государственное бюджетное профессиональное образовательное учреждение

**«Пермский химико-технологический техникум»**

Специальность 09.02.07

«Информационная система и программирование»

КУРСОВОЙ ПРОЕКТ

**Междисциплинарный курс:** «МДК 11.11 Технология разработки и защиты баз данных»

**Тема:** «Проектирование базы данных для сотрудников фондовой биржи FondFox»

Выполнил студент гр. ИС-21-2

(Д.С. Мурзин)

Руководитель проекта:

(Л.Ф. Абдалова)

2024

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](file:///C:\Users\mmurz\Downloads\Telegram%20Desktop\Курсовая_Яковенко.docx#_Toc169976608)

[1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 5](file:///C:\Users\mmurz\Downloads\Telegram%20Desktop\Курсовая_Яковенко.docx#_Toc169976609)

[1.1. Описание предметной области 5](file:///C:\Users\mmurz\Downloads\Telegram%20Desktop\Курсовая_Яковенко.docx#_Toc169976610)

[1.1.1 Построение функциональной диаграммы 7](file:///C:\Users\mmurz\Downloads\Telegram%20Desktop\Курсовая_Яковенко.docx#_Toc169976611)

[1.1.2 Построение диаграммы прецедентов 9](file:///C:\Users\mmurz\Downloads\Telegram%20Desktop\Курсовая_Яковенко.docx#_Toc169976612)

[1.2. Средства технологии проектирования. Выбор СУБД 10](file:///C:\Users\mmurz\Downloads\Telegram%20Desktop\Курсовая_Яковенко.docx#_Toc169976613)

[1.3. Построение концептуальной модели предметной области 12](file:///C:\Users\mmurz\Downloads\Telegram%20Desktop\Курсовая_Яковенко.docx#_Toc169976614)

[1.4. Проектирование логической структуры Базы данных 13](file:///C:\Users\mmurz\Downloads\Telegram%20Desktop\Курсовая_Яковенко.docx#_Toc169976615)

[2. ПРАКТИЧЕСКАЯ ЧАСТЬ 14](file:///C:\Users\mmurz\Downloads\Telegram%20Desktop\Курсовая_Яковенко.docx#_Toc169976616)

[2.1. Проектирование физической структуры базы данных 14](file:///C:\Users\mmurz\Downloads\Telegram%20Desktop\Курсовая_Яковенко.docx#_Toc169976617)

[2.2. Организация ввода-вывода данных в БД 15](file:///C:\Users\mmurz\Downloads\Telegram%20Desktop\Курсовая_Яковенко.docx#_Toc169976618)

[2.3. Организация манипулирования данными БД 18](file:///C:\Users\mmurz\Downloads\Telegram%20Desktop\Курсовая_Яковенко.docx#_Toc169976619)

[2.4. Реализация запросов и получение отчетов 19](file:///C:\Users\mmurz\Downloads\Telegram%20Desktop\Курсовая_Яковенко.docx#_Toc169976620)

[2.5 Администрирование базы данных 20](file:///C:\Users\mmurz\Downloads\Telegram%20Desktop\Курсовая_Яковенко.docx#_Toc169976621)

[2.6. Разработка интерфейса приложения 21](file:///C:\Users\mmurz\Downloads\Telegram%20Desktop\Курсовая_Яковенко.docx#_Toc169976622)

[ЗАКЛЮЧЕНИЕ 28](file:///C:\Users\mmurz\Downloads\Telegram%20Desktop\Курсовая_Яковенко.docx#_Toc169976623)

**ВВЕДЕНИЕ**

В наше время финансов, когда ежедневно появляются новые компании и акции становятся все популярнее, растет потребность в специализированных платформах, которые могут обеспечить качественное обслуживание сотрудников финансовых организаций. В этом контексте становится важным разрабатывать решения, направленные на внутренние потребности компаний, работающих на фондовом рынке.

Приложение «FondFox» — это инновационная платформа, созданная специально для сотрудников фондовой биржи. Основная цель приложения — предоставить удобные и эффективные инструменты для работы с акциями и улучшить качество взаимодействия внутри финансовых организаций.

Особенность «FondFox» заключается в том, что оно ориентировано исключительно на сотрудников финансовых организаций и предоставляет им такие возможности, как просмотр списка акций компаний, добавление акций в персональный список, отслеживание состояния своего счёта и обмен акциями с другими сотрудниками.

Актуальность приложения для сотрудников фондовой биржи «FondFox» обусловлена стремительным развитием технологий и возрастающей потребностью в удобных и надежных инструментах для людей в области финансов. Приложение стремится предоставить сотрудникам доступ к актуальной информации, что позволяет им принимать обоснованные решения и увеличивать свою прибыль.

Целью данного курсового проекта является введение «FondFox» как внутренней платформы для сотрудников финансовых организаций, направленное на решение ряда ключевых задач. Основные задачи включают:

1. Анализ предметной области.
2. Разработка логической и концептуальной модели базы данных (БД).
3. Реализация логической модели в конкретной системе управления базами данных (СУБД)
4. Разработка приложения и внедрение функционала для повышения эффективности работы сотрудников.
5. Создание инструментов для улучшения качества аналитики.
6. Обеспечение оптимизации взаимодействия внутри команды путем создания среды, в которой сотрудники могут легко обмениваться информацией и сотрудничать для достижения общих целей.
7. Внедрение механизмов поддержки профессионального развития команды.
8. Разработка технической документации.

