ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ РЕСПУБЛИКИ ХАКАСИЯ

«ХАКАССКИЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»

**ОТЧЕТ**

по учебной практике

по профессиональному модулю

**ПМ.01. РАЗРАБОТКА МОДУЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ КОМПЬЮТЕРНЫХ СИСТЕМ**

специальности 09.02.07 Информационные системы и программирование   
Квалификация Программист

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент гр. | ИС(ПРО)-41 |  |  |  | Ушаков И. С. |
|  |  |  | *подпись* |  | *Фамилия И.О.* |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Руководитель практики  от ГБПОУ РХ ХПК |  |  |  |  |  |  | Черкашин Д.С. |
|  | *оценка* |  | *дата* |  | *подпись* |  | *Фамилия И.О.* |

Абакан 2024 г.

# Разработка программных модулей

## Описание предметной области

Компания «Мастер пол», занимающаяся производством напольных покрытий, разрабатывает систему управления для оптимизации своей деятельности. Система включает несколько ключевых подсистем, каждая из которых решает определенные задачи.

### Подсистемы и их функциональные возможности:

1. **Продукция**
   * Хранение данных о продукции: артикул, тип, наименование, описание, изображения, минимальная цена для партнёров, размеры упаковки, вес, сертификаты качества, стандарты и время производства.
   * Учет изменений: история корректировки минимальной стоимости, себестоимости, информация о задействованных цехах и персонале.
   * Связь с производственными процессами: указание материалов, необходимых для изготовления продукции, и управление производственными данными.
2. **Склад и материалы**
   * Управление запасами: регистрация поступлений сырья и готовой продукции, отслеживание остатков, списание материалов в производство, учет товарно-материальных ценностей.
   * Резервирование материалов: автоматическое бронирование сырья для производственных заказов.
   * Учет критических остатков: система напоминаний при достижении минимального уровня запасов.
3. **Производство**
   * Организация производственного процесса: контроль качества продукции, учет времени изготовления, расчет себестоимости.
   * Управление технологическими картами: определение технологий производства и контроль использования материалов.
   * Мониторинг занятости цехов и сотрудников: распределение заданий, учет рабочего времени и планирование производственных смен.
4. **Сотрудники**
   * Управление персоналом: регистрация данных сотрудников (паспортные данные, банковские реквизиты, состояние здоровья, семейное положение).
   * Контроль доступа: автоматическое управление дверями и турникетами через систему электронных пропусков, сбор данных о перемещении сотрудников.
   * Обучение и сертификация: учет допусков сотрудников к оборудованию и анализ их профессиональной эффективности.
5. **Партнеры**
   * Ведение информации о партнерах: тип, наименование, адрес, ИНН, контактные данные, логотип, рейтинг, места продаж.
   * Управление скидками: расчет скидок на основе объемов продаж и мотивация партнёров на увеличение закупок.
   * Ведение истории сотрудничества: анализ данных о продажах для формирования предложений партнерам.
   * Обработка заявок: прием заявок, расчет стоимости и сроков производства, уведомления о статусах оплаты и производства.
6. **Менеджеры**
   * Регистрация новых партнёров, управление рейтингами, ведение истории изменений.
   * Работа с заявками: анализ данных о продажах и формирование предложений для партнёров на основе их покупательской истории.
   * Контроль за оплатами: уведомления о поступлении предоплаты и управление процессом от производства до доставки готовой продукции.
7. **Поставщики**
   * Управление данными поставщиков: наименование, ИНН, тип компании, история поставок.
   * Контроль качества материалов: аналитика по качеству поставок и принятие решений о дальнейшем сотрудничестве с поставщиками.
8. **Заявки и заказы**
   * Создание заявок: оформление заказов на продукцию, расчет стоимости и сроков изготовления.
   * Управление статусами: автоматическое аннулирование заявок при отсутствии предоплаты, отслеживание выполнения и закрытие заказов.

Система нацелена на повышение эффективности работы всех подразделений компании, автоматизацию процессов и улучшение взаимодействия между партнёрами, сотрудниками и поставщиками.

## Спецификация требований

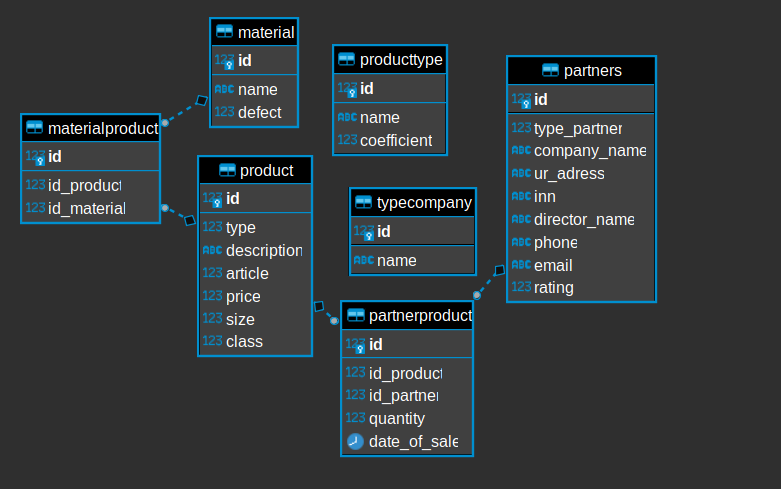


Диаграмма представляет собой схему базы данных, состоящую из восьми таблиц: Partners, PartnerProduct, Product, MaterialProduct, Material, ProductType и TypeCompany.

