра:

Основной цвет: синий (2d3250) – используется для фона страниц регистрации и авторизации; белый (FFFFFF) для основных форм и черный (000000) для темной темы.

Цвет панели: FF2D3250 для светлой темы и светло-серый (FF808080) для темной темы.

Текст: черный (000000) для основного контента, белый (FFFFFF) для текста на цветных кнопках.

2) Типографика:

Заголовки: шрифт Segoe UI, 18pt, полужирный.

Основной текст: шрифт Segoe UI, 14pt, обычный.

3) Сетка и выравнивание:

Все элементы интерфейса выровнены по сетке с отступами 16px.

Группы связанных элементов визуально отделены.

4) Функциональные компоненты интерфейса

На таблице 2.13 представлены компоненты интерфейса.

Таблица 2.13 – Функциональные компоненты интерфейса

Элемент	Назначение	Примеры использования	Особенности
Кнопки	Инициирование действий	"Войти", "Регистрация"	Цветовое выделение
Поля ввода	Ввод текстовых данных	Логин, Пароль	Маскировка пароля
Таблицы	Структурированное отображение данных	Таблица компонентов	Сортировка по столбцам
Вкладки	Переключение между разделами	Заказ на производство/заказ поставщику	Без перезагрузки страницы
Меню	Навигация по системе	Главное меню	Иерархическая структура
Сообщения	Информирование пользователя	Приветственные сообщения	Контекстное отображение

Представленные визуальные компоненты образуют целостную систему взаимодействия, где каждый элемент имеет четкое функциональное назначение:

- интерактивные элементы обеспечивают управление системой;
- информационные компоненты предоставляют данные;
- навигационные элементы организуют перемещение по интерфейсу;
- специальные элементы решают узкоспециализированные задачи.

Все компоненты выдержаны в едином стиле, что обеспечивает последовательность пользовательского опыта.

5) Примеры реализации интерфейса

Для наглядности ниже приведены примеры интерфейсных решений.

На рисунке 2.4 представлена форма авторизации.

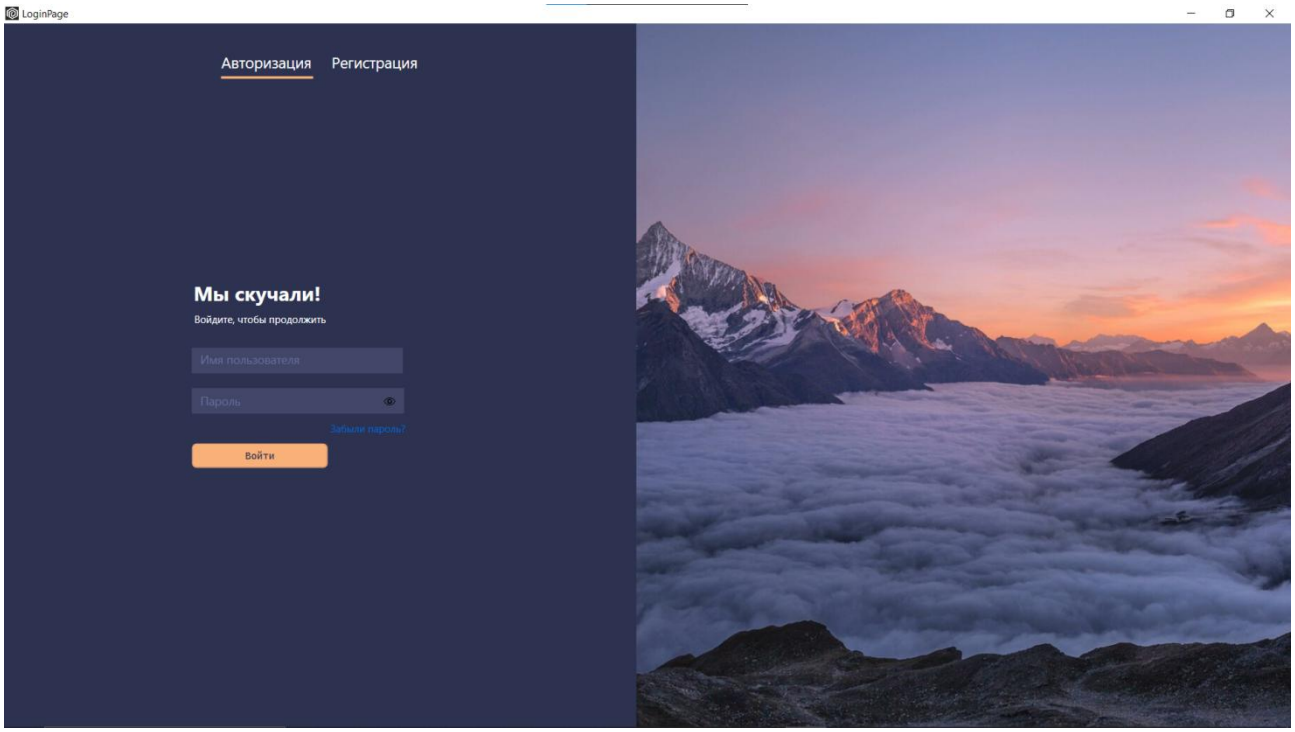


Рисунок 2.4 – Форма авторизации

На рисунке 2.5 представлена заставка.

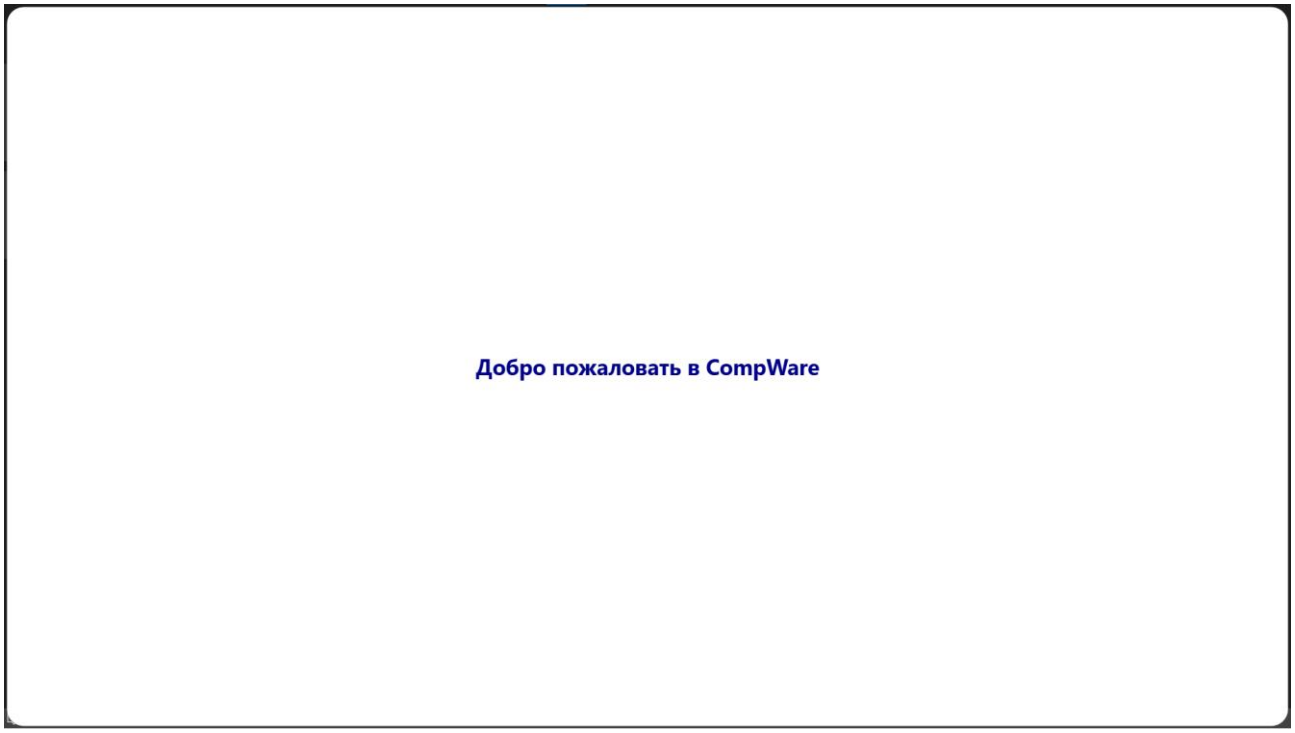


Рисунок 2.5 – Заставка

На рисунке 2.6 представлена главная форма.

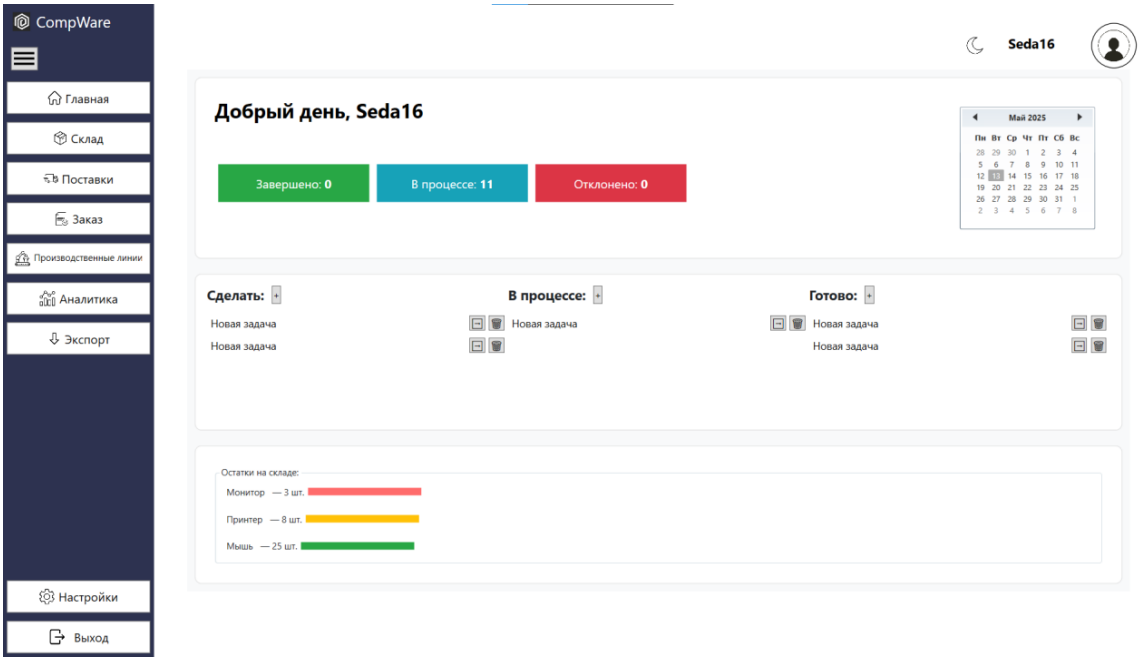


Рисунок 2.6 – Главная форма

На рисунке 2.7 представлена форма компонентов.

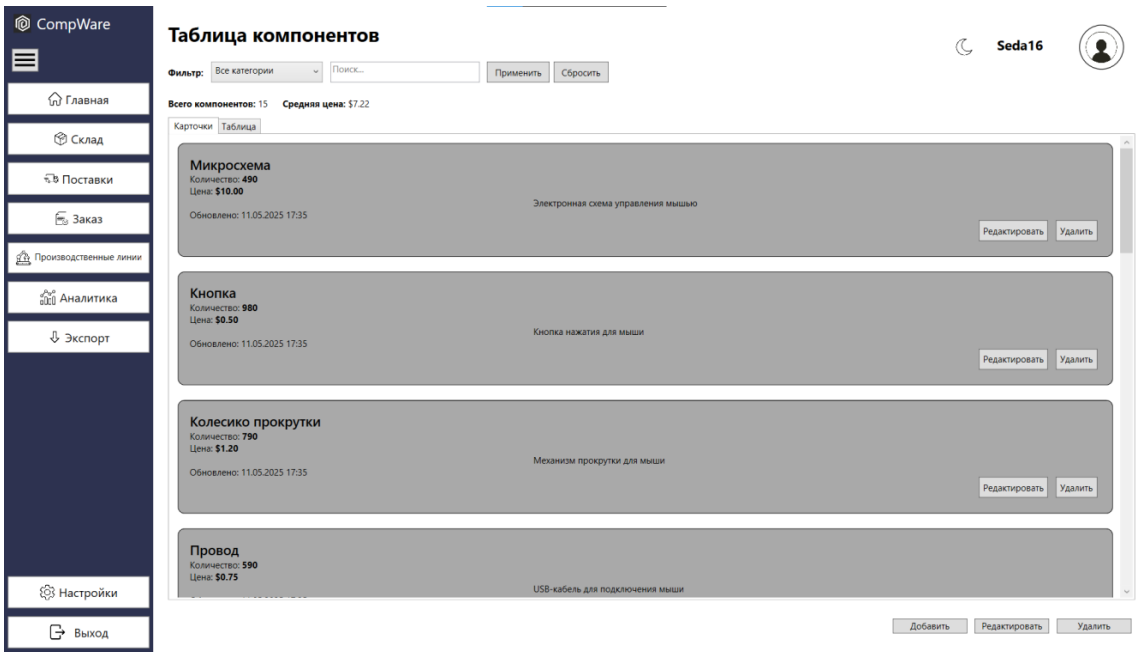


Рисунок 2.7 – Форма компонентов

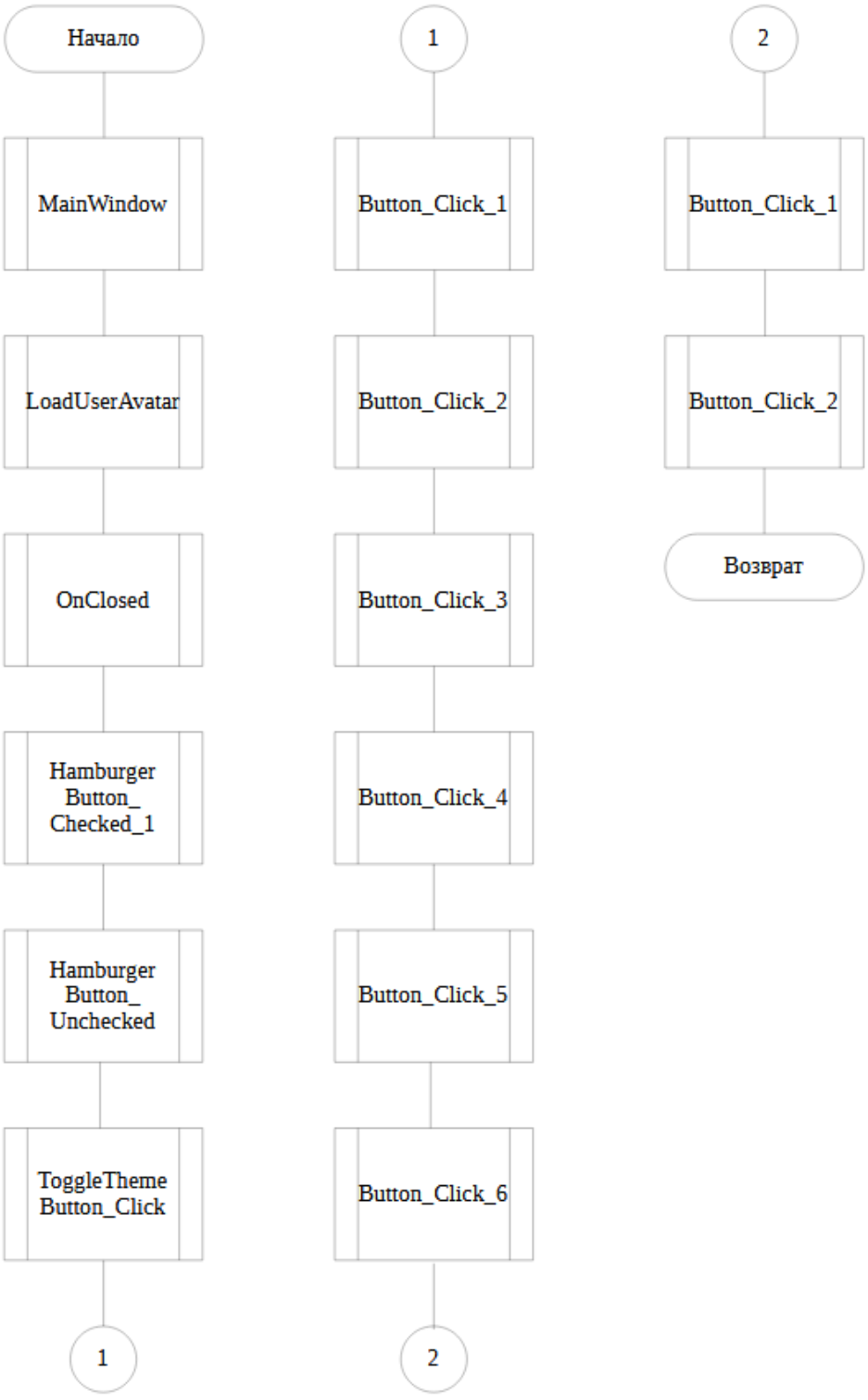
Представленный пользовательский интерфейс системы CompWare разработан с учетом современных принципов юзабилити и обеспечивает:

- простоту освоения для новых пользователей;
- визуальную согласованность всех элементов;
- понятную обратную связь о действиях системы.

Приведенные примеры интерфейсных решений демонстрируют ключевые особенности реализации UI.