Таким образом, приложение «FondFox» активно поддерживает профессионалов, работающих в сфере финансов, предлагая им все необходимые инструменты и ресурсы для успешной работы.

**ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ**

* 1. **Описание предметной области.**

Предметной областью является фондовая биржа «FondFox». Приложение для сотрудников фондовой биржи представляет собой специализированную платформу, предназначенную для поддержки брокеров и оптимизации управления акциями и балансами внутри компании.

Известны следующие данные о пользователях: ФИО, почта, счет, характеристика и фото. Эти данные помогают в идентификации и управлении учетными записями сотрудников, обеспечивая точность и безопасность информации.

Для каждого брокера создается учетная запись, в которой указываются их данные, включая логин и пароль для входа в систему. Это позволяет брокерам легко и безопасно получать доступ к платформе. Брокеры могут просматривать акции компании, добавлять их в свой персональный список и отслеживать состояние своего счёта. Это обеспечивает удобный доступ к актуальной информации об акциях и возможность оперативного принятия решений.

Каждый брокер закреплен за отдельным отделом, за которым следит старший брокер. Старший брокер выполняет важную роль в управлении командой, имея возможность добавлять и удалять сотрудников из отдела. Это позволяет старшим брокерам эффективно контролировать работу своих подчиненных, обеспечивая слаженность и продуктивность команды.

Администратор платформы играет ключевую роль в поддержании функционирования системы. Администратор выполняет функции добавления акций в приложение, а также может добавлять или удалять отделы и сотрудников. Эта роль включает в себя ответственность за актуализацию данных и структуру компании, что способствует поддержанию порядка и эффективности работы на платформе.

Приложение данной фондовой биржи организует работу с различными акциями, предоставляя сотрудникам возможность просматривать списки акций, добавлять их в персональный список, отслеживать состояние своего счёта и обмениваться акциями с другими сотрудниками. Каждая акция имеет своё название, код и текущую стоимость, что позволяет брокерам быть в курсе всех изменений и тенденций на рынке компании.

В случае необходимости, учетная запись сотрудника на платформе «FondFox» может быть деактивирована по решению администрации или по запросу работодателя. Это предусмотрено для защиты прав и интересов всех сторон и обеспечения гармоничной рабочей среды. Такой подход позволяет поддерживать высокие стандарты безопасности и доверия в работе с финансовыми данными.

* + 1. **Построение функциональной диаграммы.**

Приложение должно обеспечивать взаимодействие пользователей с СУБД, предоставляя возможность получения и записи данных в базу данных. Основной функцией приложения является обеспечение эффективного доступа к информации об акциях и учетных записях сотрудников фондовой биржи «FondFox» посредством взаимодействия с СУБД. Приложение также должно поддерживать ограничения, основанные на разрешениях пользователей, обеспечивая безопасность и конфиденциальность данных.

В рамках данного курсового проекта были определены требования к функциональным характеристикам системы. Эти требования описывают функциональность и возможности разрабатываемого приложения, включая просмотр списка акций, добавление акций в персональный список, отслеживание состояния счёта и обмен акциями. Далее будут представлены диаграммы, иллюстрирующие эти требования. Цель состоит в обеспечении эффективной работы системы и удовлетворении потребностей пользователей.

Основные функции системы представлены на функциональной диаграмме (Рис.1, Рис. 2):