1. Partners:

* id - int (PK)
* type\_partner - int
* company\_name - varchar
* ur\_adress - varchar
* inn - varchar
* director\_name - varchar
* phone - varchar
* email - varchar
* rating - int

1. PartnerProduct:

* id - int (PK)
* id\_product - int (FK)
* id\_partner - int (FK)
* quantity - int
* date\_of\_sale - date

1. Product:

* id - int (PK)
* type - int (FK)
* description - varchar
* article - int
* price - float
* size - float
* class - int

1. MaterialProduct:

* id - int (PK)
* id\_product - int (FK)
* id\_material - int (FK)

1. Material:

* id - int (PK)
* name - varchar
* defect - float

1. ProductType:

* id - int (PK)
* name - varchar
* coefficient - float

1. TypeCompany:

* id - int (PK)
* name - varchar

Функциональные требования

1. Регистрация и управление партнерами

* Менеджер регистрирует нового партнера, указывая тип партнера, наименование компании, юридический адрес, ИНН, ФИО директора, контактные данные, логотип, и начальный рейтинг.
* Система сохраняет данные о местах продаж партнера, включая розничные и оптовые точки, интернет-магазины.
* Менеджер может обновить рейтинг партнера и добавить комментарий о причине изменения.
* В системе сохраняется история всех изменений рейтинга партнера.

1. Создание и обработка заявок

* Менеджер создает заявку на продукцию, заполняя список продукции с указанием стоимости и сроков производства.
* Система автоматически проверяет наличие материалов на складе и рассчитывает время изготовления.
* Партнер может отменить заявку до момента внесения предоплаты через личный кабинет.
* Если предоплата не поступает в течение трех дней, система автоматически аннулирует заявку и отправляет уведомление партнеру.
* После получения предоплаты менеджер подтверждает заявку, и продукция отправляется в производство.
* Когда продукция готова, система предлагает организовать доставку или самовывоз.

1. Управление продукцией

* Менеджер добавляет и обновляет информацию о продукции: артикул, тип, наименование, описание, изображения, минимальная стоимость, габариты и вес.
* Система хранит историю изменений минимальной стоимости продукции.
* Продукция связана с технологическими картами, описывающими необходимые материалы для производства.

1. Управление складом

* Система регистрирует поступление материалов от поставщиков и учитывает остатки на складе.
* При создании заявки материалы автоматически резервируются, а по завершении производства списываются со склада.
* Менеджер может вручную списывать материалы при необходимости и регистрировать новые поступления.
* Система уведомляет менеджеров при достижении минимального уровня материалов.

1. Управление производственным процессом

* Мастер производства назначает сотрудников на выполнение задач и контролирует качество выпускаемой продукции.
* Система фиксирует время начала и завершения производства каждой единицы продукции.
* Сотрудники регистрируют свое присутствие на производстве с помощью электронных пропусков.

1. Учет сотрудников и контроль доступа

* Система регистрирует сотрудников, включая ФИО, дату рождения, паспортные данные, банковские реквизиты, состояние здоровья, семейное положение.
* Карта сотрудника обеспечивает доступ в производственные помещения. Система турникетов собирает данные о перемещении сотрудников.
* На основании данных о перемещении создается статистика для анализа эффективности работы сотрудников.