2.4 Схемы алгоритма программы и подпрограмм

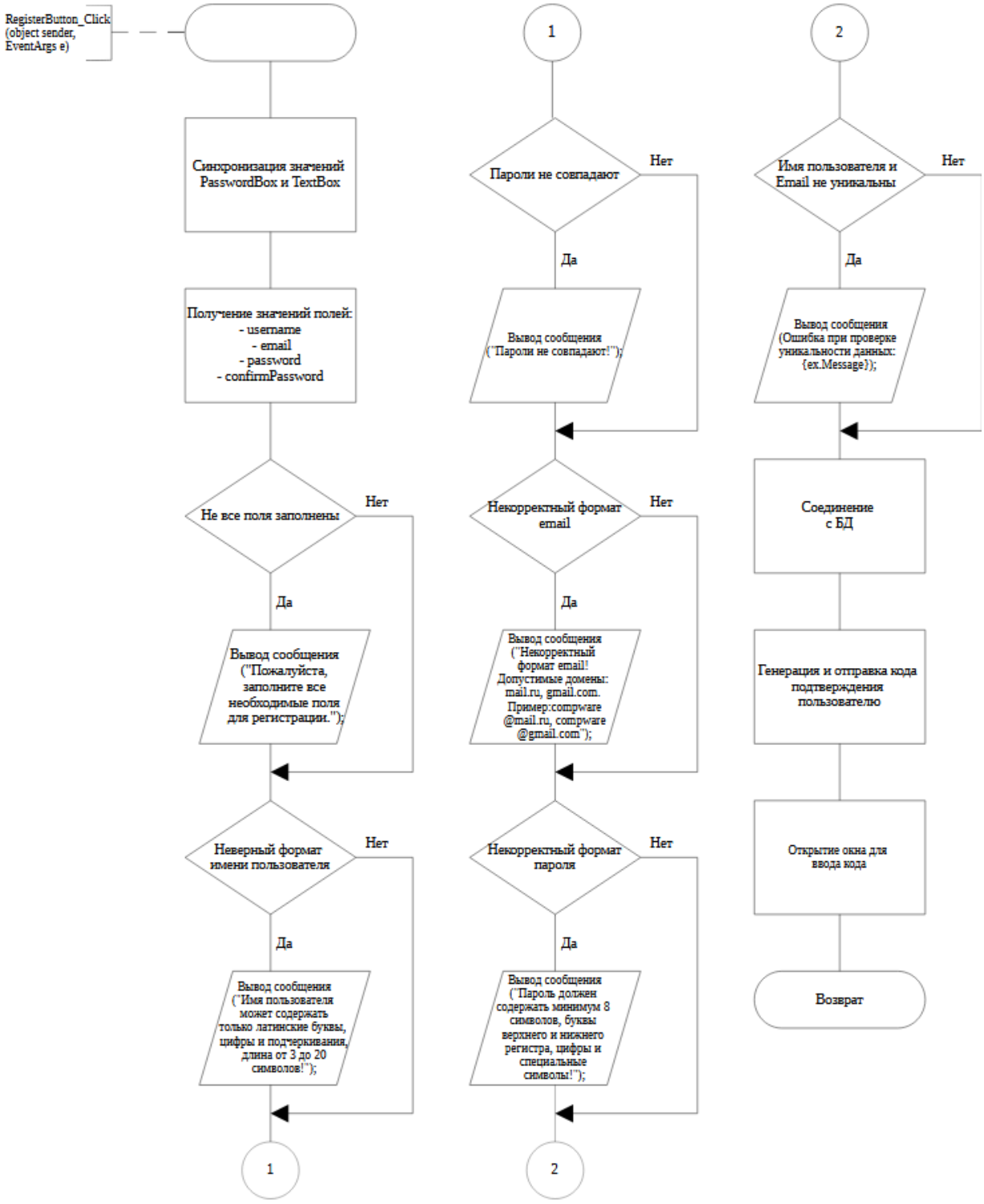
2.4.1 Схема алгоритма основной программы



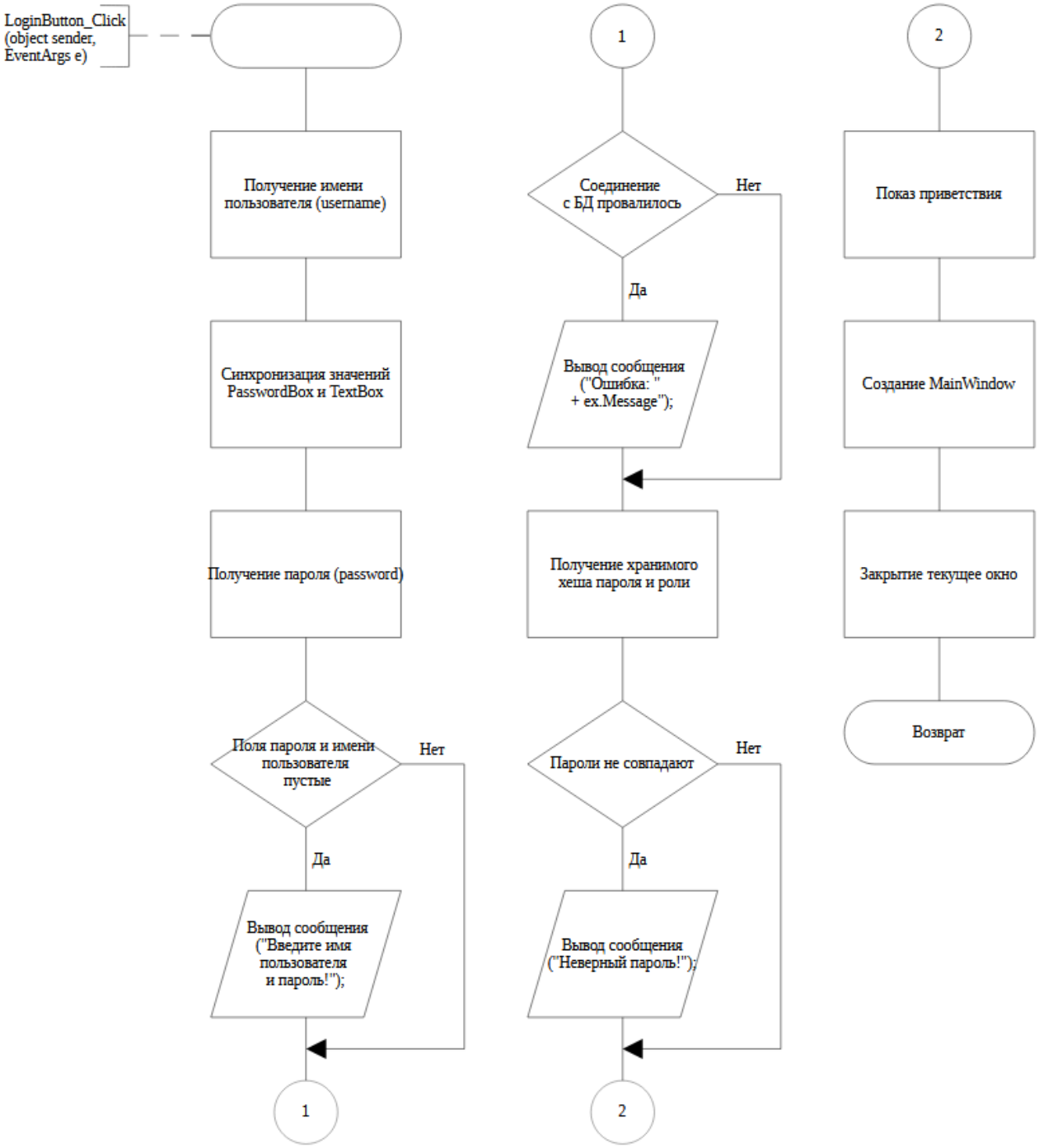
Назначение:

- конструктор MainWindow(string role) – инициализирует главное окно, загружает аватар пользователя и сохраняет его роль;
- LoadUserAvatar() – загружает аватар пользователя из сохраненного пути или устанавливает изображение по умолчанию;
- OnClosed(EventArgs e) – отписывается от события обновления аватара при закрытии окна;
- HamburgerButton_Checked_1(object sender, RoutedEventArgs e) – анимирует раскрытие боковой панели до ширины 200 пикселей;
- HamburgerButton_Unchecked(object sender, RoutedEventArgs e) – анимирует сворачивание боковой панели до ширины 53 пикселя;
- ToggleThemeButton_Click(object sender, RoutedEventArgs e) – переключает между светлой и темной темой оформления приложения;
- Button_Click_1(object sender, RoutedEventArgs e) – очищает главную сетку и добавляет информацию о складе.;
- Button_Click_3(object sender, RoutedEventArgs e) – открывает страницу управления складом (StoragePage);
- Button_Click_4(object sender, RoutedEventArgs e) – открывает страницу поставок (SuppliesPage):
- Button_Click_5(object sender, RoutedEventArgs e) – открывает страницу производственных линий (ProductionLinesPage) с учетом роли пользователя;
- Button_Click_6(object sender, RoutedEventArgs e) – закрывает приложение;
- Button_Click_7(object sender, RoutedEventArgs e) – открывает страницу настроек (SettingsPage);
- Button_Click_2(object sender, RoutedEventArgs e) – возвращает на главную страницу (MainPage);
- Button_Click_8(object sender, RoutedEventArgs e) – открывает страницу заказов (order).

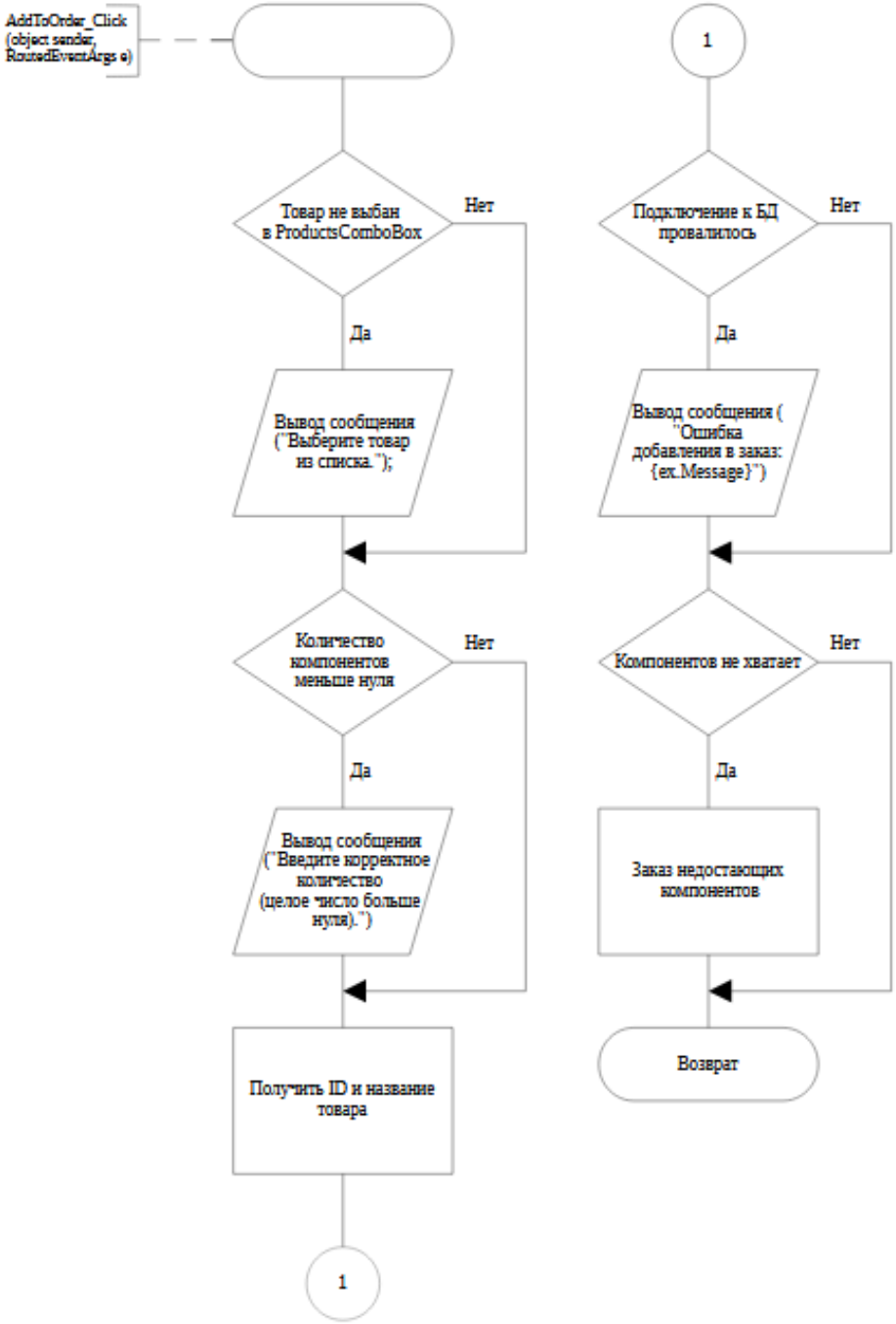
2.4.2 Схема алгоритма подпрограммы RegisterButton_Click



2.4.3 Схема алгоритма подпрограммы LoginButton_Click



2.4.4 Схема алгоритма подпрограммы AddToOrder_Click



2.5 Отладка и тестирование программы

2.5.1 Отладка программы

Отладка представляет собой систематический процесс выявления, анализа и устранения ошибок в программном коде. Данный процесс является неотъемлемой частью жизненного цикла разработки программного обеспечения и направлен на обеспечение корректной работы программных систем.

Основные этапы отладки:

1) Выявление ошибки

Ошибки могут быть обнаружены в ходе тестирования, code review или эксплуатации программного обеспечения. Важным аспектом является точное описание условий, при которых возникает ошибка, включая входные данные и состояние системы.

2) Локализация проблемы

Для определения источника ошибки используются специализированные инструменты: отладчики, системы логирования, мониторинга производительности. Применяются методы пошагового выполнения кода, анализа значений переменных и стека вызовов.

3) Анализ причин возникновения

Установление первопричины ошибки осуществляется путем исследования:

- корректности алгоритмов;
- обработки граничных условий;
- взаимодействия компонентов системы;
- соответствия фактического поведения программы ожидаемому.

4) Устранение ошибки

Внесение изменений в код должно сопровождаться:

- минимальным воздействием на работоспособность системы;
- сохранением исходной функциональности;
- соответствием стандартам кодирования.

5) Верификация исправления

Проводится комплексное тестирование:

- модульное тестирование исправленного компонента;

- регрессионное тестирование для проверки отсутствия побочных эффектов;
- интеграционное тестирование взаимодействий с другими компонентами.

6) Инструментальные средства:

Современные среды разработки (IDE) предоставляют интегрированные средства отладки, включая:

- точки останова (breakpoints);
- пошаговое выполнение (step execution);
- инспекцию переменных (variable watch);
- анализ потока управления (control flow analysis).

Эффективные практики:

- ведение подробного журнала изменений;
- использование систем контроля версий для отслеживания модификаций;
- применение модульного и интеграционного тестирования;
- регулярный анализ кода на потенциальные уязвимости.

Отладка является критически важным процессом, от качества выполнения которого напрямую зависит надежность, безопасность и производительность программного обеспечения. Грамотная организация данного процесса позволяет существенно сократить временные затраты на разработку и повысить качество конечного продукта.

Примеры отладки кода:

Пример 1: Устранение дублирования кода

В коде страницы управления производственными линиями (ProductionLinesPage.xaml.cs) есть несколько методов, которые выполняют схожие операции с базой данных. Например, методы StopLineButton_Click и StartLineButton_Click обновляют статус производственной линии. Эти методы можно объединить в один, чтобы уменьшить количество кода и улучшить читаемость.

Неоптимизированный код:

```
private void StopLineButton_Click(object sender, RoutedEventArgs e)
{ if (ProductionLinesDataGrid.SelectedItem is ProductionLine selectedLine)
   { using (var connection = new MySqlConnection(ConnectionString))
      { connection.Open();
        var command = new MySqlCommand(
```

```
                "UPDATE ProductionLines SET Status = 'Остановлена', LastUpdated =
@LastUpdated WHERE Id = @Id", connection);
                command.Parameters.AddWithValue("@Id", selectedLine.Id);
                command.Parameters.AddWithValue("@LastUpdated", DateTime.Now);
                command.ExecuteNonQuery();}
            LoadProductionLines();}
        else
        { MessageBox.Show("Выберите производственную линию для остановки!"); }
    }
    private void StartLineButton_Click(object sender, RoutedEventArgs e)
    {if (ProductionLinesDataGrid.SelectedItem is ProductionLine selectedLine)
        { using (var connection = new MySqlConnection(ConnectionString))
            { connection.Open();
                var command = new MySqlCommand(
                    "UPDATE ProductionLines SET Status = 'Работает', LastUpdated =
@LastUpdated WHERE Id = @Id", connection);
                command.Parameters.AddWithValue("@Id", selectedLine.Id);
                command.Parameters.AddWithValue("@LastUpdated", DateTime.Now);
                command.ExecuteNonQuery();      }
            LoadProductionLines();   }
        else
        { MessageBox.Show("Выберите производственную линию для запуска!");   }
    }
    Исправленный код:
    private void UpdateLineStatus(string status)
    { if (ProductionLinesDataGrid.SelectedItem is ProductionLine selectedLine)
        {  using (var connection = new MySqlConnection(ConnectionString))
            { connection.Open();
                var command = new MySqlCommand(
                    "UPDATE ProductionLines SET Status = @Status, LastUpdated =
@LastUpdated WHERE Id = @Id", connection);
                command.Parameters.AddWithValue("@Id", selectedLine.Id);
                command.Parameters.AddWithValue("@Status", status);
                command.Parameters.AddWithValue("@LastUpdated", DateTime.Now);
                command.ExecuteNonQuery();}
```

```
            LoadProductionLines();    }
        else
        { MessageBox.Show("Выберите производственную линию!"); }
    }
    private void StopLineButton_Click(object sender, RoutedEventArgs e)
    {  UpdateLineStatus("Остановлена");}
    private void StartLineButton_Click(object sender, RoutedEventArgs e)
    {    UpdateLineStatus("Работает");}
```

Пример 2: Устранение лишних подключений к базе данных

В методе LoadProductionLines происходит открытие и закрытие соединения с базой данных. Если этот метод вызывается несколько раз, это может привести к излишней нагрузке на базу данных. Можно оптимизировать этот процесс, используя кэширование данных.