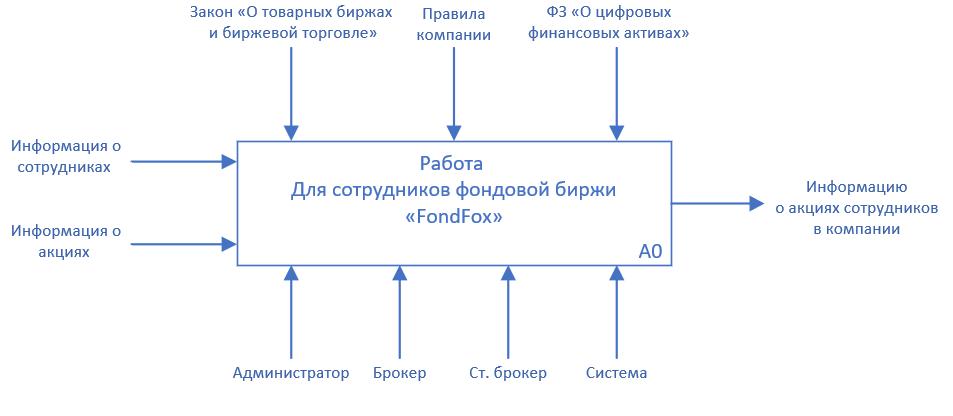


Рис 1. Функциональная диаграмма 1

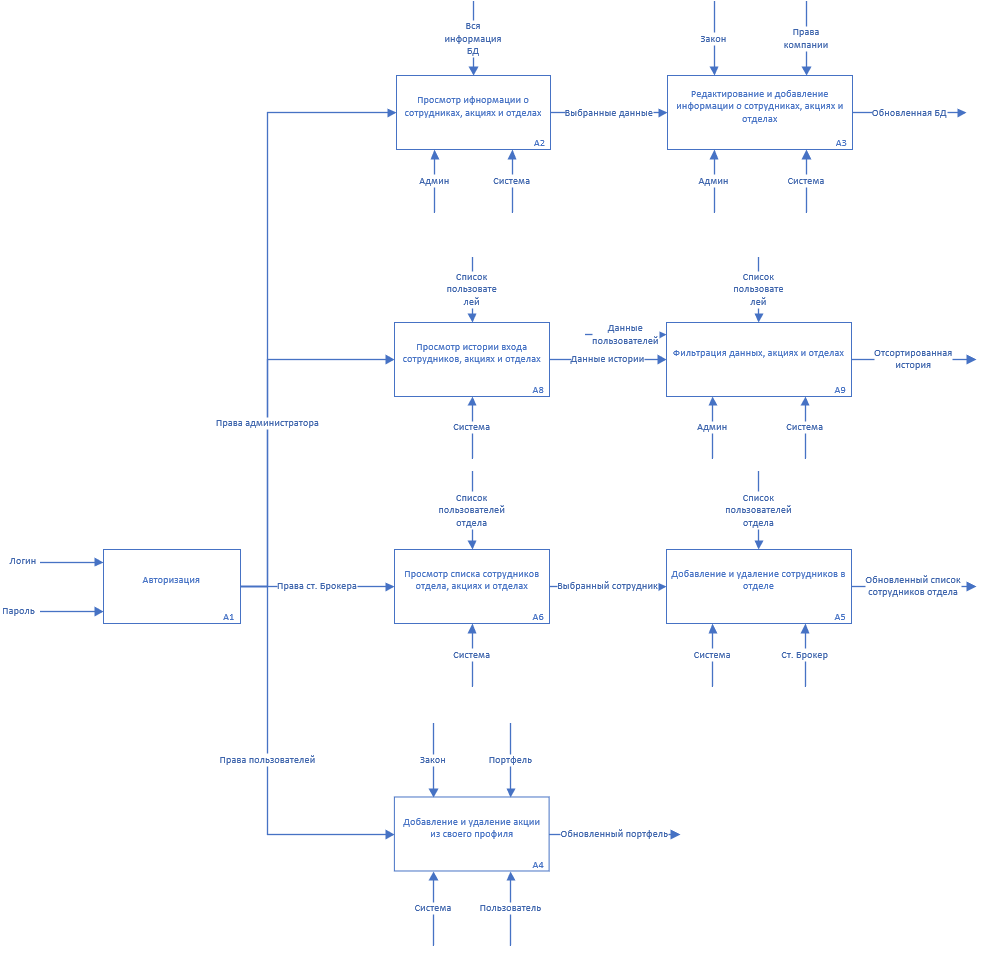


Рис 2. Функциональная диаграмма 2

**1.1.2. Построение диаграммы прецедентов**

Для выполнения данных требований функции системы будут распределены между СУБД и программным кодом. Доступные функции для различных ролей в системе представлены на диаграмме прецедентов (Рис.3):