1. Управление поставщиками

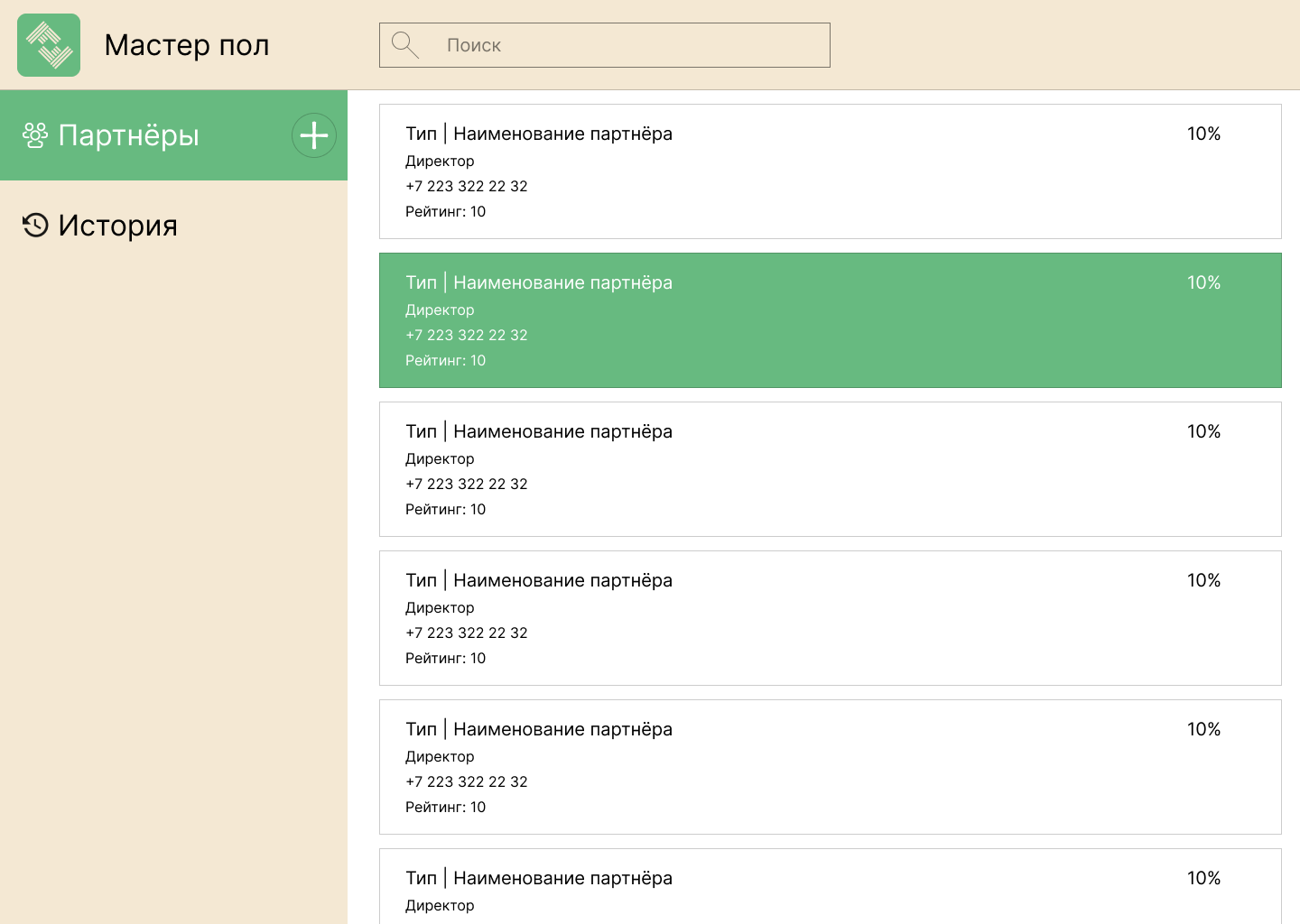
* Менеджеры контролируют список поставщиков и могут добавлять, редактировать или удалять поставщиков.
* Система сохраняет историю поставок материалов, связывая поставки с конкретными партнерами.
* Аналитики используют данные о качестве поставляемых материалов для оценки и принятия решений о дальнейшем сотрудничестве.