Неоптимизированный код:

```
    private void LoadProductionLines()
    { ProductionLines.Clear();
        using (var connection = new MySqlConnection(ConnectionString))
        {connection.Open();
            var command = new MySqlCommand("SELECT * FROM ProductionLines", connection);
            var reader = command.ExecuteReader();
            while (reader.Read())
            {ProductionLines.Add(new ProductionLine
                { Id = reader.GetInt32("Id"),
                    Name = reader.GetString("Name"),
                    Status = reader.GetString("Status"),
                    Capacity = reader.GetInt32("Capacity"),
                    Description = reader.GetString("Description"),
                    LastUpdated = reader.GetDateTime("LastUpdated")
                });  }
        }
    }
```

Исправленный код:

```
    private List<ProductionLine> cachedProductionLines;
    private void LoadProductionLines()
```

```
{    if (cachedProductionLines == null)
   {cachedProductionLines = new List<ProductionLine>();
      using (var connection = new MySqlConnection(ConnectionString))
      { connection.Open();
         var command = new MySqlCommand("SELECT * FROM ProductionLines", connection);
         var reader = command.ExecuteReader();
         while (reader.Read())
         {  cachedProductionLines.Add(new ProductionLine
            { Id = reader.GetInt32("Id"),
               Name = reader.GetString("Name"),
               Status = reader.GetString("Status"),
               Capacity = reader.GetInt32("Capacity"),
               Description = reader.GetString("Description"),
               LastUpdated = reader.GetDateTime("LastUpdated")
            }); }
      }
   }
   ProductionLines.Clear();
   foreach (var line in cachedProductionLines)
   {ProductionLines.Add(line);  }
}
```

Пример 3: Создание подпрограмм

Методы AddLineButton_Click, EditLineButton_Click и DeleteLineButton_Click можно объединить в один метод, чтобы избежать дублирования логики.

Неоптимизированный код:

```
private void AddLineButton_Click(object sender, RoutedEventArgs e)
{var addWindow = new EditProductionLineWindow();
   if (addWindow.ShowDialog() == true)
   { var newLine = addWindow.ProductionLine;
      using (var connection = new MySqlConnection(ConnectionString))
      {connection.Open();
         var command = new MySqlCommand(
            "INSERT INTO ProductionLines (Name, Status, Capacity, Description, LastUpdated) " +
```

```
                    "VALUES (@Name, @Status, @Capacity, @Description, @LastUpdated)",
connection);
                command.Parameters.AddWithValue("@Name", newLine.Name);
                command.Parameters.AddWithValue("@Status", newLine.Status);
                command.Parameters.AddWithValue("@Capacity", newLine.Capacity);
                command.Parameters.AddWithValue("@Description", newLine.Description);
                command.Parameters.AddWithValue("@LastUpdated", newLine.LastUpdated);
                command.ExecuteNonQuery();  }
            LoadProductionLines();}
        }
        private void EditLineButton_Click(object sender, RoutedEventArgs e)
        {if (ProductionLinesDataGrid.SelectedItem is ProductionLine selectedLine)
           {var editWindow = new EditProductionLineWindow(selectedLine);
            if (editWindow.ShowDialog() == true)
            { var updatedLine = editWindow.ProductionLine;
              using (var connection = new MySqlConnection(ConnectionString))
              { connection.Open();
                var command = new MySqlCommand(
                    "UPDATE ProductionLines SET Name = @Name, Status = @Status,
Capacity = @Capacity, " +
                    "Description = @Description, LastUpdated = @LastUpdated WHERE Id =
@Id", connection);
                command.Parameters.AddWithValue("@Id", updatedLine.Id);
                command.Parameters.AddWithValue("@Name", updatedLine.Name);
                command.Parameters.AddWithValue("@Status", updatedLine.Status);
                command.Parameters.AddWithValue("@Capacity", updatedLine.Capacity);
                command.Parameters.AddWithValue("@Description",
updatedLine.Description);
                command.Parameters.AddWithValue("@LastUpdated",
updatedLine.LastUpdated);
                command.ExecuteNonQuery(); }
              LoadProductionLines();
            }
           }else
           {MessageBox.Show("Выберите производственную линию для редактирования!");
```

```
        }
        private void DeleteLineButton_Click(object sender, RoutedEventArgs e)
        { if (ProductionLinesDataGrid.SelectedItem is ProductionLine selectedLine)
            { using (var connection = new MySqlConnection(ConnectionString))
                { connection.Open();
                    var command = new MySqlCommand("DELETE FROM ProductionLines WHERE Id = @Id", connection);
                    command.Parameters.AddWithValue("@Id", selectedLine.Id);
                    command.ExecuteNonQuery();}
                ProductionLines.Remove(selectedLine); }
            else
            { MessageBox.Show("Выберите производственную линию для удаления!"); }
        }
```

Исправленный код:

```
        private void HandleLineOperation(string operation, ProductionLine line = null)
        { if (operation == "Add")
            { var addWindow = new EditProductionLineWindow();
                if (addWindow.ShowDialog() == true)
                { line = addWindow.ProductionLine;
                    operation = "Insert"; }
            }
            if (line != null)
            { using (var connection = new MySqlConnection(ConnectionString))
                {connection.Open();
                    MySqlCommand command;
                    if (operation == "Insert")
                    {command = new MySqlCommand(
                            "INSERT INTO ProductionLines (Name, Status, Capacity, Description, LastUpdated) " +
                            "VALUES (@Name, @Status, @Capacity, @Description, @LastUpdated)", connection);
                    }
                    else if (operation == "Update"){
                        command = new MySqlCommand(
```

```
                    "UPDATE ProductionLines SET Name = @Name, Status = @Status, Capacity = @Capacity, " +
                    "Description = @Description, LastUpdated = @LastUpdated WHERE Id = @Id", connection);
                    command.Parameters.AddWithValue("@Id", line.Id); }
                else if (operation == "Delete")
                { command = new MySqlCommand("DELETE FROM ProductionLines WHERE Id = @Id", connection);
                    command.Parameters.AddWithValue("@Id", line.Id);  }
                else
                { return; }
                if (operation != "Delete")  {
                    command.Parameters.AddWithValue("@Name", line.Name);
                    command.Parameters.AddWithValue("@Status", line.Status);
                    command.Parameters.AddWithValue("@Capacity", line.Capacity);
                    command.Parameters.AddWithValue("@Description", line.Description);
                    command.Parameters.AddWithValue("@LastUpdated", line.LastUpdated); }
                command.ExecuteNonQuery();}
            LoadProductionLines();}
        else
        {MessageBox.Show("Выберите производственную линию!"); }
    }
    private void AddLineButton_Click(object sender, RoutedEventArgs e)
    { HandleLineOperation("Add");}
    private void EditLineButton_Click(object sender, RoutedEventArgs e){
        if (ProductionLinesDataGrid.SelectedItem is ProductionLine selectedLine) {
            HandleLineOperation("Update", selectedLine);  }
        else  {
            MessageBox.Show("Выберите производственную линию для редактирования!"); }
    }
    private void DeleteLineButton_Click(object sender, RoutedEventArgs e){
        if (ProductionLinesDataGrid.SelectedItem is ProductionLine selectedLine) {
            HandleLineOperation("Delete", selectedLine);    }
        else {
```

```
        MessageBox.Show("Выберите производственную линию для удаления!"); }
}
```

2.5.2 Тестирование программы

Тестирование программного обеспечения представляет собой систематический процесс проверки корректности работы программы путем сравнения фактических результатов выполнения с ожидаемыми значениями. Данный процесс осуществляется на специально разработанном наборе тестовых случаев, позволяющем эффективно выявлять потенциальные проблемы и подтверждать соответствие продукта установленным требованиям.

Основные цели и задачи тестирования:

1) Проверка соответствия техническим требованиям:
   - подтверждение выполнения всех заявленных функций;
   - соответствие спецификациям проекта;
   - проверка граничных условий и исключительных ситуаций.
2) Обеспечение качества продукта:
   - демонстрация стабильности работы системы;
   - подтверждение готовности к эксплуатации;
   - обеспечение надежности и отказоустойчивости.
3) Выявление и устранение дефектов:
   - обнаружение ошибок на ранних стадиях разработки;
   - снижение затрат на последующее исправление;
   - минимизация рисков при промышленной эксплуатации.

Тестирование является критически важным процессом на всех этапах создания программного обеспечения:

- на этапе модульной разработки позволяет проверять отдельные компоненты;
- при интеграции обеспечивает корректность взаимодействия модулей;
- в финальной стадии подтверждает соответствие системы бизнес-требованиям.

Ключевые преимущества качественного тестирования:

- снижение стоимости разработки за счет раннего обнаружения ошибок;
- повышение надежности и стабильности конечного продукта;
- увеличение удовлетворенности пользователей;

- сокращение затрат на техническую поддержку;
- минимизация рисков при внедрении системы.

Тестирование представляет собой неотъемлемую часть процесса разработки программного обеспечения, непосредственно влияющую на качество, надежность и успешность конечного продукта. Грамотно организованный процесс тестирования позволяет создавать программные системы, соответствующие как техническим требованиям, так и ожиданиям пользователей.

В таблицах 2.14 – 2.17 представлены позитивные и негативные тест кейсы для программы.

В таблицах 2.14 – 2.15 представлены позитивные тест кейсы.

Таблица 2.14 – Позитивный тест кейс для процесса «Регистрация»

Действие	Ожидаемый результат	Результат теста
1. Открываем форму «Регистрация пользователей»	– Форма открыта; – Все поля по умолчанию пусты; – Отображаются серые подсказки в полях.	Выполнено
2. Заполняем поля формы – Имя пользователя = Seda16 – Электронный адрес = sedakarenovna2005@mail.ru – Пароль = Seda2005 – Повторно пароль = Seda2005	– Поля заполнены; – Подсказки поднялись наверх и подсвечиваются оранжевым цветом.	Выполнено
3. Нажимаем на кнопку «Зарегистрироваться»	– Данные проверяются на корректность; – Данные проверяются на отсутствие конфликтов; – Высылается сообщение на почту; – Подсказки отображаются при наведении; – Открывается форма ввода кода подтверждения.	Выполнено

Таблица 2.15 – Позитивный тест кейс для процесса «Авторизация»

Действие	Ожидаемый результат	Результат теста
1. Открываем форму «Авторизация»	– Форма открыта; – Все поля по умолчанию пусты; – Кнопка «Войти» активна; – Кнопка «Забыли пароль?» активна и подсвечивается синим цветом; – Все подсказки отображаются.	Выполнено
2. Заполняем поля формы – Имя пользователя = Seda16 Пароль = Seda2005	– Поля заполнены; – Подсказки поднялись наверх и подсвечиваются оранжевым цветом.	Выполнено

3. Нажимаем кнопку «Войти»	– Данные проверяется на корректность; – Переход на главную форму.	Выполнено

В таблицах 2.16 – 2.17 представлены негативные тест кейсы.

Таблицы 2.15 – Негативный тест кейс для процесса «Регистрация»

Действие	Ожидаемый результат	Результат теста
1. Открываем форму «Регистрация пользователей»	– Форма открыта; – Все поля по умолчанию пусты; – Отображаются серые подсказки в полях; – Кнопка «Зарегистрироваться» неактивна.	Выполнено
2. Нажимаем «Зарегистрироваться»	Выводится сообщение: «Пожалуйста, заполните все необходимые поля для регистрации.».	Выполнено
3. Заполняем поля формы регистрации – Имя пользователя = Seda – Электронный адрес = sedakarenovna2005@mail.ru – Пароль = Seda2005 – Повторно пароль = Seda2005	Выводится сообщение: «Пользователь с таким именем уже зарегистрирован.»	Выполнено
3. Заполняем поля формы регистрации – Имя = Seda – Фамилия = Маргарян – Электронный адрес = sedakarenovna2005@mail.ru – Пароль = sed – Повторно пароль = sed	Выводится сообщение: «Пароль не соответствует данным требованиям: – должен содержать от 4 до 16 символов; – не должно быть символов из набора: * & { } | +; 5 – должны встречаться заглавные буквы; – должны встречаться цифры	Выполнено

Таблицы 2.16 – Негативный тест кейс для процесса «Авторизация»

Действие	Ожидаемый результат	Результат теста
1. Открываем форму «Авторизация»	– Форма открыта. – Все поля по умолчанию пусты. – Кнопка «Войти» активна. – Кнопка «Регистрация» активна. – Кнопка «Забыли пароль» активна и подсвечена синим цветом.	Выполнено
2. Нажимаем «Войти»	Выводится сообщение: «Пожалуйста, введите имя пользователя и пароль.».	Выполнено
3. Заполняем поля формы – Имя пользователя = Seda – Электронный адрес = sedakenovna2005@mail.ru – Пароль = Seda2005 – Повторно пароль = Seda2005	Выводится сообщение: «Неверное имя пользователя или пароль.».	Выполнено

2.5.3 Тестирование программы в нормальных условиях

В рамках тестирования в нормальных условиях была проведена комплексная проверка всех форм интерфейса, включая:

- корректность загрузки и отображения данных;
- логику переходов между формами;
- визуальное представление информации;
- функциональность интерактивных элементов (кнопок, полей ввода и других управляющих компонентов).

На рисунках 2.8 – 2.13 представлены результаты тестирования. Формы и контент на них прогрузился.