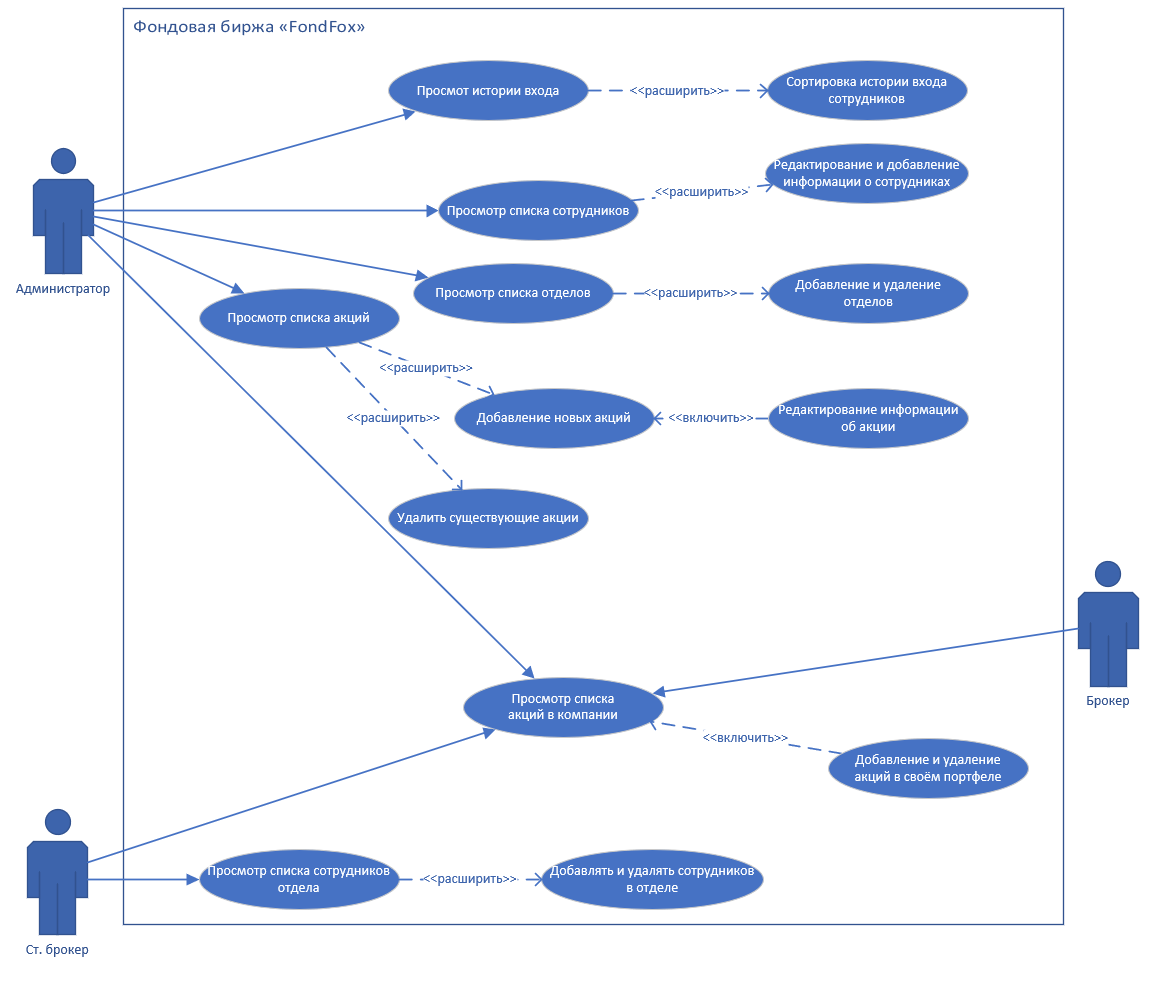


Рис 3. Диаграмма прецедентов

* 1. **Средства технологии проектирования. Выпор СУБД**

Для проектирования базы данных мы предпочитаем использовать семантическую модель, так как её гибкость предоставляет возможность эффективно решать не только уже поставленные задачи, но и те, которые могут возникнуть в будущем. Важно отметить, что количество сущностей и связей, составляющих семантические модели, обычно невелико, что делает такие модели простыми и легкими для восприятия.

ER-диаграмма представляет собой тип структурной схемы, которая широко используется при проектировании баз данных. ERD включает в себя различные символы и соединители, визуализирующие два основных аспекта: ключевые объекты системы и их взаимосвязи. Именно по этой причине её называют диаграммой «Сущность-Взаимоотношение» (ERD). ER-диаграммы включают несколько основных компонентов: Сущности, представляющие объекты, информация о которых хранится в базе данных; Атрибуты, которые являются характеристиками сущностей и описывают их свойства или особенности; и Взаимоотношения, которые описывают связи и взаимодействия между различными сущностями. Для создания ER-диаграмм обычно используются специализированные инструменты для проектирования баз данных, такие как Lucidchart, Visio, ERwin и другие. Существует несколько различных нотаций для ER-диаграмм, включая нотации Чена и Баркера.

При выборе технологии проектирования для курсового проекта одной из ключевых задач является выбор системы управления базами данных (СУБД). На рынке представлено множество различных СУБД, каждая из которых обладает своими особенностями, преимуществами и недостатками. Выбор подходящей СУБД является важным шагом, так как от этого зависит эффективность и надежность будущей базы данных.

Рассматривая такие СУБД, как MariaDB, SQLite, MongoDB и Redis, я остановил свой выбор на Microsoft SQL Server. Этот продукт, разработанный Microsoft, широко используется в предприятиях разного масштаба. Microsoft SQL Server отличается удобным интерфейсом, надежностью и целостностью данных, а также обширными возможностями для работы с базами данных. Также мой личный опыт работы с Microsoft SQL Server также сыграл свою роль в принятии этого решения.

Для приложения на платформе Windows необходимо обеспечить следующие требования:

* Версия операционной системы должна быть одной из следующих: Windows 98, Me, Xp, Vista, 7, 8, 8.1, 10.
* Для обеспечения производительности приложения требуется процессор с тактовой частотой не менее 1,8 ГГц или более быстрый процессор x64. Рекомендуется использовать четырехъядерные процессоры или лучше.
* Оперативная память должна быть не менее 2 ГБ для запуска приложения, для стандартных профессиональных решений рекомендуется 4 ГБ ОЗУ или больше.
* На жестком диске должно быть не менее 200 КБ свободного места для установки и запуска приложения.
* Для оптимальной работы приложения рекомендуется использовать видеокарту с минимальным разрешением WXGA (1366 на 768 пикселей). Для достижения наилучшего качества изображения и удобства работы рекомендуется использовать видеокарту с разрешением 1920 на 1080 пикселей или выше.
* СУБД MySQL Server (версия 2019 года).
* Дополнительные программные компоненты для .NET Framework.

**1.3 Построение концептуальной модели.**

Концептуальная модель базы данных представляет собой описание основных сущностей и их взаимосвязей, не зависящее от конкретной модели базы данных или синтаксиса СУБД. Эта модель помогает лучше понять структуру и логику данных в рамках предметной области. Она является важным этапом в проектировании базы данных, так как позволяет формализовать знания о системе и уточнить требования. Разработка такой модели способствует выявлению всех необходимых элементов и их взаимосвязей, что упрощает дальнейшие этапы проектирования.