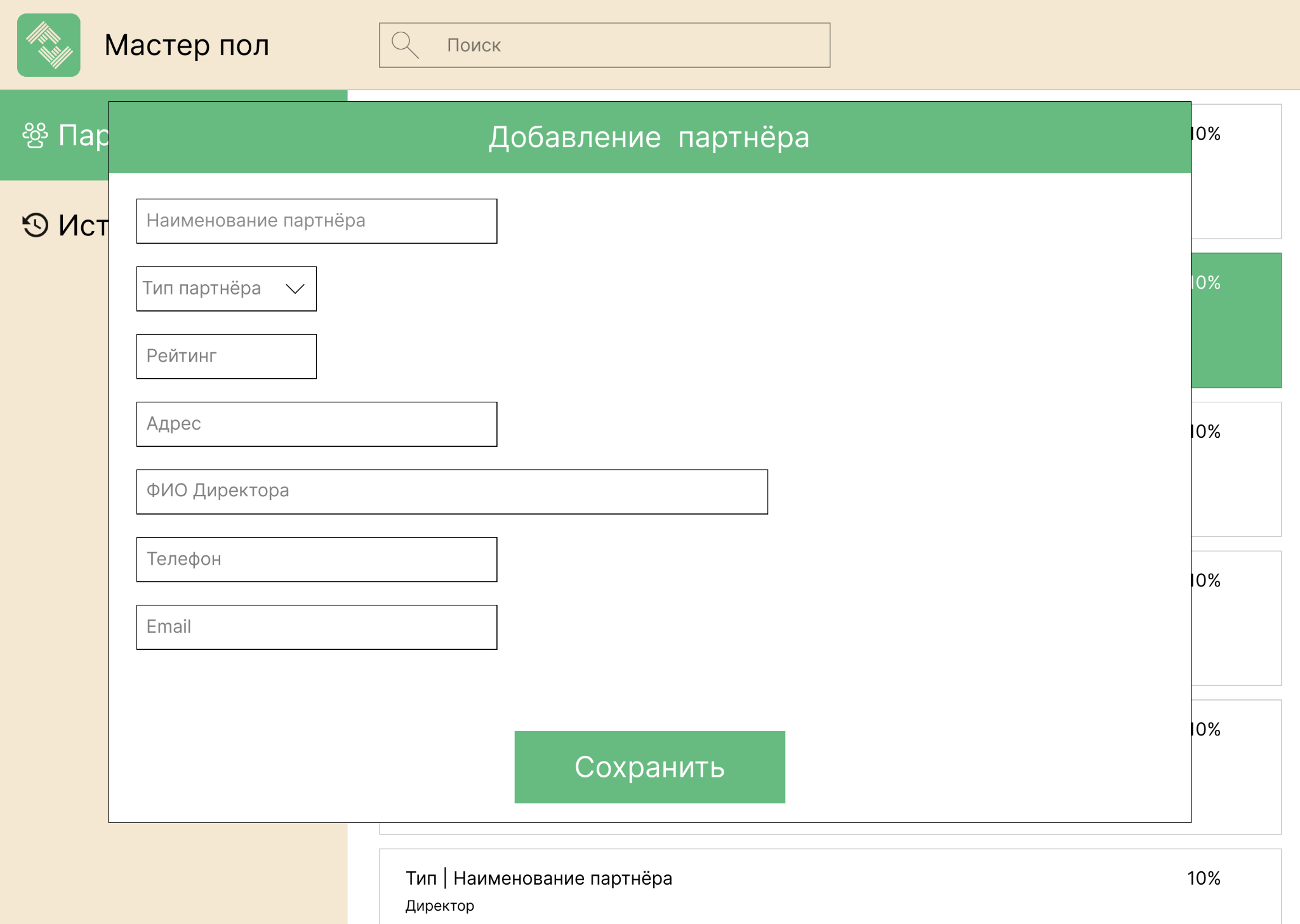
1. Аналитика и отчетность

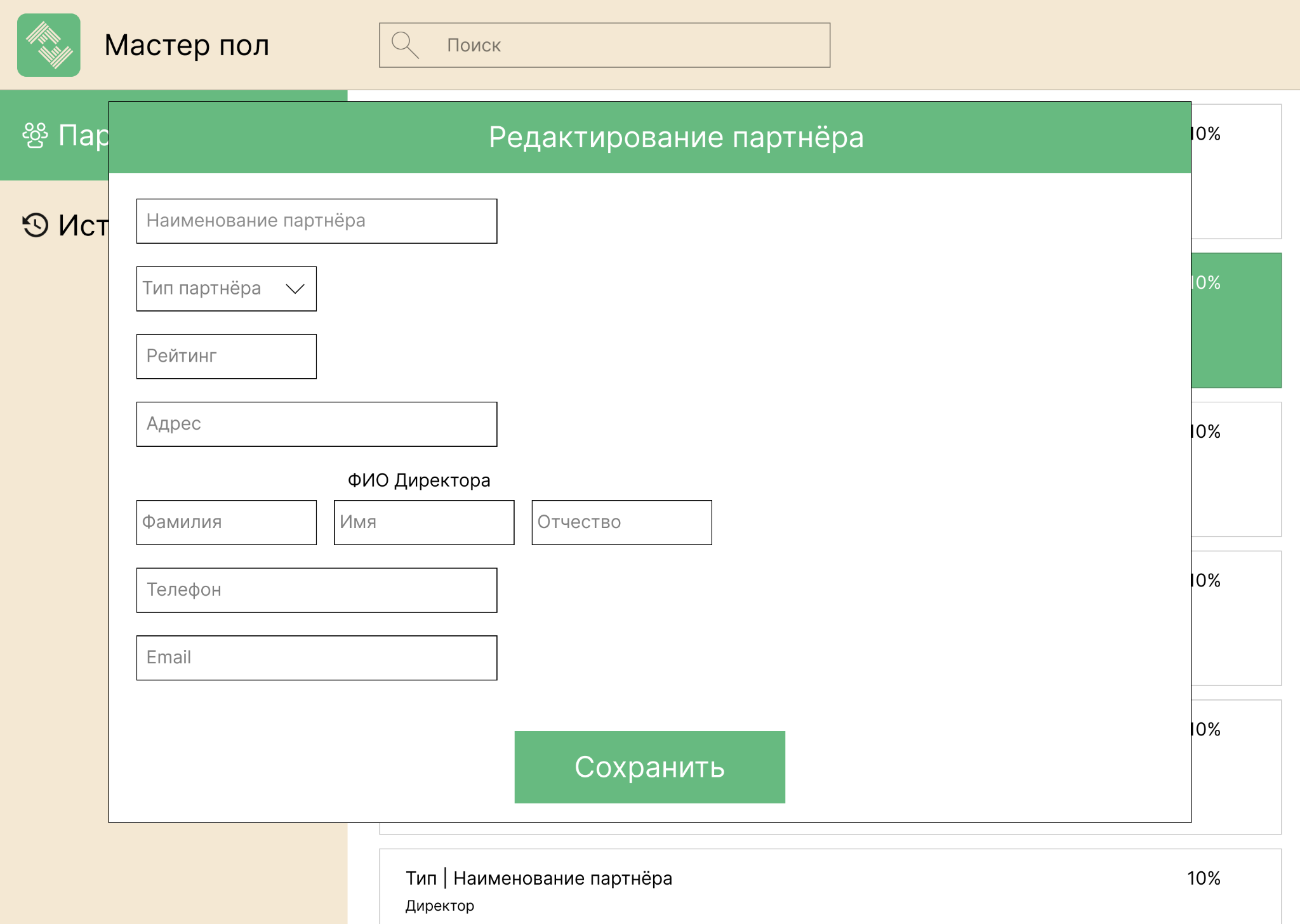
* Система предоставляет менеджерам и аналитикам сводные отчеты о движении материалов, продажах продукции, рейтингах партнеров.
* Генерация отчетов по перемещению сотрудников и эффективности работы цехов.
* Формирование данных о динамике продаж для расчета скидок партнерам.