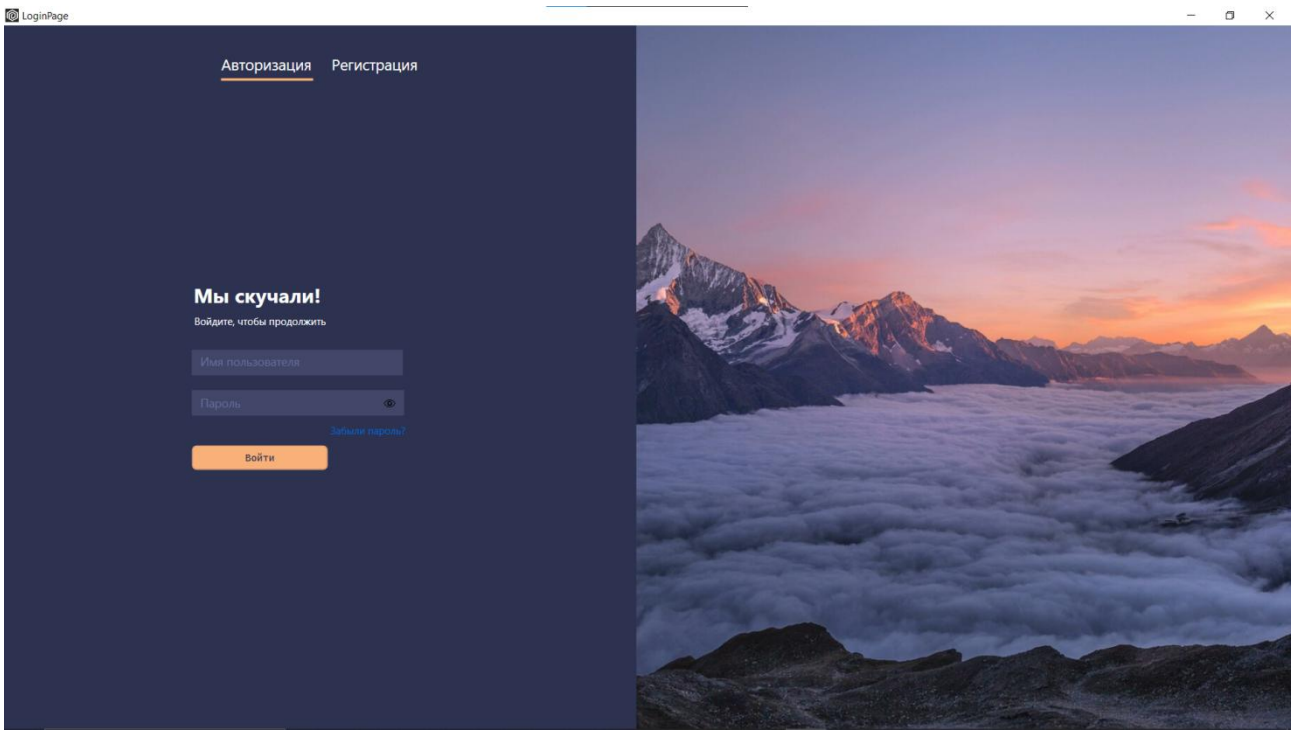


Рисунок 2.8 – Успешное отображение формы авторизации

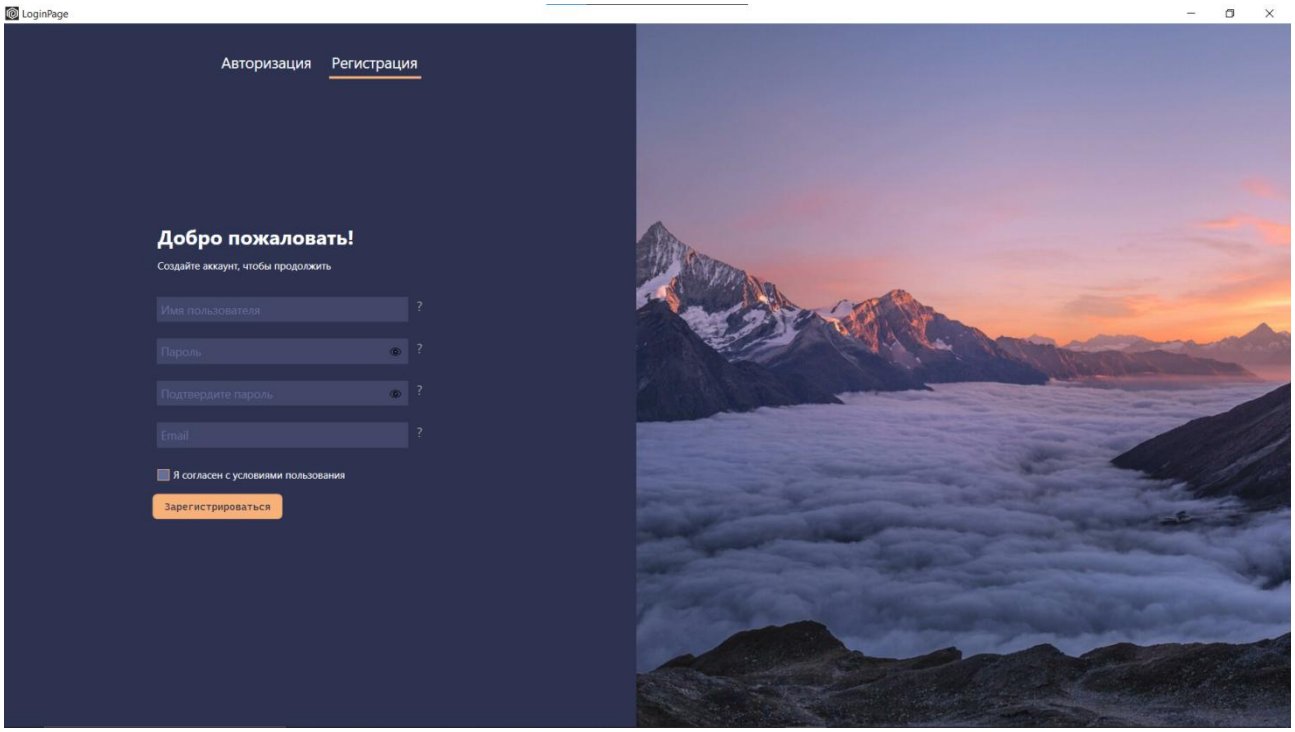


Рисунок 2.9 – Успешное отображение формы регистрации

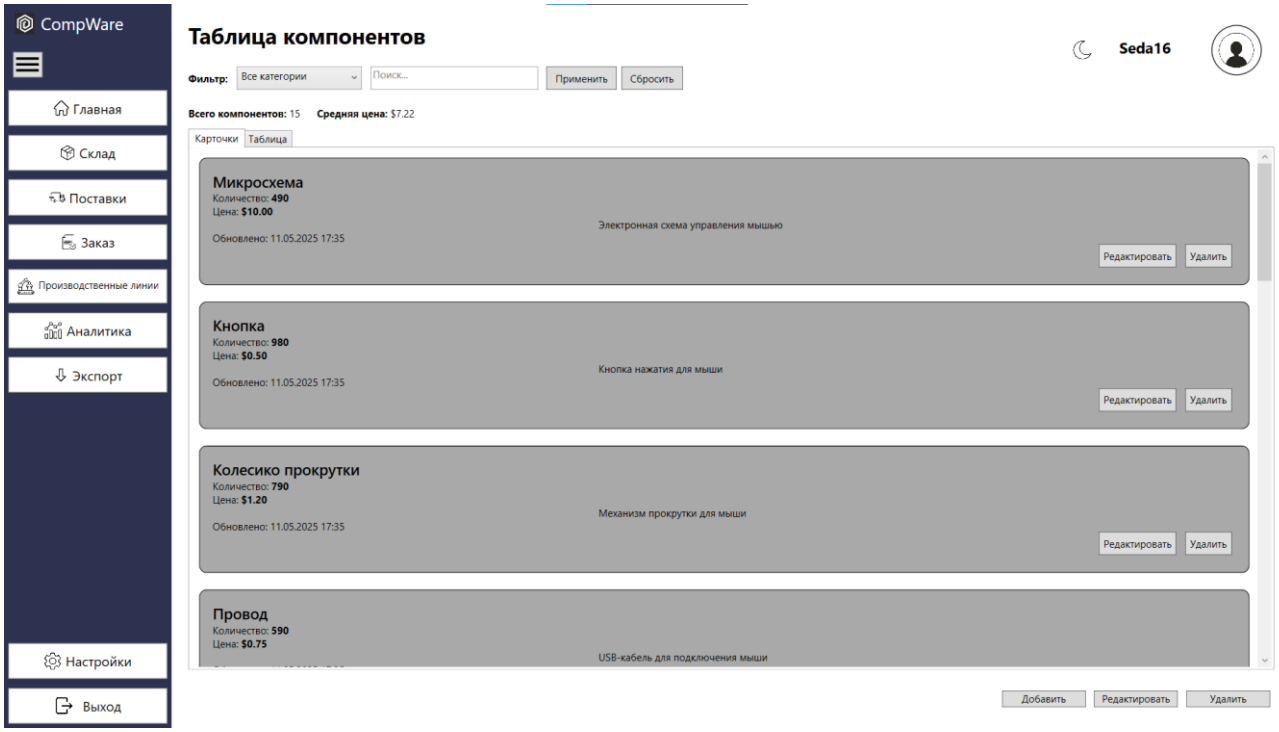


Рисунок 2.10 – Успешное отображение формы компонентов

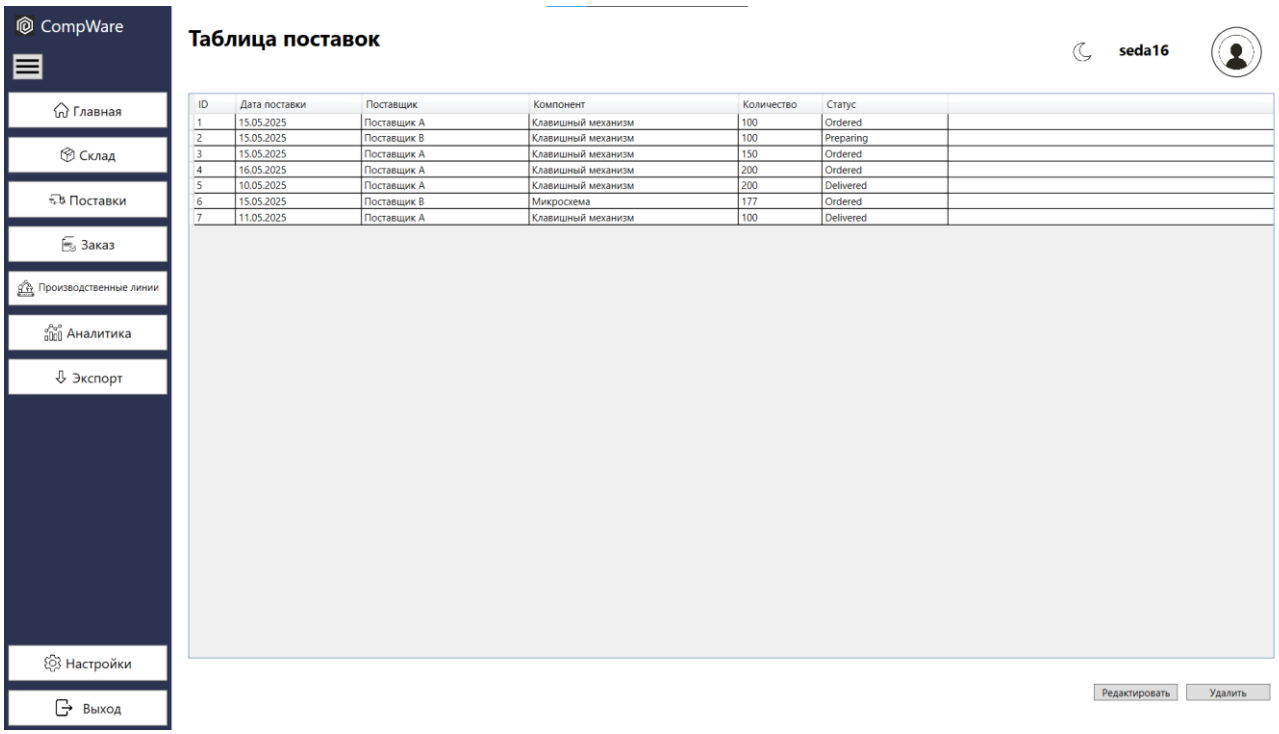


Рисунок 2.11 – Успешное отображение формы поставок

На рисунке 2.12 представлен вывод сообщения при вводе корректных данных для регистрации.

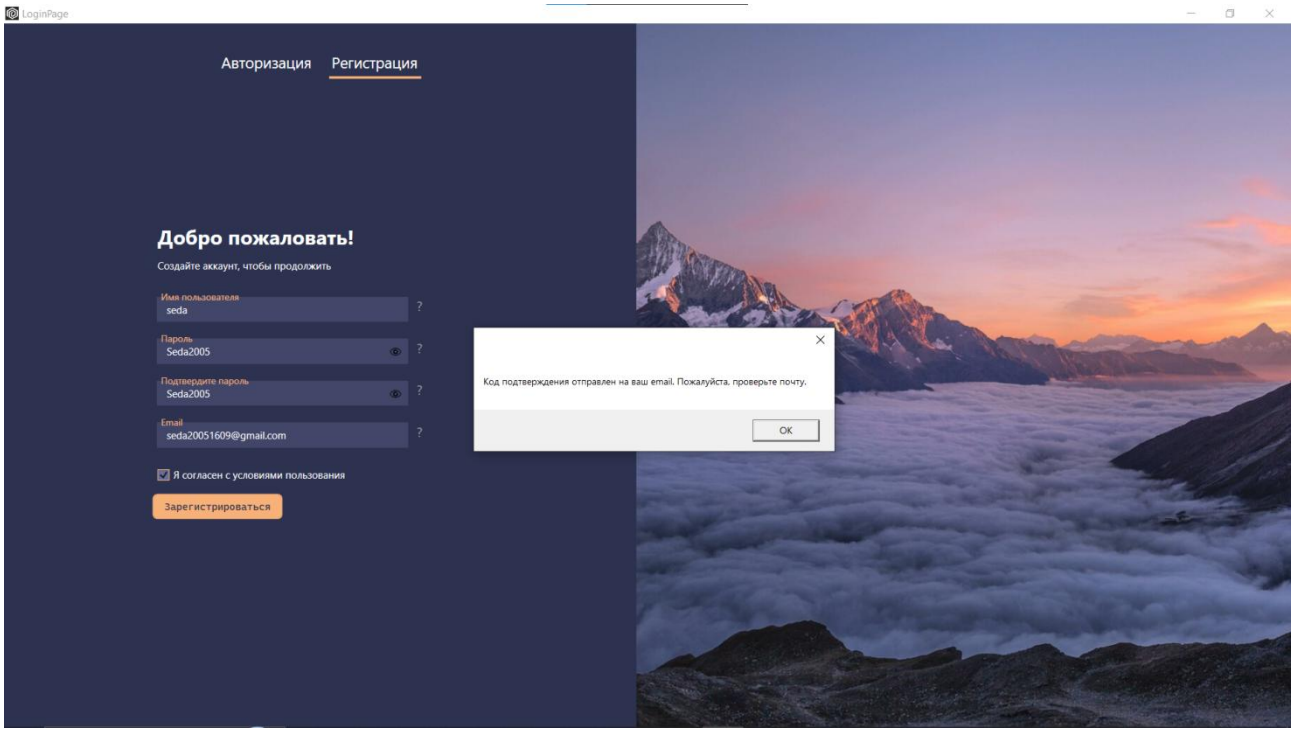


Рисунок 2.12 – Вывод сообщения при вводе корректных данных

На рисунках 2.13 представлены вывод сообщения при вводе корректных данных при создании заказа.

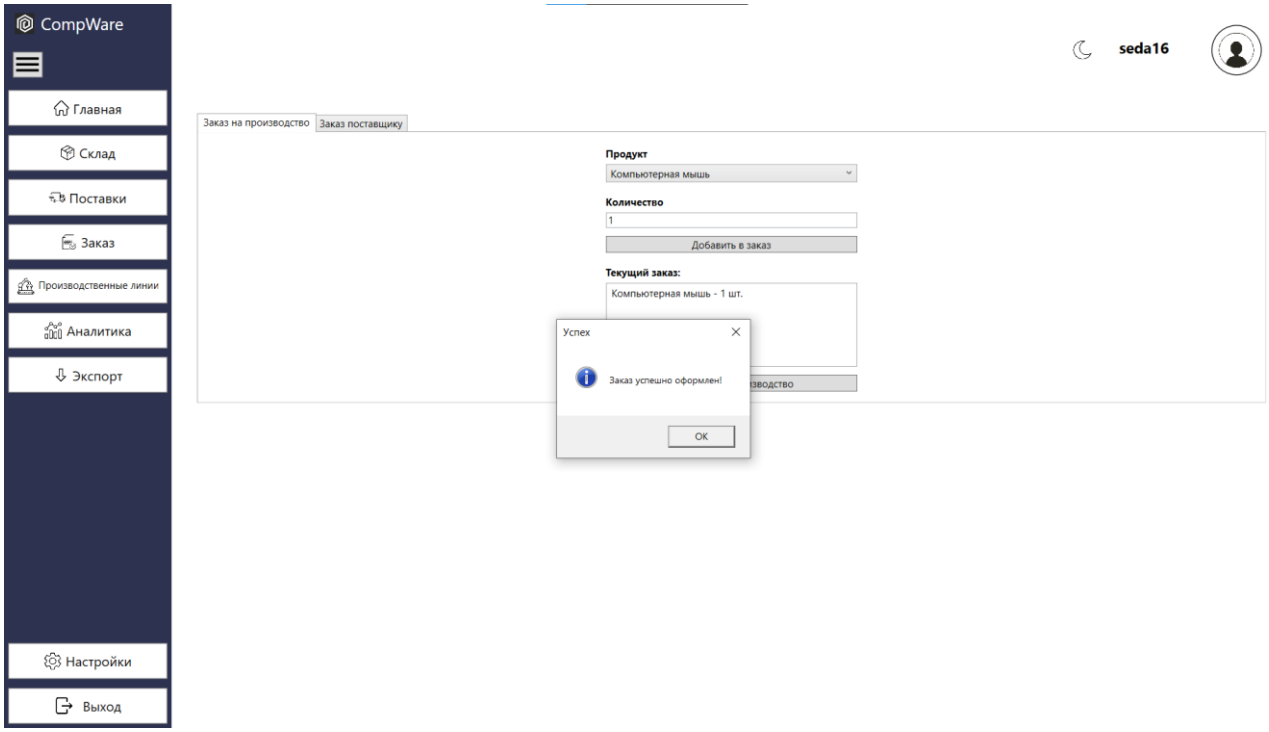


Рисунок 2.13 – Вывод сообщения при вводе корректных данных

Тестирование подтвердило стабильную работу всех проверяемых компонентов в нормальных условиях эксплуатации.

2.5.4 Тестирование в экстремальных условиях

На рисунках 2.14-2.16 видно, что при регистрации, авторизации и создании заказа, если пользователь не введет данные, либо заполнит не все поля, то программа выдаст ошибку с просьбой заполнить все поля.