На основе описания предметной области была создана концептуальная модель базы данных (см. Рис. 4):

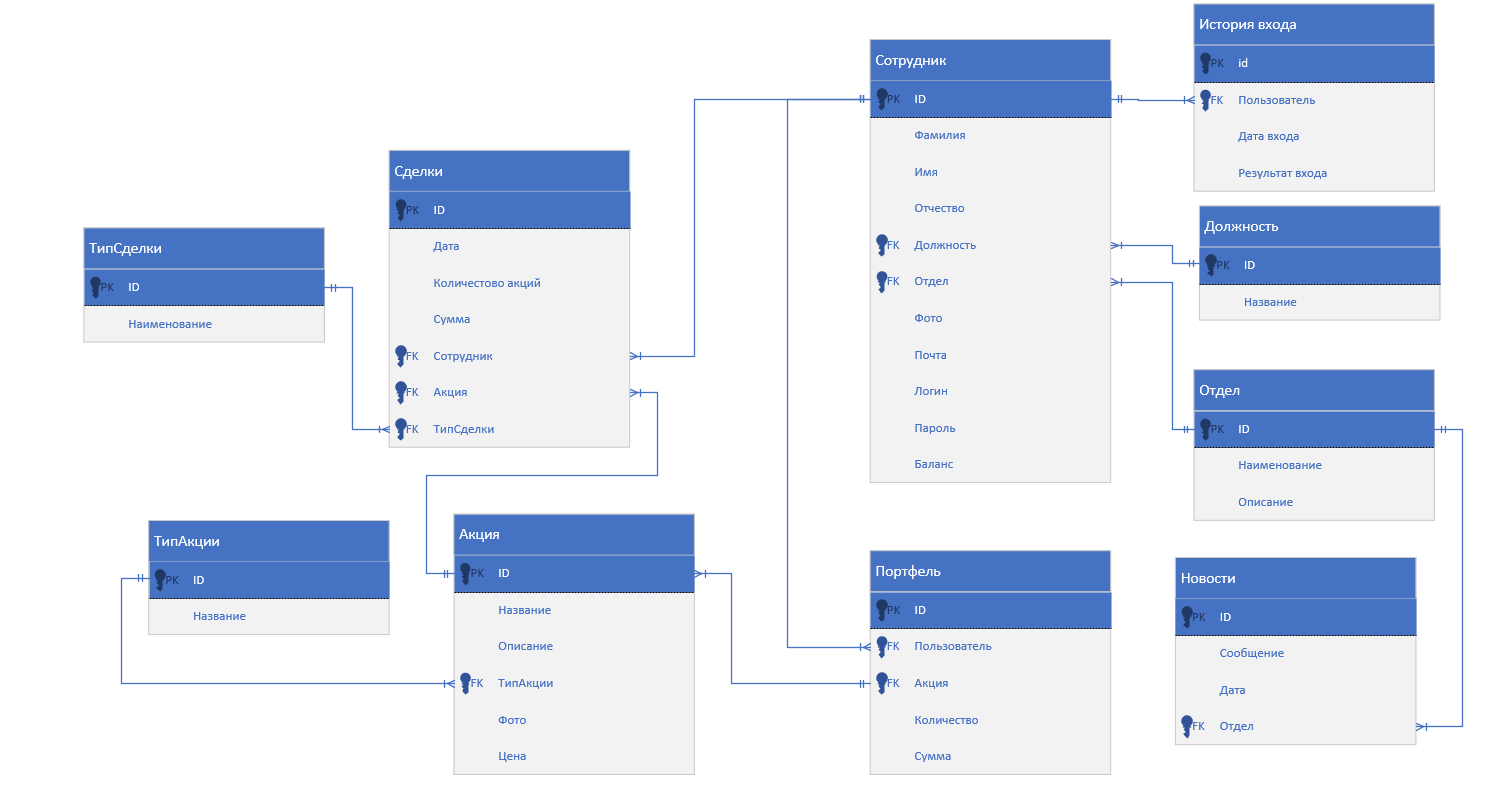


Рис.4 Концептуальная модель

**1.4 Проектирование логической структуры Базы**

Логическая модель базы данных представляет собой начальный прототип будущей базы данных. Она разрабатывается в терминах информационных единиц, но не привязывается к конкретной СУБД. Этот этап проектирования позволяет сформировать абстрактное представление о данных и их структурах, не ограничиваясь техническими деталями конкретной системы управления базами данных.

На основе описания предметной области была создана логическая модель базы данных (см. Рис. 5):