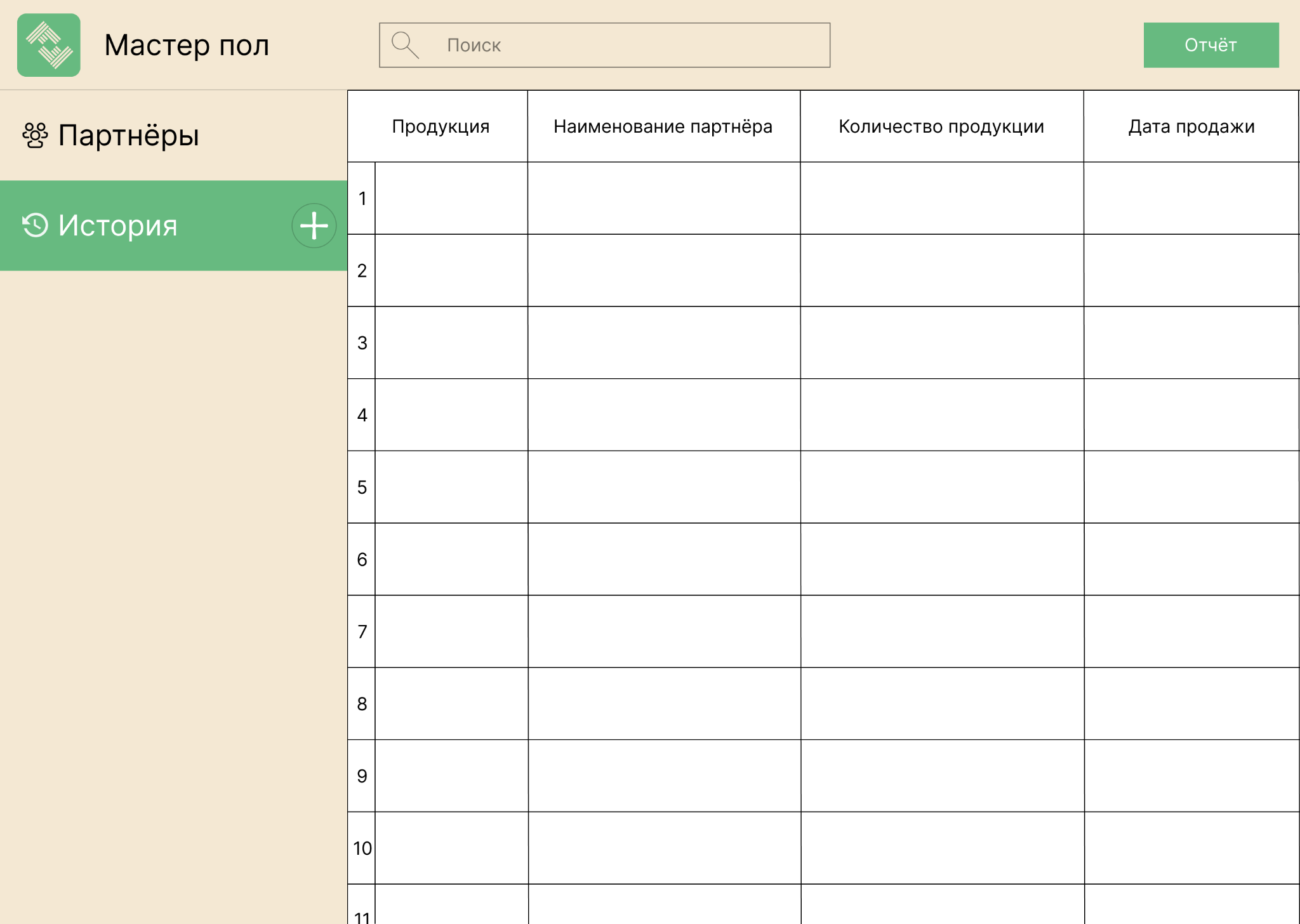
## Разработка макетов пользовательского интерфейса

https://www.figma.com/design/5eMXWlRlSeOEuKSEJsYmR6/%D0%A3%D0%9F-01.01?node-id=0-1&t=igbLFmUVVsL7QAxX-1









## Разработка программного модуля

Создание базы данных:

CREATE TABLE TypeCompany (

id INT PRIMARY KEY,

name VARCHAR(255)

);

CREATE TABLE Partners (

id INT PRIMARY KEY,

type\_partner INT,

company\_name VARCHAR(255),

ur\_adress VARCHAR(255),

inn VARCHAR(50),

director\_name VARCHAR(255),

phone VARCHAR(50),

email VARCHAR(255),

rating INT,

FOREIGN KEY (type\_partner) REFERENCES TypeCompany(id)

);

CREATE TABLE ProductType (

id INT PRIMARY KEY,

name VARCHAR(255),

coefficient FLOAT

);

CREATE TABLE Product (

id INT PRIMARY KEY,

type INT,

description VARCHAR(255),

article INT,

price FLOAT,

size FLOAT,

class INT,

FOREIGN KEY (type) REFERENCES ProductType(id)

);

CREATE TABLE PartnerProduct (

id INT PRIMARY KEY,

id\_product INT,

id\_partner INT,

quantity INT,

date\_of\_sale DATE,

FOREIGN KEY (id\_product) REFERENCES Product(id),

FOREIGN KEY (id\_partner) REFERENCES Partners(id)

);

CREATE TABLE Material (

id INT PRIMARY KEY,

name VARCHAR(255),

defect FLOAT

);

CREATE TABLE MaterialProduct (

id INT PRIMARY KEY,

id\_product INT,

id\_material INT,

FOREIGN KEY (id\_product) REFERENCES Product(id),

FOREIGN KEY (id\_material) REFERENCES Material(id)