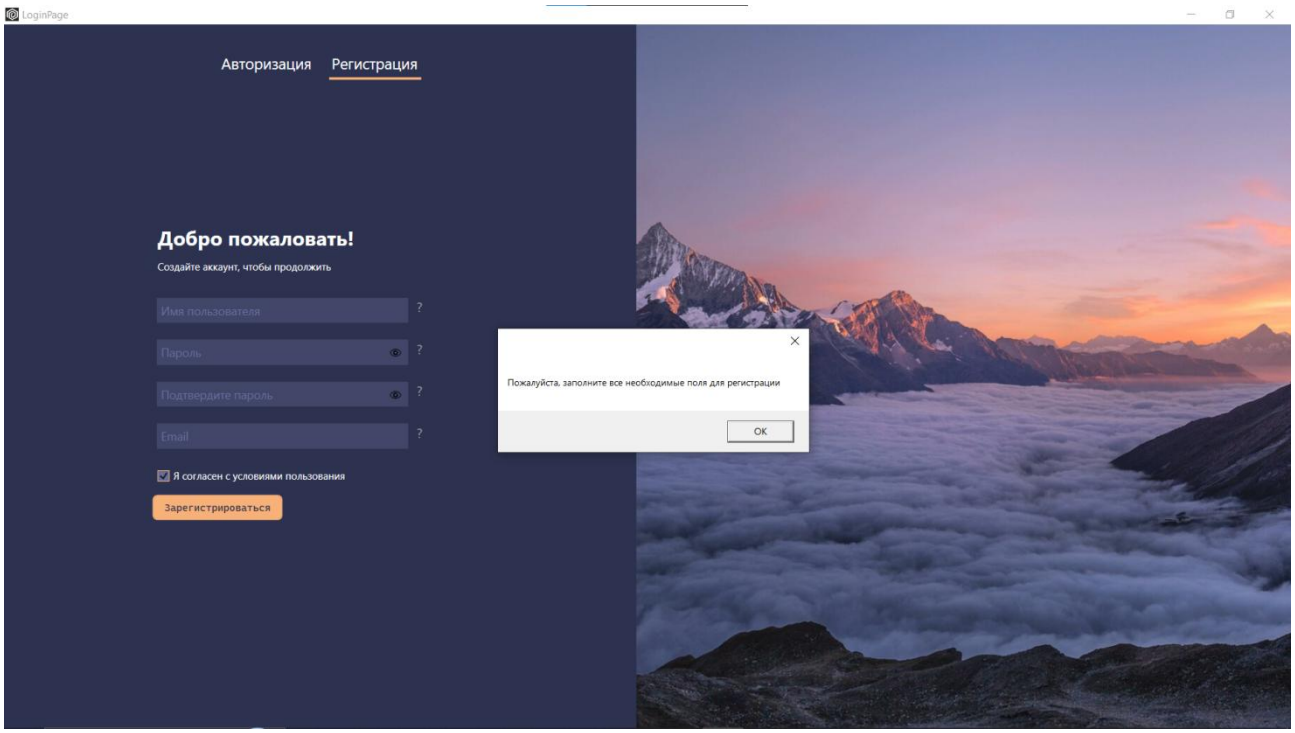


Рисунок 2.14 – Вывод сообщения при незаполненных полях регистрации

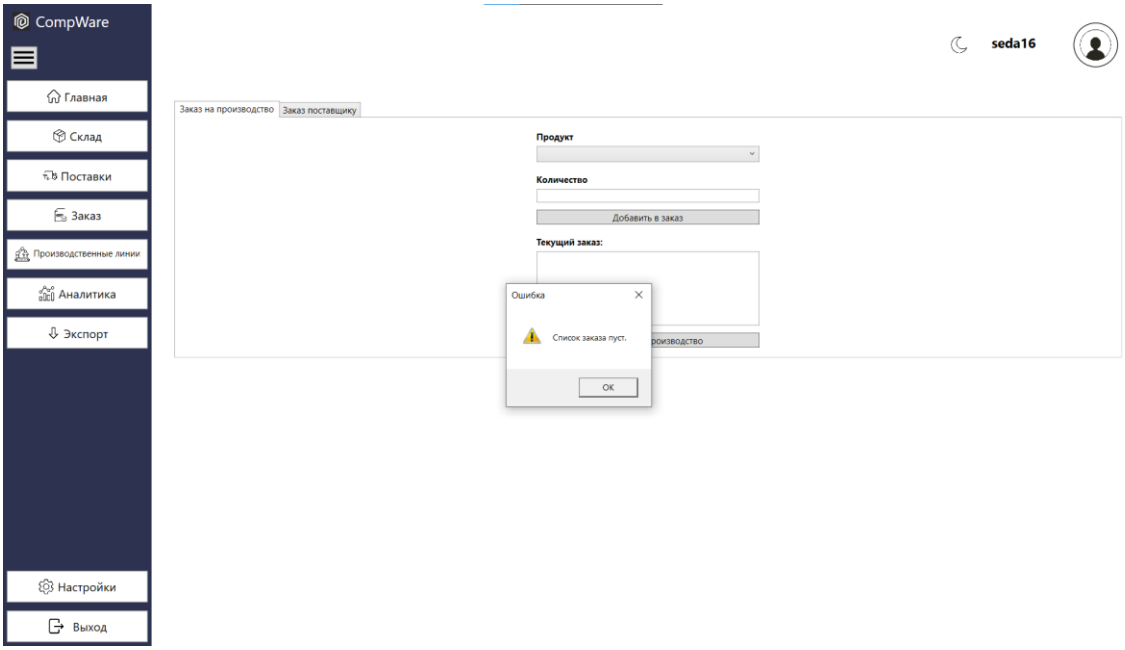


Рисунок 2.15 – Вывод сообщения о пустом списке

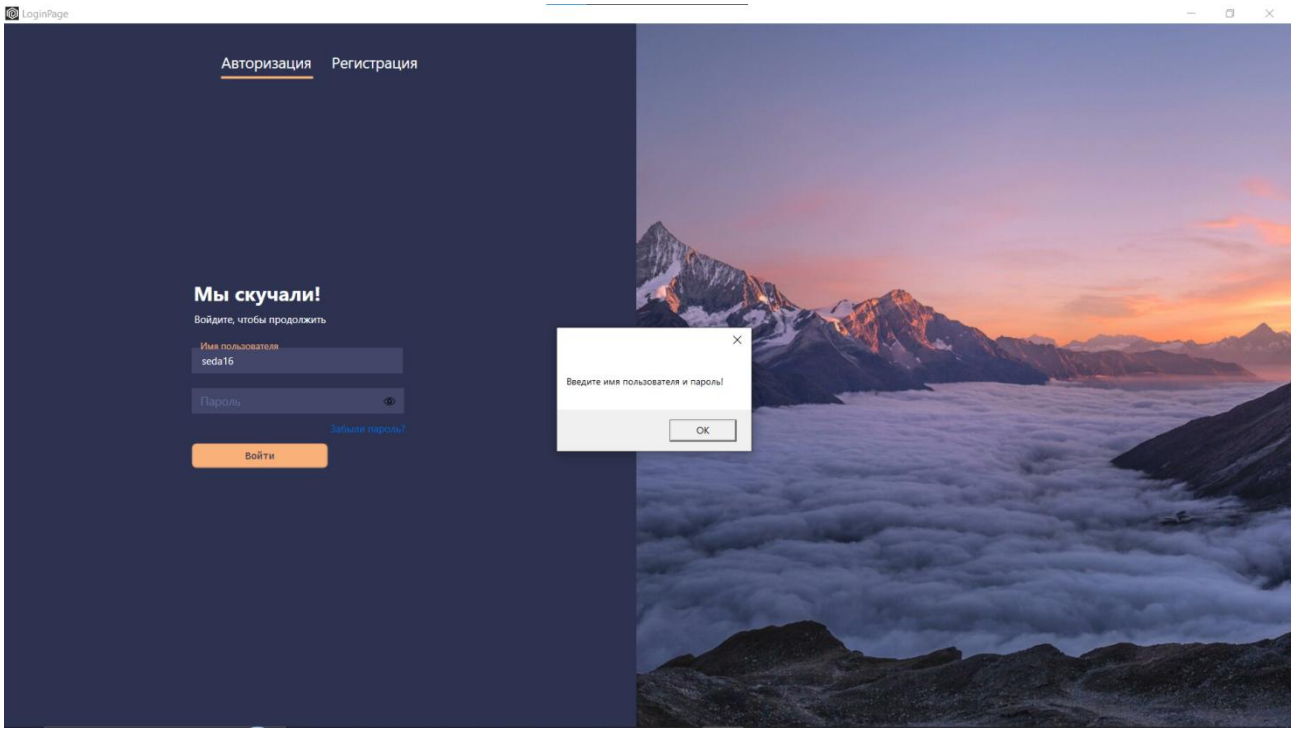


Рисунок 2.16 – Вывод сообщения при незаполненных полях авторизации

На основании вышеприведенного тестирования системы можно сделать вывод, что программа работает правильно в экстремальных условиях, а значит можно рассчитывать на ее максимально возможную производительность.

2.5.5 Тестирование программы в исключительных условиях

На рисунках 2.17-2.18 представлено тестирование в исключительных условиях

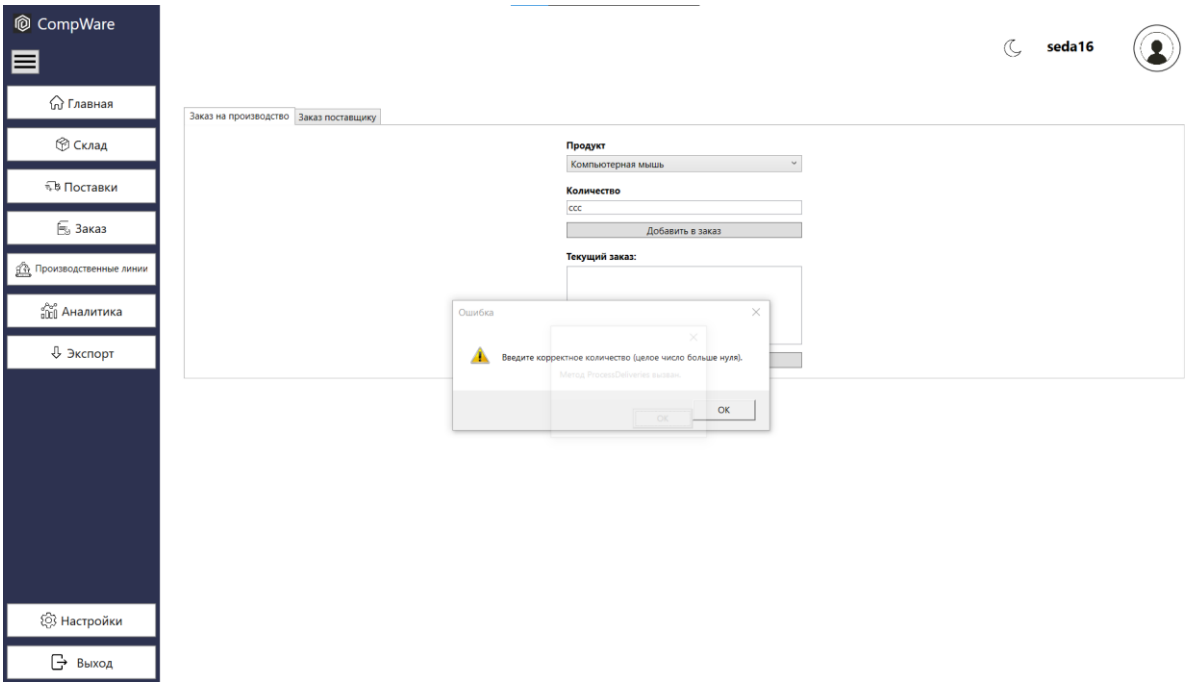


Рисунок 2.17 – Ввод некорректного числа

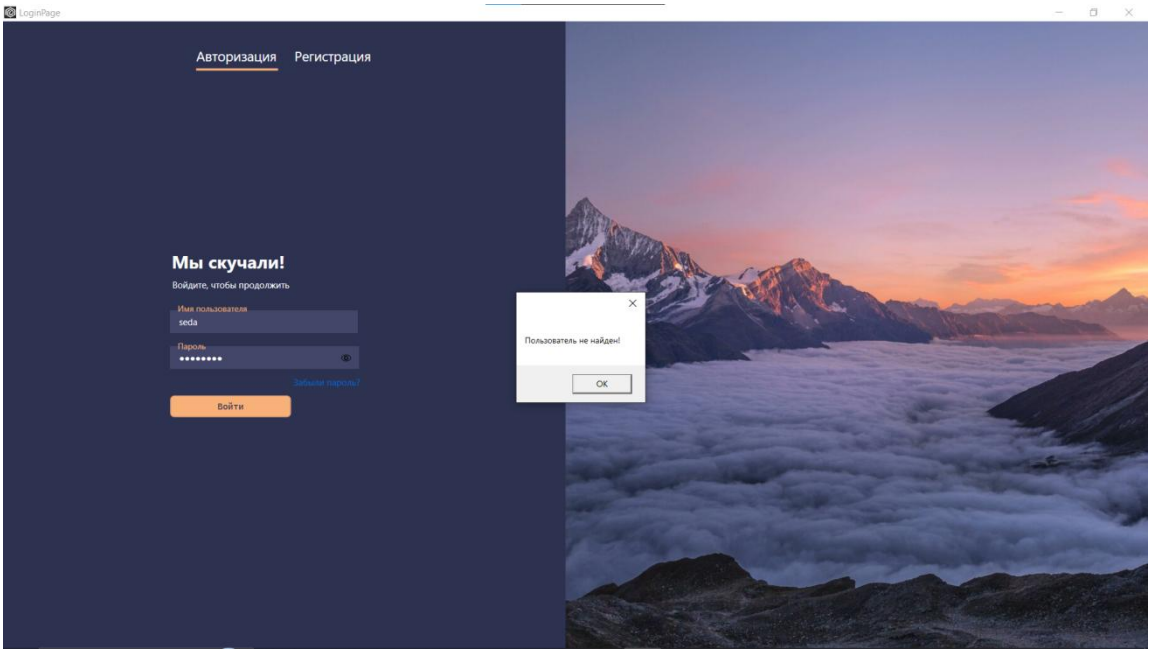


Рисунок 2.18 – Ввод несуществующего имени пользователя

Результат тестирования программы в исключительных ситуациях показал, что программа отвергает некорректные данные и выводит сообщения.

Программа прошла тестирование в нормальных и экстремальных условиях, а также в исключительных ситуациях. На основании этого можно сделать вывод, что программа работает корректно.

2.6 Руководство пользователя

2.6.1 Назначение программы

Программа разработана для автоматизации управления производственными процессами на предприятиях, занимающихся производством компьютеров и периферийного оборудования. Она позволяет вести учет комплектующих, отслеживать производственные линии, управлять заказами, контролировать поставки и ремонтное обслуживание. Основная цель программы — повысить эффективность производства, минимизировать издержки и улучшить контроль за всеми этапами производственного процесса.

2.6.2 Условия выполнения программы

Для корректной работы программы требуются следующие условия:

- персональный компьютер с процессором частотой не менее 2 ГГц;
- оперативная память объемом не менее 1 ГБ;
- установленный и сконфигурированный сервер баз данных MySQL версии 8.0.31;
- операционная система Windows 7 и выше;
- жесткий диск емкостью не менее 1 ТБ.

2.6.3 Входные данные

После установки и запуска приложения необходимо выполнить следующие шаги:

- регистрация: после запуска приложения появится форма регистрации. Если нет аккаунта у пользователя, необходимо зарегистрироваться в системе, введя свои данные;
- авторизация: необходимо ввести логин и пароль в соответствующие поля;
- вход в систему: после ввода учетных данных требуется нажать на кнопку «Войти», чтобы войти в систему.

После выполнения этих шагов пользователь будет авторизован в системе и готов к работе с приложением.

2.6.4 Выполнение программы

1) Иконка приложения находится на рабочем столе (Рисунок 2.19). Откроем его, 2 раза кликнув левой кнопкой мышки.

Рисунок 2.19 – Рабочий стол

2) Запускается заставка приложения, продемонстрированная на рисунке 2.20. Выводится приветственное сообщение.

Рисунок 2.20 – Заставка приложения

3) На рисунке 2.21 представлена форма авторизации. После закрытия заставки открывается форма авторизации. Если пользователь уже зарегистрирован в системе – необходимо авторизоваться, введя имя пользователя и пароль (при необходимости можно восстановить). Если же пользователь попал на форму приложения впервые, необходимо нажать на кнопку «Регистрация».

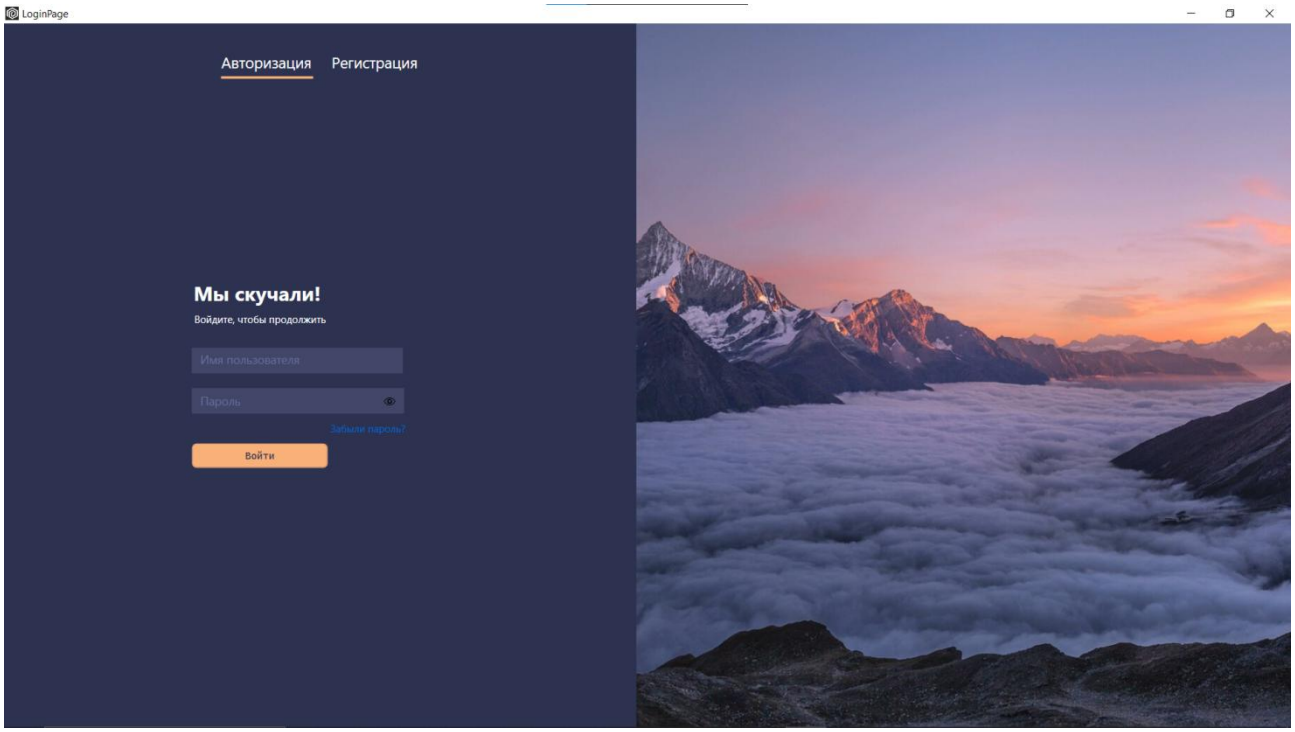


Рисунок 2.21 – Форма авторизации

4) На рисунке 2.22 представлена форма регистрации пользователей в системе. При нажатии на кнопку «Регистрация» открывается форма ниже.