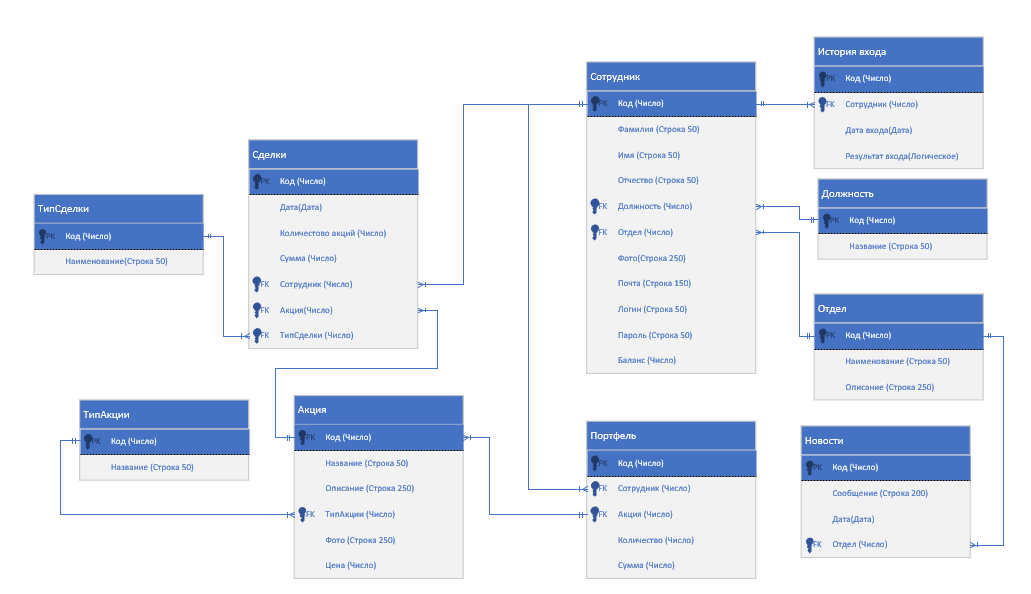


Рис.5 Логическая модель

1. **ПРАКТИЧЕСКАЯ ЧАСТЬ**

**2.1 Проектирование физической структуры базы**

Физическая структура базы данных описывает её организацию в соответствии с требованиями третьей нормальной формы. Эта модель будет использоваться в дальнейшем для разработки приложения. Она основана на логической модели данных, которая была преобразована в конкретные таблицы, колонки, ключи и связи.

На основе описания предметной области была создана физическая модель базы данных (см. Рис. 6):

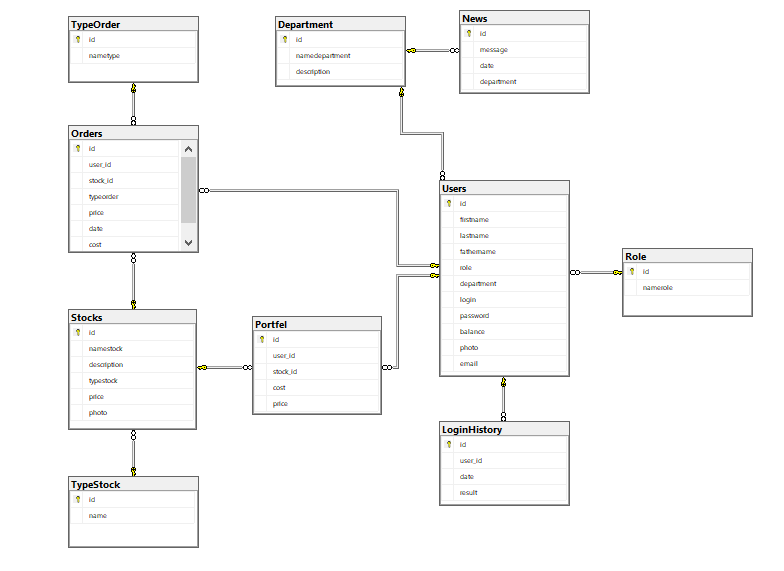


Рис.6 Физическая модель

**2.2. Организация ввода-вывода данных в базу данных**

На окне Торговой площадки есть страница акций, а также страница добавление акции в базу данных (см. Рис 7). Добавление доступно только администратору. (см. Рис 8).