);

import sys

from PySide6.QtGui import QIcon

from PySide6.QtWidgets import (

QApplication, QWidget, QVBoxLayout, QHBoxLayout, QLabel, QPushButton, QListWidget, QListWidgetItem, QLineEdit, QTableWidget, QTableWidgetItem,

QDialog, QFormLayout, QSpinBox, QComboBox

)

from PySide6.QtCore import Qt, QSize

from sqlalchemy import create\_engine, Column, Integer, String, Float, ForeignKey, Date

from sqlalchemy.ext.declarative import declarative\_base

from sqlalchemy.orm import relationship, sessionmaker

# Подключение к базе данных

DATABASE\_URI = 'postgresql://postgres:root@localhost:5432/asd'

engine = create\_engine(DATABASE\_URI)

Base = declarative\_base()

# Определение таблиц

class TypeCompany(Base):

\_\_tablename\_\_ = 'typecompany'

id = Column(Integer, primary\_key=True)

name = Column(String(255), nullable=False)

class Partners(Base):

\_\_tablename\_\_ = 'partners'

id = Column(Integer, primary\_key=True)

type\_partner = Column(Integer, ForeignKey('typecompany.id'))

company\_name = Column(String(255), nullable=False)

ur\_adress = Column(String(255), nullable=False)

inn = Column(String(50), nullable=False)

director\_name = Column(String(255), nullable=False)

phone = Column(String(50), nullable=False)

email = Column(String(255), nullable=False)

rating = Column(Integer, nullable=True)

type\_company = relationship("TypeCompany")

class ProductType(Base):

\_\_tablename\_\_ = 'producttype'

id = Column(Integer, primary\_key=True)

name = Column(String(255), nullable=False)

coefficient = Column(Float, nullable=False)

class Product(Base):

\_\_tablename\_\_ = 'product'

id = Column(Integer, primary\_key=True)

type = Column(Integer, ForeignKey('producttype.id'))

description = Column(String(255), nullable=False)

article = Column(Integer, nullable=False)

price = Column(Float, nullable=False)

size = Column(Float, nullable=False)

class\_ = Column(Integer, nullable=False) # 'class' - зарезервированное слово, используем 'class\_'

product\_type = relationship("ProductType")

class PartnerProduct(Base):

\_\_tablename\_\_ = 'partnerproduct'

id = Column(Integer, primary\_key=True)

id\_product = Column(Integer, ForeignKey('product.id'))

id\_partner = Column(Integer, ForeignKey('partners.id'))

quantity = Column(Integer, nullable=False)

date\_of\_sale = Column(Date, nullable=False)

product = relationship("Product")

partner = relationship("Partners")

class Material(Base):

\_\_tablename\_\_ = 'material'

id = Column(Integer, primary\_key=True)

name = Column(String(255), nullable=False)

defect = Column(Float, nullable=True)

class MaterialProduct(Base):

\_\_tablename\_\_ = 'materialproduct'

id = Column(Integer, primary\_key=True)

id\_product = Column(Integer, ForeignKey('product.id'))

id\_material = Column(Integer, ForeignKey('material.id'))

product = relationship("Product")

material = relationship("Material")

# Создание таблиц в базе данных

Base.metadata.create\_all(engine)

Session = sessionmaker(bind=engine)

session = Session()

class MasterApp(QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

# Установка иконки приложения

self.setWindowTitle("Мастер пол")

self.setWindowIcon(QIcon("/home/KHPK.RU/student/Рабочий стол/logo.png")) # Замените на путь к иконке приложения

self.setFixedSize(1440, 1024)

# Главный макет

main\_layout = QVBoxLayout()

main\_layout.setContentsMargins(0, 0, 0, 0)

# Верхняя панель

top\_panel = QWidget()

top\_panel.setStyleSheet("background-color: #F4E8D3; padding: 5px;")

top\_layout = QHBoxLayout(top\_panel)

# Иконка и текст для верхней панели

top\_icon\_label = QLabel()

top\_icon\_label.setPixmap(QIcon("/home/KHPK.RU/student/Рабочий стол/logo.png").pixmap(50, 50))

top\_label = QLabel("Мастер пол")

top\_label.setStyleSheet("font-size: 24px; font-weight: bold; margin-left: 0px;")

search\_box = QLineEdit()

search\_box.setPlaceholderText("Поиск")

search\_box.setFixedWidth(200)

search\_box.setStyleSheet("padding: 5px; margin-left: 10px;")

add\_partner\_button = QPushButton("Добавить партнёра")

add\_partner\_button.setFixedWidth(150)

add\_partner\_button.clicked.connect(self.add\_partner)

# Добавляем элементы в верхнюю панель

top\_layout.addWidget(top\_icon\_label) # Иконка рядом с текстом

top\_layout.addWidget(top\_label)

top\_layout.addWidget(search\_box)

top\_layout.addWidget(add\_partner\_button)

top\_layout.addStretch()

main\_layout.addWidget(top\_panel)

# Панель навигации и содержимого

content\_layout = QHBoxLayout()

left\_panel = QWidget()

left\_panel.setFixedWidth(200)

left\_panel.setStyleSheet("background-color: #F4E8D3;")

left\_layout = QVBoxLayout(left\_panel)

left\_layout.setAlignment(Qt.AlignTop)