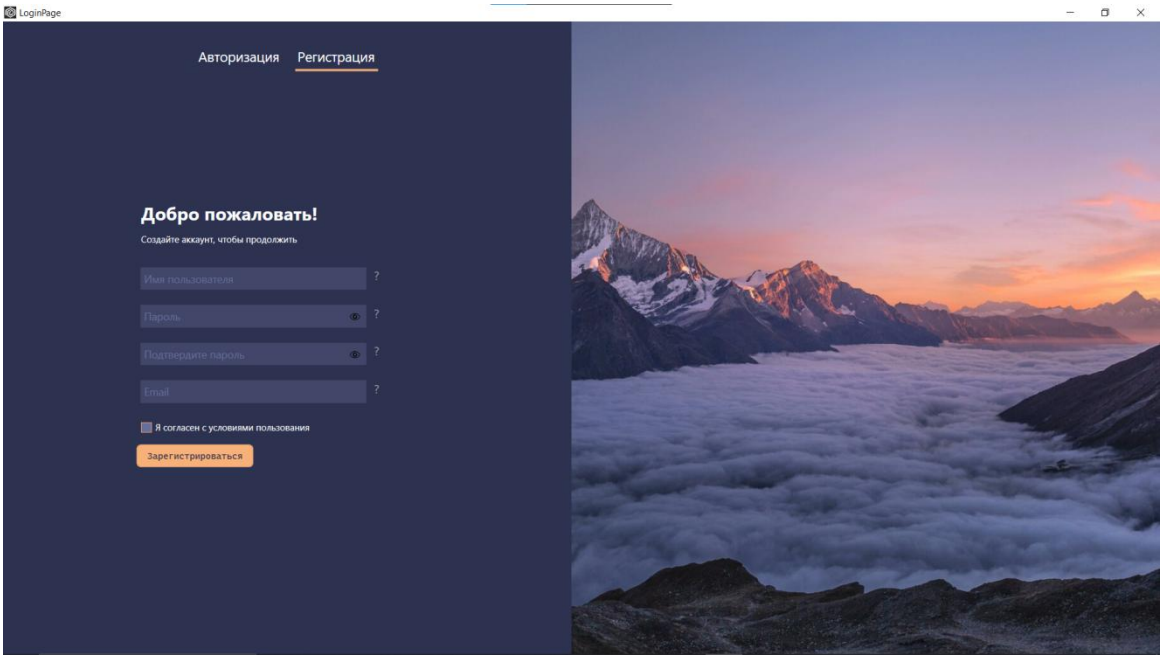


Рисунок 2.22 – Форма регистрации

5) Если же пользователь забыл пароль – его можно легко восстановить с помощью электронной почты, введенной во время регистрации. На рисунке 2.23 представлена форма восстановления пароля.

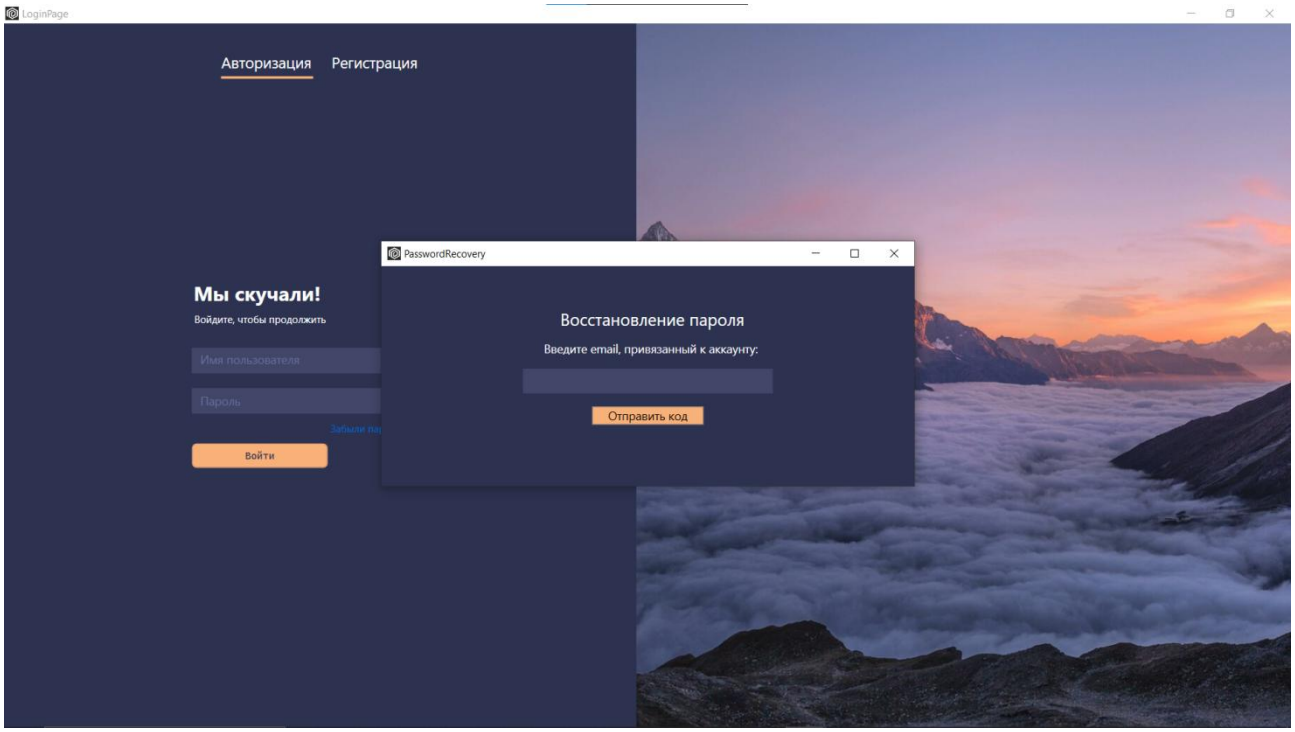


Рисунок 2.23 – Форма восстановления пароля

6) На рисунке 2.24 изображен интерфейс главной формы – меню. На этой форме расположена панель с кнопками с помощью которых открываются другие формы.

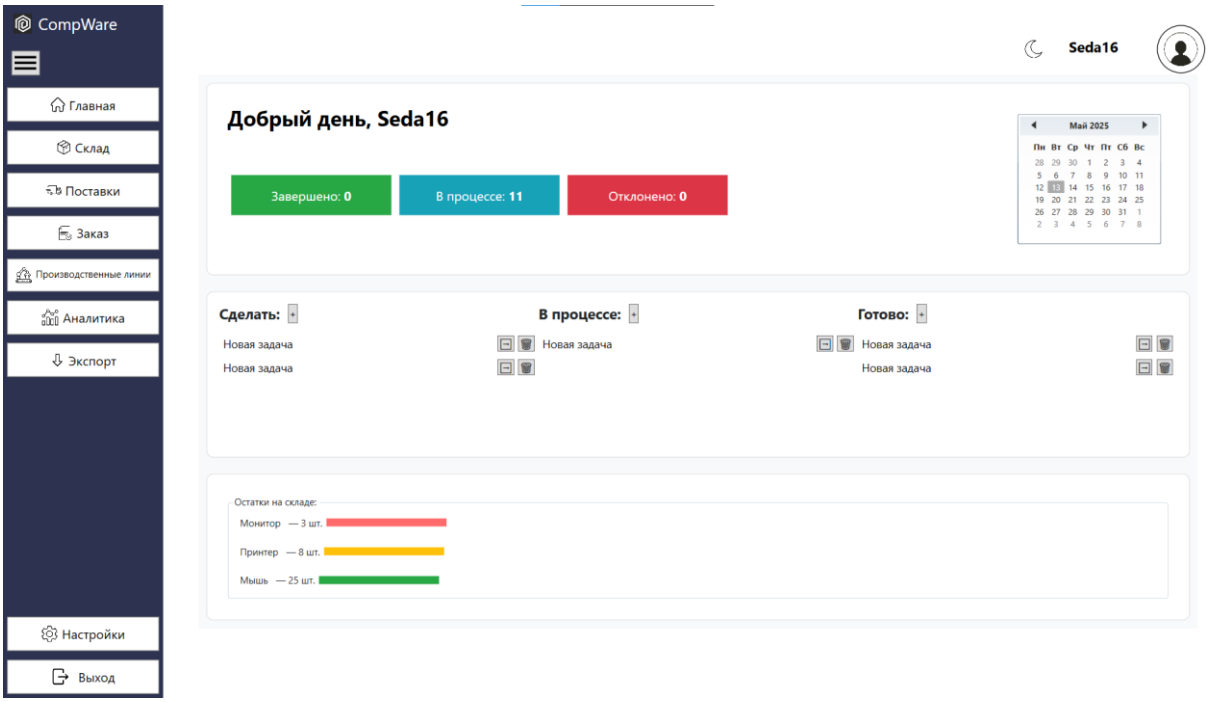


Рисунок 2.24 – Интерфейс главной формы

7) На рисунке 2.25 представлена вкладка «Склад». Загружается таблица компонентов, которые можно отсортировать.

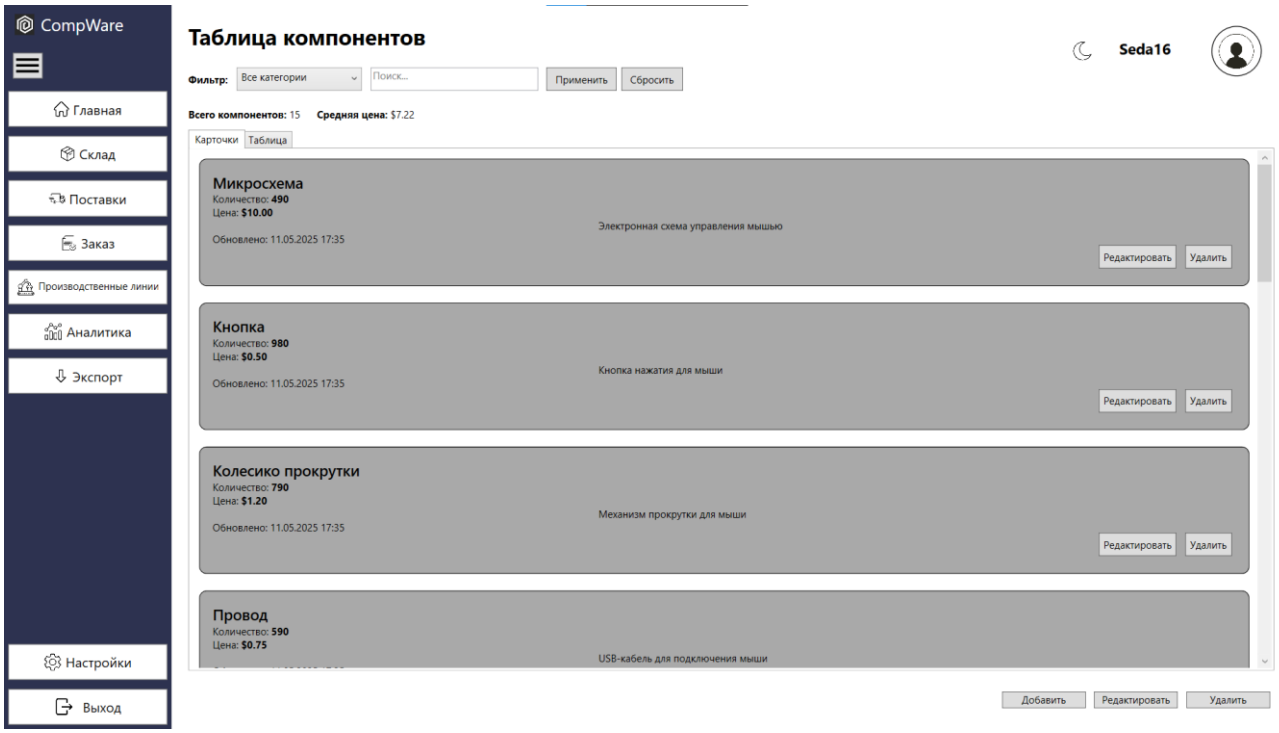


Рисунок 2.25 – Вкладка «Склад»

8) На рисунке 2.26 изображен интерфейс формы, на которую пользователь попадает при нажатии на кнопку «Поставки». Загружается таблица поставок.

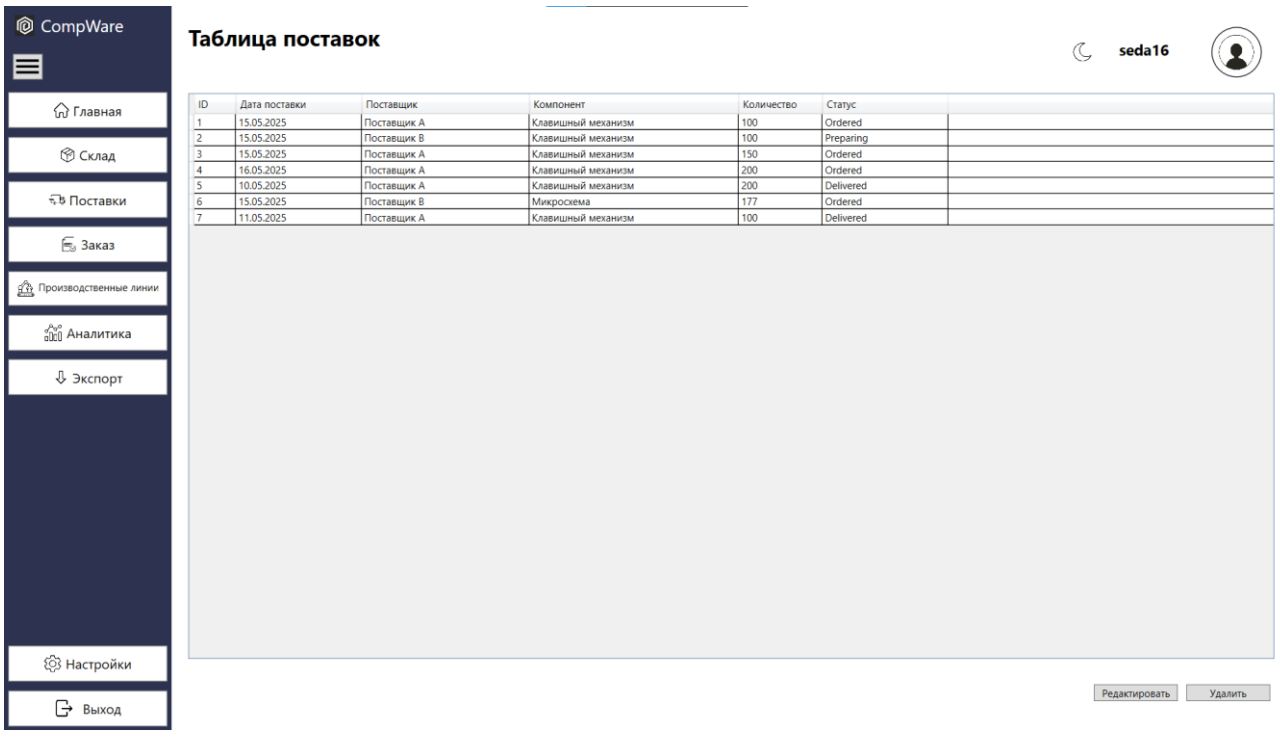


Рисунок 2.26 – Интерфейс формы «Поставки»

9) На рисунке 2.27 представлена форма добавления компонента на склад. Введя данные и нажав на кнопку «Сохранить», компонент добавляется в базу данных.

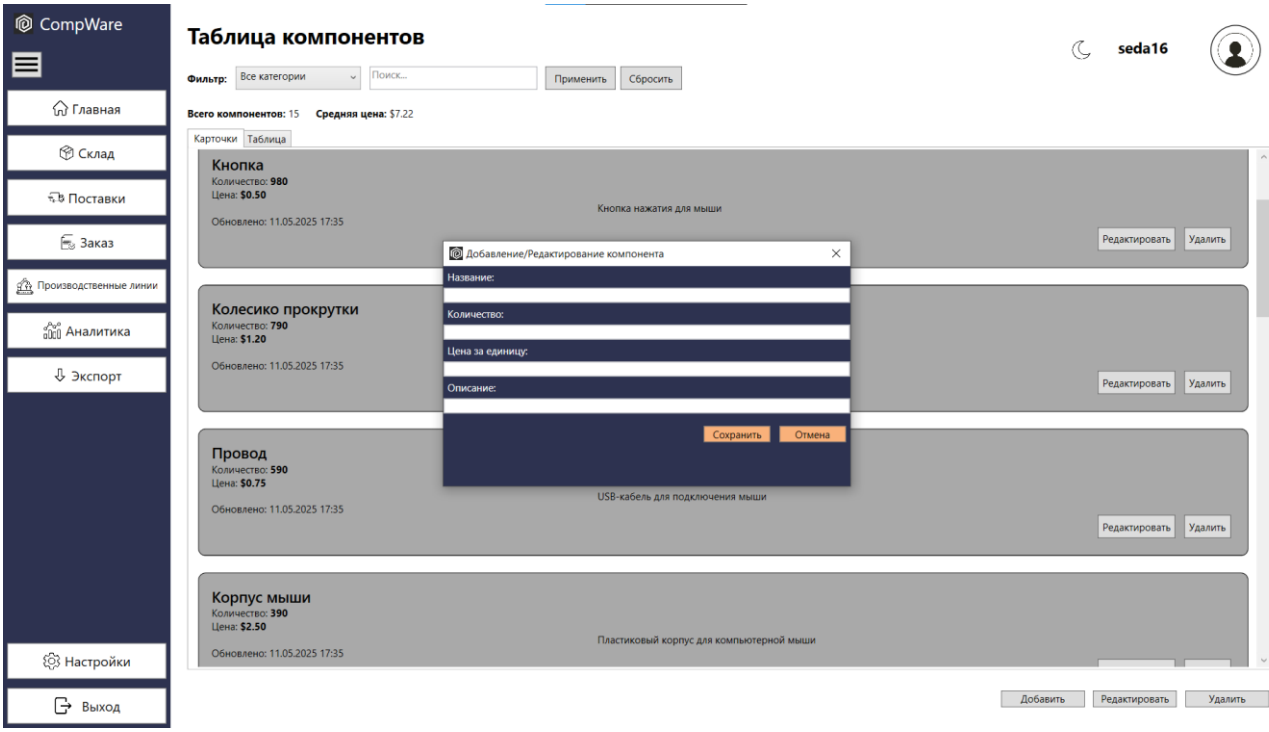


Рисунок 2.27 – Интерфейс формы добавления компонента

10) На рисунке 2.28 представлена форма редактирования компонента. Загружается информация про компонент. При нажатии на кнопку «Сохранить» данные обновляются в базе данных.