# Кнопки навигации с иконками

self.partners\_button = QPushButton("Партнёры")

self.partners\_button.setIcon(QIcon("/home/KHPK.RU/student/Рабочий стол/partner.png"))

self.partners\_button.setCheckable(True)

self.partners\_button.setChecked(True)

self.partners\_button.clicked.connect(self.select\_partners\_tab)

self.history\_button = QPushButton("История")

self.history\_button.setIcon(QIcon("/home/KHPK.RU/student/Рабочий стол/history.png"))

self.history\_button.setCheckable(True)

self.history\_button.clicked.connect(self.select\_history\_tab)

self.update\_tab\_styles()

left\_layout.addWidget(self.partners\_button)

left\_layout.addWidget(self.history\_button)

right\_panel = QWidget()

self.right\_layout = QVBoxLayout(right\_panel)

self.right\_layout.setContentsMargins(10, 10, 10, 0)

self.partners\_list = QListWidget()

self.partners\_list.setSpacing(5)

self.partners\_list.setStyleSheet("border: none;")

self.partners\_list.itemClicked.connect(self.highlight\_selected\_partner)

self.partners\_list.itemDoubleClicked.connect(self.edit\_partner)

self.load\_partners\_from\_db()

self.history\_table = QTableWidget()

self.history\_table.setColumnCount(4)

self.history\_table.setHorizontalHeaderLabels(["Продукция", "Наименование партнёра", "Количество продукции", "Дата продажи"])

self.right\_layout.addWidget(self.partners\_list)

content\_layout.addWidget(left\_panel)

content\_layout.addWidget(right\_panel)

main\_layout.addLayout(content\_layout)

self.setLayout(main\_layout)

def load\_partners\_from\_db(self):

self.partners\_list.clear()

partners = session.query(Partners).all()

for partner in partners:

item = QListWidgetItem()

item\_widget = self.create\_partner\_item(partner)

item.setSizeHint(QSize(item\_widget.sizeHint().width(), item\_widget.sizeHint().height() + 20))

self.partners\_list.addItem(item)

self.partners\_list.setItemWidget(item, item\_widget)

item.setData(Qt.UserRole, partner) # Сохраняем объект партнёра в item

def create\_partner\_item(self, partner=None):

item\_widget = QWidget()

layout = QVBoxLayout(item\_widget)

layout.setContentsMargins(10, 10, 10, 10)

if partner:

type\_name = partner.type\_company.name if partner.type\_company else "Неизвестный тип"

type\_label = QLabel(f"{type\_name} | {partner.company\_name}")

director\_label = QLabel(f"Директор: {partner.director\_name}")

phone\_label = QLabel(f"Телефон: {partner.phone}")

rating\_label = QLabel(f"Рейтинг: {partner.rating}")

item\_widget.setProperty("partner\_id", partner.id) # Добавляем ID партнёра как свойство

else:

type\_label = QLabel("Тип | Наименование партнёра")

director\_label = QLabel("Директор")

phone\_label = QLabel("+7 223 322 22 32")

rating\_label = QLabel("Рейтинг: 10")

layout.addWidget(type\_label)

layout.addWidget(director\_label)

layout.addWidget(phone\_label)

layout.addWidget(rating\_label)

item\_widget.setStyleSheet("background-color: #FFFFFF; border: 1px solid #F4E8D3; padding: 5px;")

return item\_widget

def update\_tab\_styles(self):

self.partners\_button.setStyleSheet("padding: 10px; background-color: #67BA80;" if self.partners\_button.isChecked() else "background-color: #FFFFFF; color: black;")

self.history\_button.setStyleSheet("padding: 10px; background-color: #67BA80;" if self.history\_button.isChecked() else "background-color: #FFFFFF; color: black;")

def select\_partners\_tab(self):

self.partners\_button.setChecked(True)

self.history\_button.setChecked(False)

self.update\_tab\_styles()

self.right\_layout.removeWidget(self.history\_table)

self.history\_table.setParent(None)

self.right\_layout.addWidget(self.partners\_list)

def select\_history\_tab(self):

self.history\_button.setChecked(True)

self.partners\_button.setChecked(False)

self.update\_tab\_styles()

self.right\_layout.removeWidget(self.partners\_list)

self.partners\_list.setParent(None)

self.right\_layout.addWidget(self.history\_table)

def highlight\_selected\_partner(self, item):

for i in range(self.partners\_list.count()):

widget = self.partners\_list.itemWidget(self.partners\_list.item(i))

widget.setStyleSheet("background-color: #FFFFFF; border: 1px solid #F4E8D3; padding: 5px;")

selected\_widget = self.partners\_list.itemWidget(item)

selected\_widget.setStyleSheet("background-color: #67BA80; color: #FFFFFF; border: 1px solid #F4E8D3; padding: 5px;")

def edit\_partner(self, item):

partner = item.data(Qt.UserRole)

self.show\_partner\_edit\_dialog(partner)

def show\_partner\_edit\_dialog(self, partner):

dialog = QDialog(self)

dialog.setWindowTitle(f"Редактировать партнёра: {partner.company\_name}")

form\_layout = QFormLayout(dialog)

name\_edit = QLineEdit(partner.company\_name)

ur\_adress\_edit = QLineEdit(partner.ur\_adress)

inn\_edit = QLineEdit(partner.inn)

director\_name\_edit = QLineEdit(partner.director\_name)

phone\_edit = QLineEdit(partner.phone)

email\_edit = QLineEdit(partner.email)

rating\_edit = QSpinBox()

rating\_edit.setValue(partner.rating if partner.rating else 0)

# Выпадающий список типов компании

type\_combo = QComboBox()

types = session.query(TypeCompany).all()

for type\_ in types:

type\_combo.addItem(type\_.name, type\_.id)

type\_combo.setCurrentIndex(type\_combo.findData(partner.type\_partner))

# Добавляем выпадающий список для типа партнёра с данными из БД

partner\_type\_combo = QComboBox()

types = session.query(TypeCompany).all()

for type\_ in types:

partner\_type\_combo.addItem(type\_.name, type\_.id)

form\_layout.addRow("Наименование партнёра:", name\_edit)

form\_layout.addRow("Тип партнёра:", partner\_type\_combo)

form\_layout.addRow("Юридический адрес:", ur\_adress\_edit)

form\_layout.addRow("ИНН:", inn\_edit)

form\_layout.addRow("Имя директора:", director\_name\_edit)

form\_layout.addRow("Телефон:", phone\_edit)

form\_layout.addRow("Email:", email\_edit)

form\_layout.addRow("Рейтинг:", rating\_edit)

save\_button = QPushButton("Сохранить")

save\_button.clicked.connect(lambda: self.save\_partner\_changes(dialog, partner, type\_combo, partner\_type\_combo, name\_edit, ur\_adress\_edit, inn\_edit, director\_name\_edit, phone\_edit, email\_edit, rating\_edit))

form\_layout.addWidget(save\_button)

dialog.exec()

def save\_partner\_changes(self, dialog, partner, type\_combo, partner\_type\_combo, name\_edit, ur\_adress\_edit, inn\_edit, director\_name\_edit, phone\_edit, email\_edit, rating\_edit):

partner.type\_partner = type\_combo.currentData() # Сохраняем тип партнёра

partner.company\_name = name\_edit.text()

partner.ur\_adress = ur\_adress\_edit.text()

partner.inn = inn\_edit.text()

partner.director\_name = director\_name\_edit.text()

partner.phone = phone\_edit.text()

partner.email = email\_edit.text()

partner.rating = rating\_edit.value()

# Сохраняем изменения в базе данных

session.commit()

dialog.accept()

self.load\_partners\_from\_db() # Обновляем список партнёров

def add\_partner(self):

dialog = QDialog(self)

dialog.setWindowTitle("Добавить партнёра")

form\_layout = QFormLayout(dialog)

type\_combo = QComboBox()

types = session.query(TypeCompany).all()

for type\_ in types:

type\_combo.addItem(type\_.name, type\_.id)

name\_edit = QLineEdit()

ur\_adress\_edit = QLineEdit()

inn\_edit = QLineEdit()

director\_name\_edit = QLineEdit()

phone\_edit = QLineEdit()

email\_edit = QLineEdit()

rating\_edit = QSpinBox()

rating\_edit.setMinimum(0)

# Добавляем выпадающий список для типа партнёра с данными из БД

partner\_type\_combo = QComboBox()

types = session.query(TypeCompany).all()

for type\_ in types:

partner\_type\_combo.addItem(type\_.name, type\_.id)

form\_layout.addRow("Наименование партнёра:", name\_edit)

form\_layout.addRow("Тип партнёра:", partner\_type\_combo)

form\_layout.addRow("Юридический адрес:", ur\_adress\_edit)

form\_layout.addRow("ИНН:", inn\_edit)

form\_layout.addRow("Имя директора:", director\_name\_edit)

form\_layout.addRow("Телефон:", phone\_edit)

form\_layout.addRow("Email:", email\_edit)

form\_layout.addRow("Рейтинг:", rating\_edit)

add\_button = QPushButton("Добавить")

add\_button.clicked.connect(lambda: self.save\_new\_partner(dialog, type\_combo, partner\_type\_combo, name\_edit, ur\_adress\_edit, inn\_edit, director\_name\_edit, phone\_edit, email\_edit, rating\_edit))

form\_layout.addWidget(add\_button)

dialog.exec()

def save\_partner\_changes(self, dialog, partner, type\_combo, partner\_type\_combo, name\_edit, ur\_adress\_edit, inn\_edit, director\_name\_edit, phone\_edit, email\_edit, rating\_edit):

# Обновляем данные партнёра

partner.type\_partner = type\_combo.currentData() # Сохраняем ID типа компании

partner.company\_name = name\_edit.text()

partner.ur\_adress = ur\_adress\_edit.text()

partner.inn = inn\_edit.text()

partner.director\_name = director\_name\_edit.text()

partner.phone = phone\_edit.text()

partner.email = email\_edit.text()

partner.rating = rating\_edit.value()

# Сохраняем ID типа партнёра (сейчас из комбобокса)

partner.type\_partner = partner\_type\_combo.currentData() # Сохраняем ID типа партнёра

# Сохраняем изменения в базе данных

session.commit()

# Закрываем диалог и обновляем отображение

dialog.accept()

self.load\_partners\_from\_db()

if \_\_name\_\_ == "\_\_main\_\_":

app = QApplication(sys.argv)

master\_app = MasterApp()

master\_app.show()

sys.exit(app.exec())