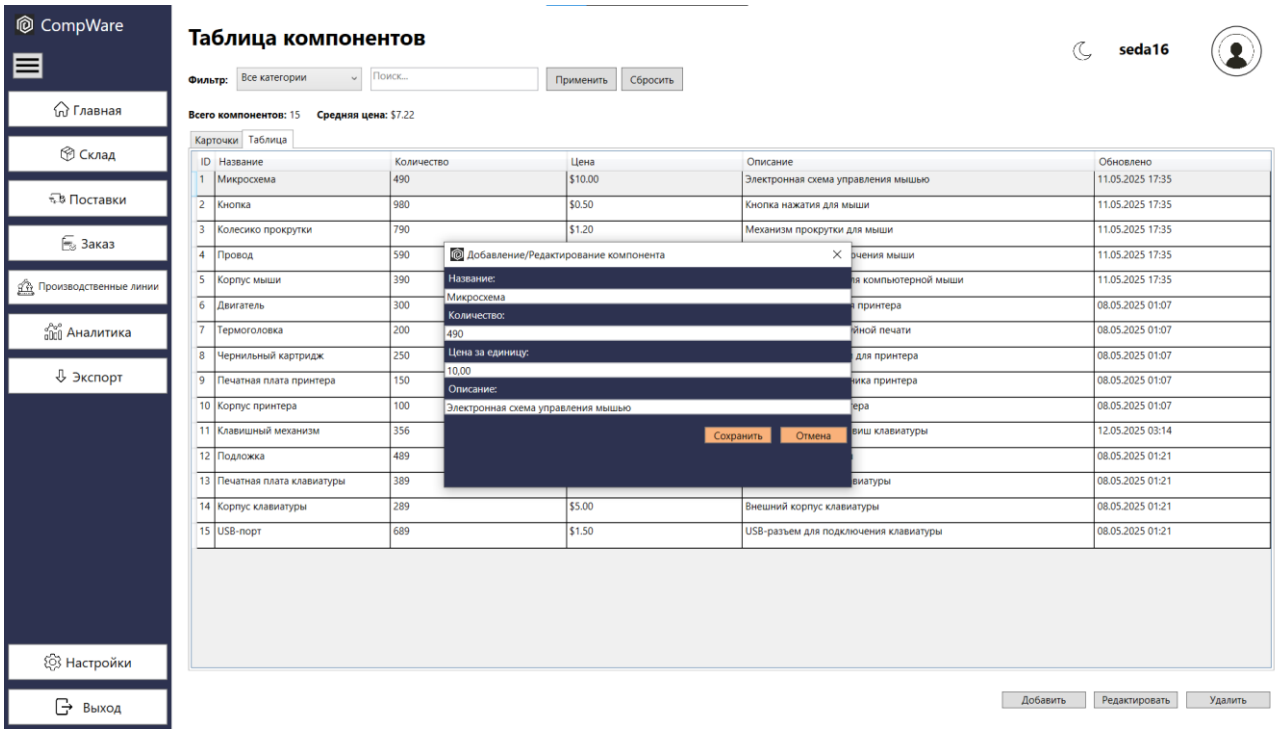


Рисунок 2.28 – Интерфейс формы редактирования компонента

11) На рисунке 2.29 представлена форма редактирования производственной линии.

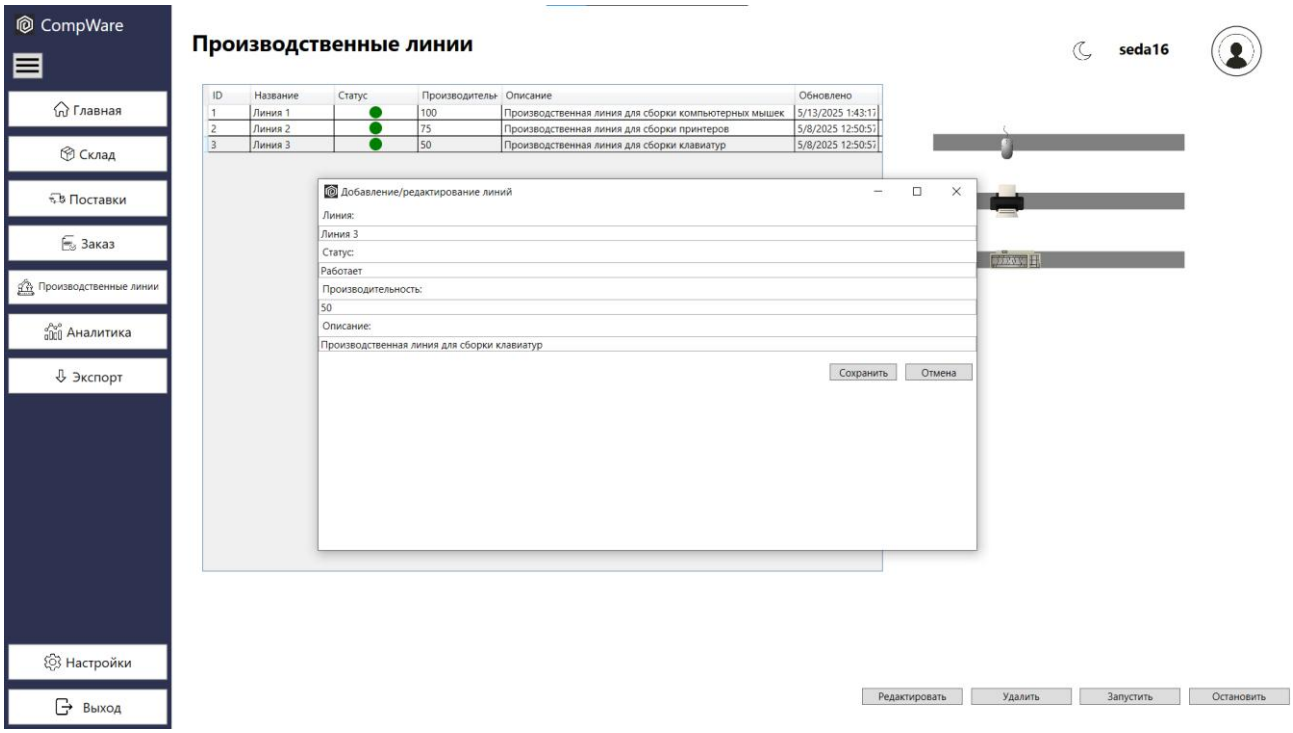


Рисунок 2.29 – Интерфейс формы добавления поставки

12) На рисунке 2.30 представлена форма оформления заказа.

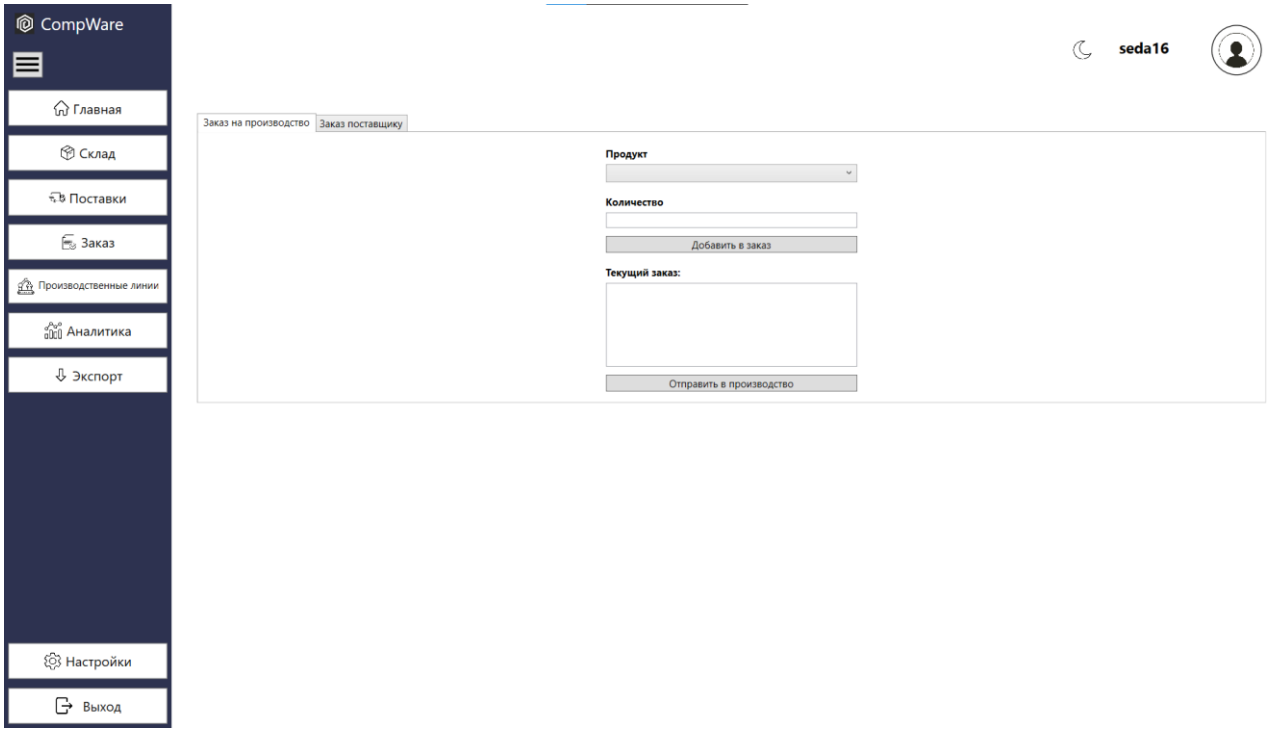


Рисунок 2.30– Форма оформления заказа

13) На рисунке 2.31 представлена форма производственных линий. Здесь пользователь может редактировать, удалить, запустить и остановить линии.

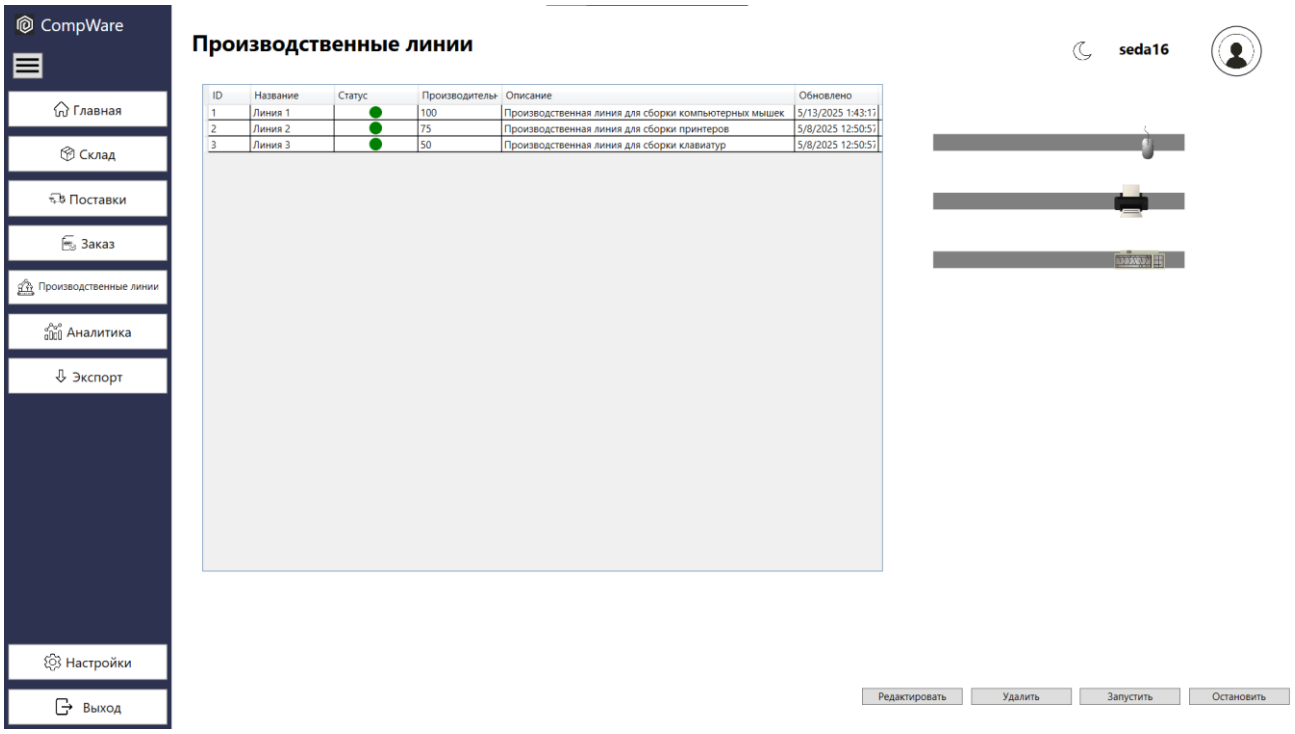


Рисунок 2.31 – Форма производственных линий

14) На рисунке 2.32 представлена форма настроек. Здесь пользователь может изменить свои личные данные и аватар.

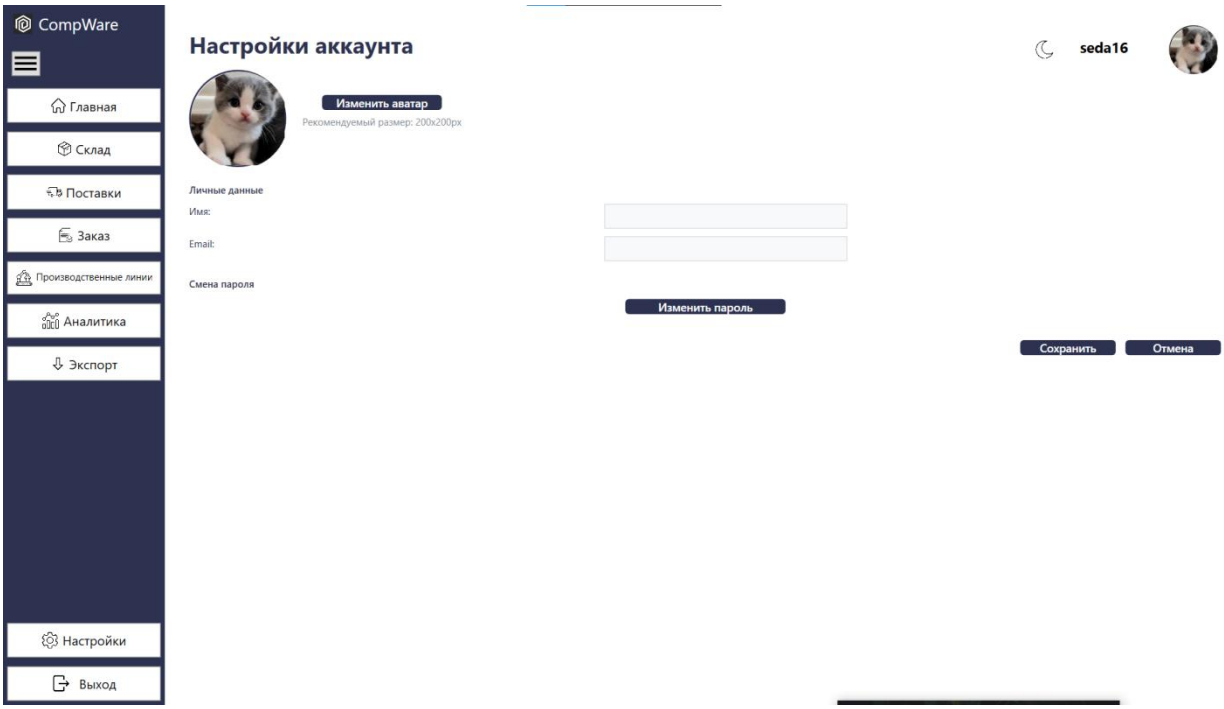


Рисунок 2.32 – Форма настроек

15) Завершить работу в системе можно нажав на кнопку «Выход».

В ходе работы с программой могут появляться различные сообщения. Тексты сообщений предоставляют информацию о текущем статусе выполнения операции, предупреждениях или возможных ошибках. Пользователь должен внимательно читать сообщения и действовать в соответствии с инструкциями.

На рисунке 2.33 представлен вывод сообщения при заполнении не всех полей.

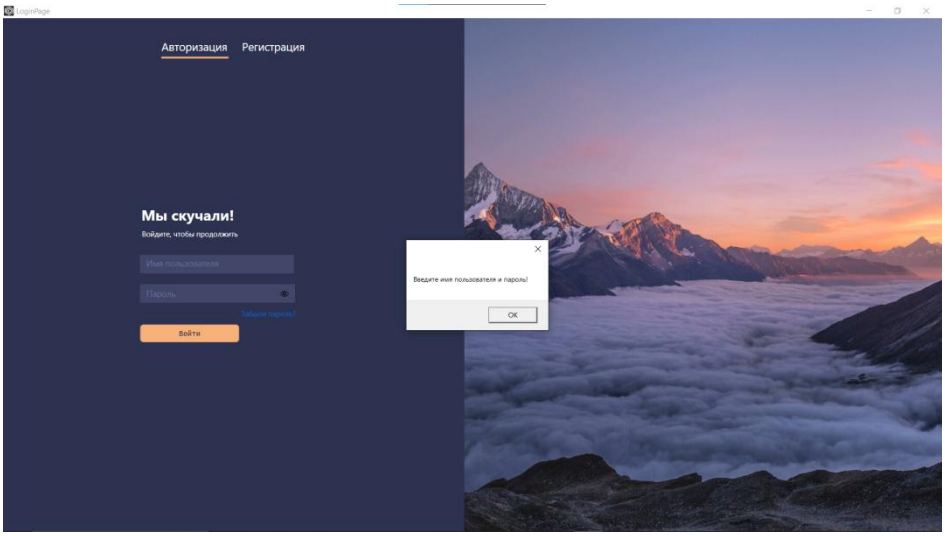


Рисунок 2.33 – Вывод сообщения при заполнении не всех полей

На рисунке 2.34 представлен вывод сообщения при вводе неправильного имени пользователя или же неправильного пароля.

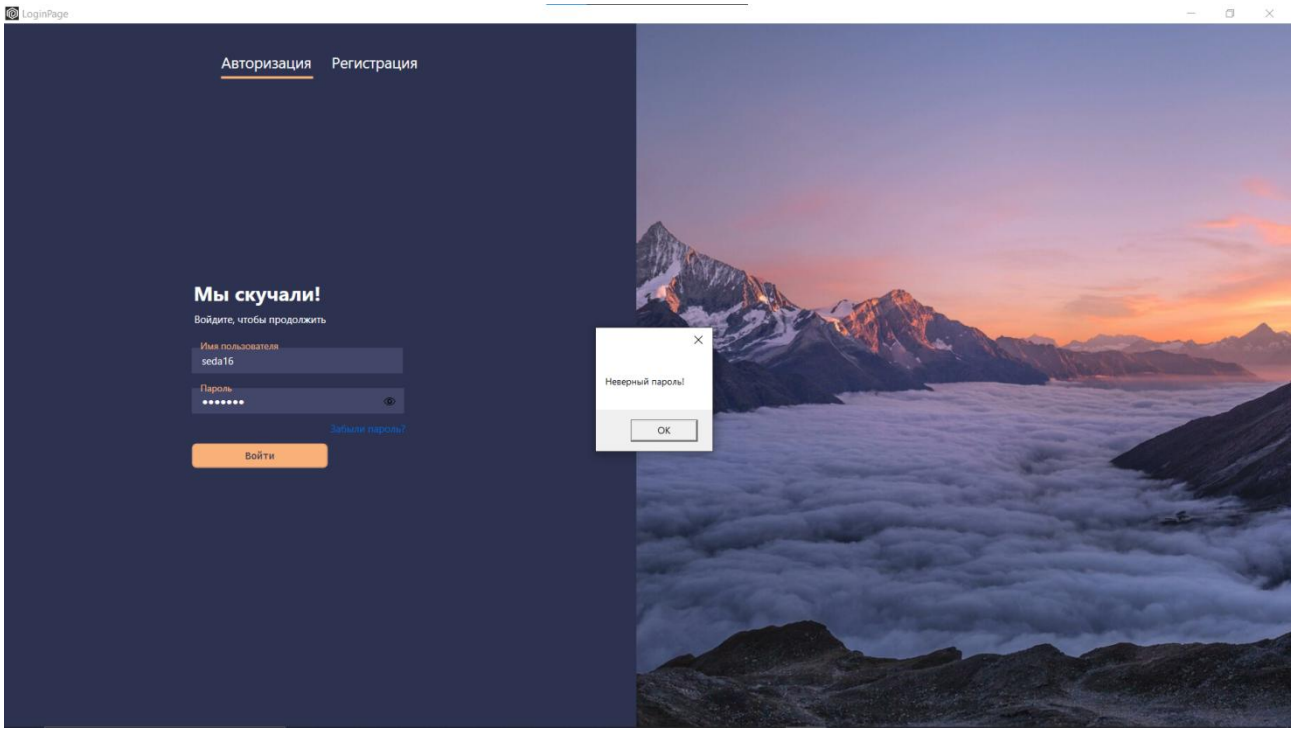


Рисунок 2.34 – Вывод сообщения при неправильном имени пользователя/пароле

На рисунке 2.35 представлен вывод сообщения при заполнении не всех полей во время регистрации.

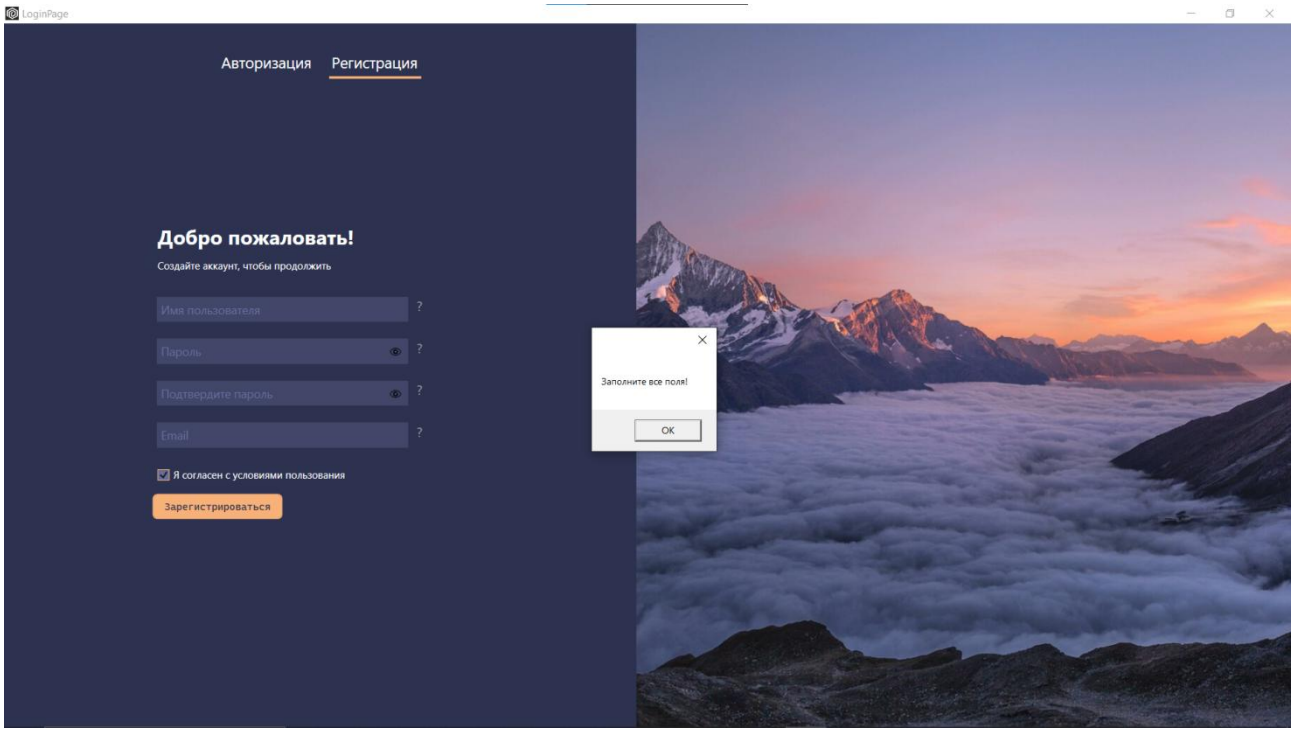


Рисунок 2.35 – Вывод сообщения при заполнении не всех полей

На рисунке 2.36 представлен вывод сообщения, когда пароли не совпадают.

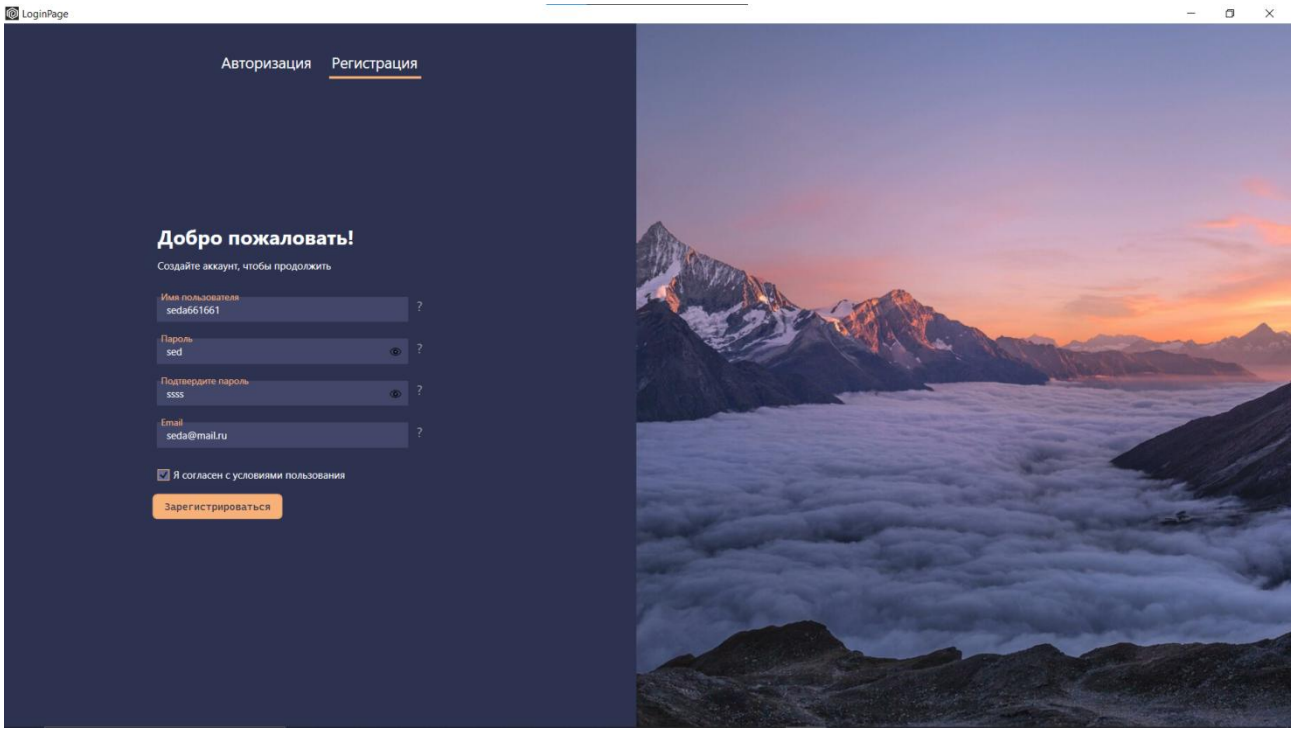


Рисунок 2.36 – Не совпадение паролей

На рисунке 2.37 представлен вывод сообщения при нажатии на кнопку «Редактировать».

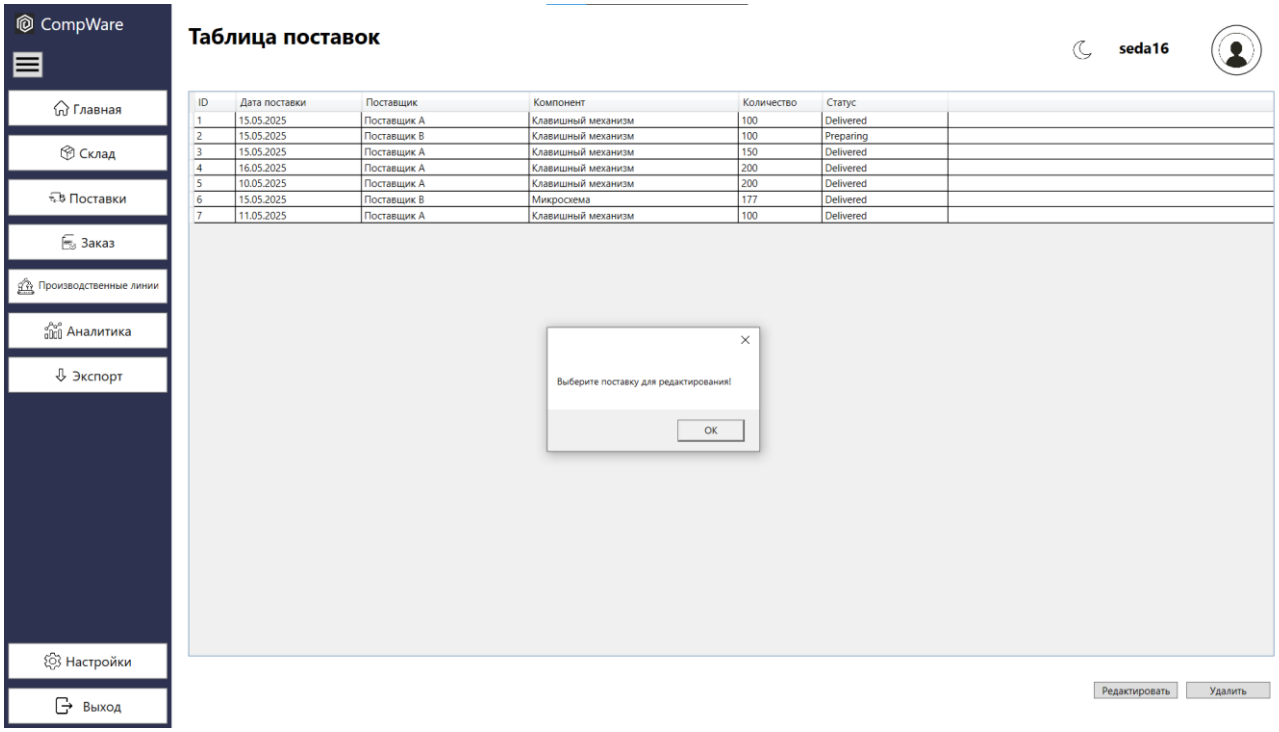


Рисунок 2.37 – Вывод сообщения

На рисунке 2.38 представлен вывод сообщения при удалении компонента.

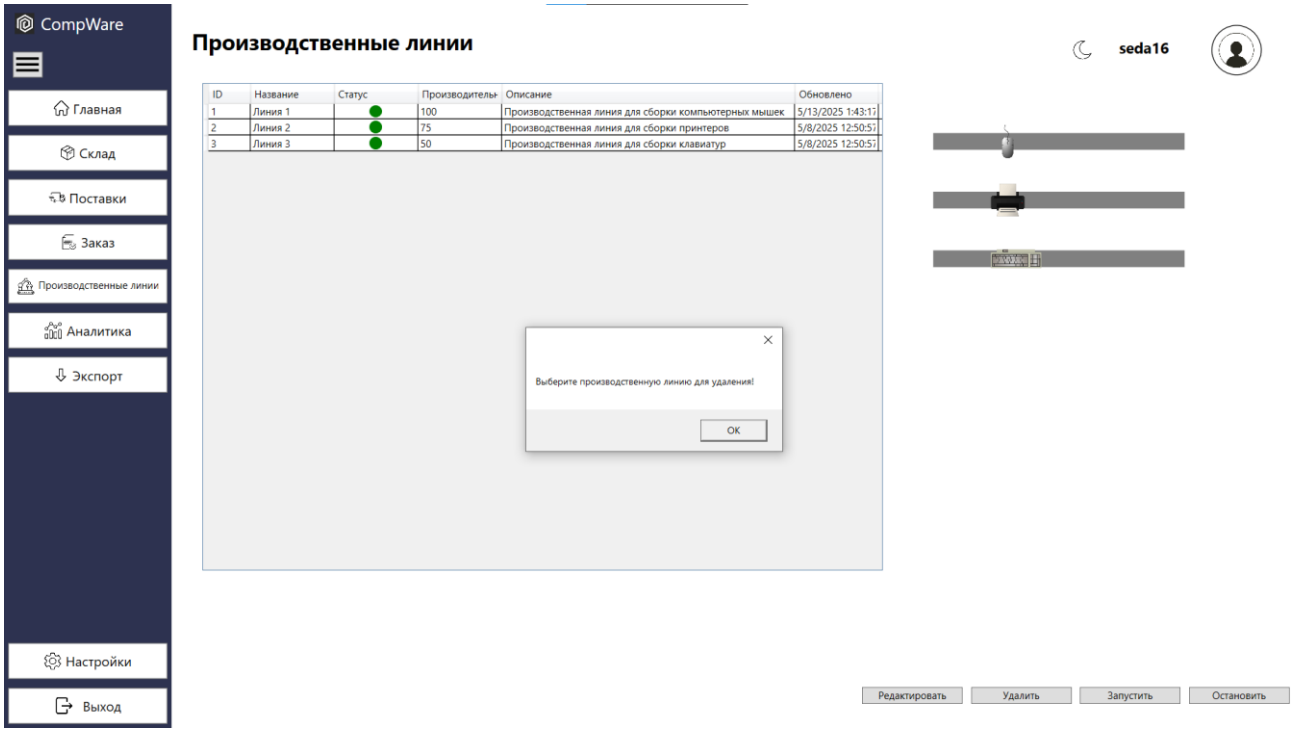


Рисунок 2.38 – Вывод сообщения

2.6.5 Выходные данные

Программа генерирует следующие выходные данные:

- списки комплектующих с актуальным количеством на складе;
- статус производственных линий (работает, остановлена, на обслуживании);
- состояние текущих заказов (выполняется, завершен, отменен);

Данное руководство предоставляет основную информацию для успешного использования программы.

2.7 Вывод по разделу

В данном разделе была проведена комплексная работа по проектированию и реализации программной системы. На первом этапе разработана архитектура системы, определены её ключевые компоненты и их взаимодействие, что обеспечило четкое понимание структуры будущего приложения. Затем выполнено проектирование структуры данных, позволяющей эффективно организовать хранение и обработку информации.

Особое внимание уделено разработке пользовательского интерфейса, который был спроектирован с учетом потребностей целевой аудитории для обеспечения удобства и интуитивной понятности. Для реализации функциональности программы созданы схемы алгоритмов, определяющие логику работы как основной системы, так и её подпрограмм.

Важной частью раздела стали процессы отладки и тестирования. Программа была тщательно проверена в различных условиях, включая нормальные, экстремальные и исключительные сценарии, что позволило выявить и устранить ошибки, а также гарантировать стабильную работу системы.

Завершающим этапом стало составление руководства пользователя, включающего описание назначения программы, условий её эксплуатации, входных данных и последовательности работы. Это обеспечит пользователей всей необходимой информацией для эффективного взаимодействия с системой.

Таким образом, в рамках раздела успешно выполнены все этапы проектирования, реализации и тестирования программного обеспечения, что создает прочную основу для его дальнейшего внедрения и использования.