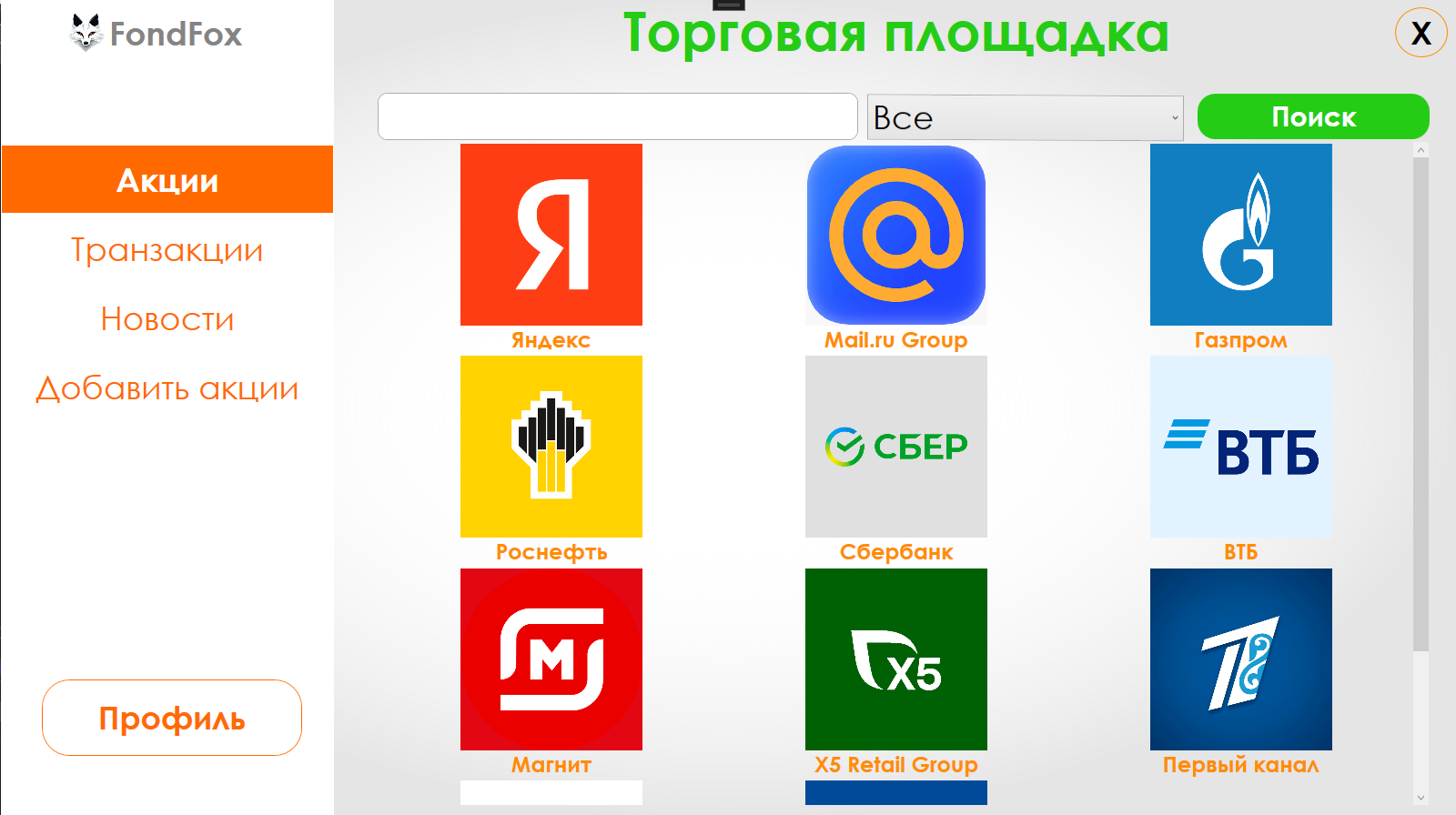


Рис.7 Страница всех акций

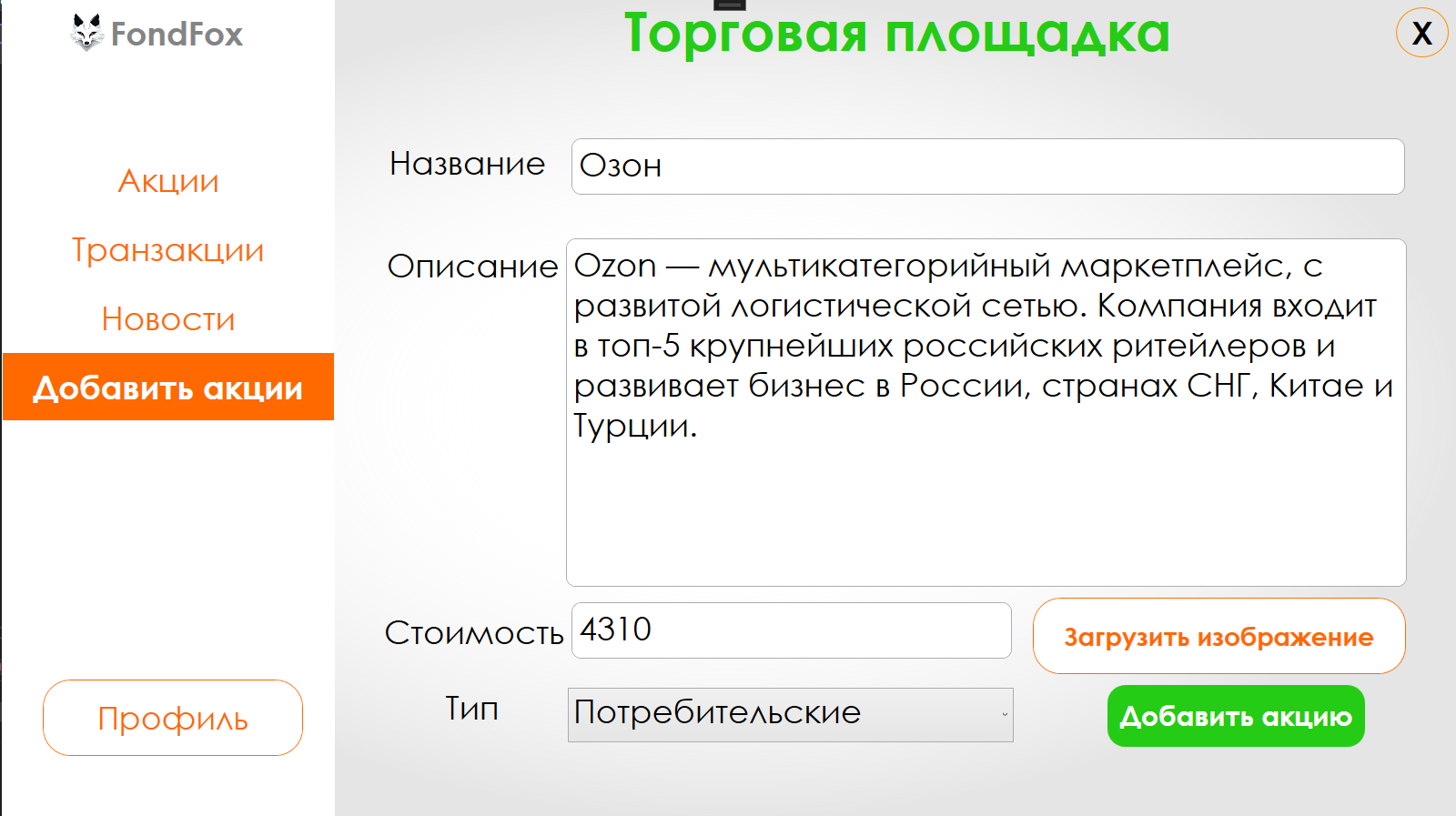


Рис.8 Страница добавления акций

Есть окно добавления сотрудников, которое доступно только администратору. Здесь можно увидеть список всех сотрудников, а также добавить нового или удалить старого по ID (см. Рис 9). При удалении сотрудника программа уточнит намеренность действий (см. Рис 10).

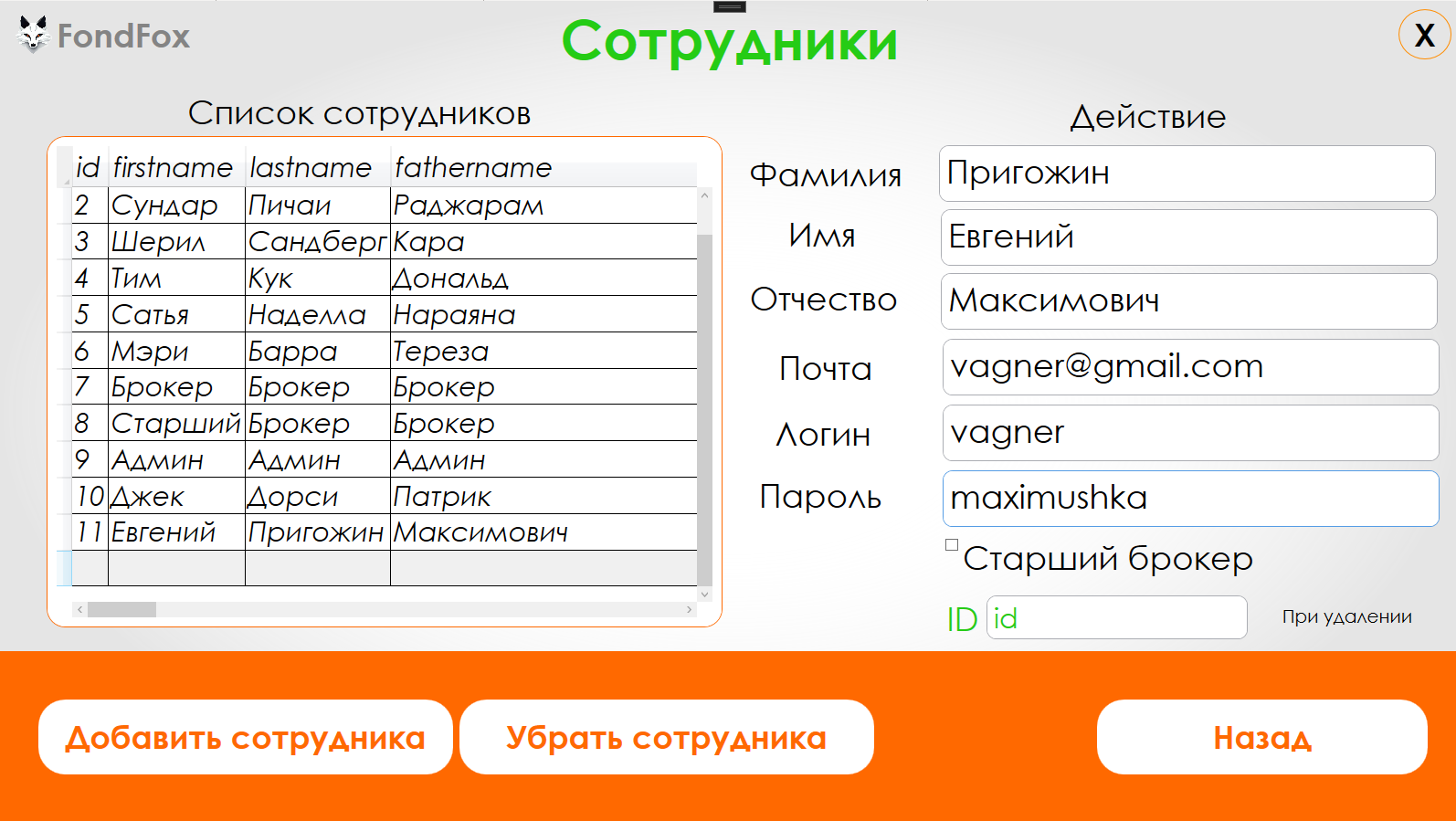


Рис.9 Окно добавления сотрудника

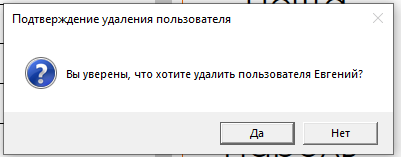


Рис 10 Сообщение о намеренности действий

Все пользователи и акции записываются в базу данных (см. Рис 11,12)

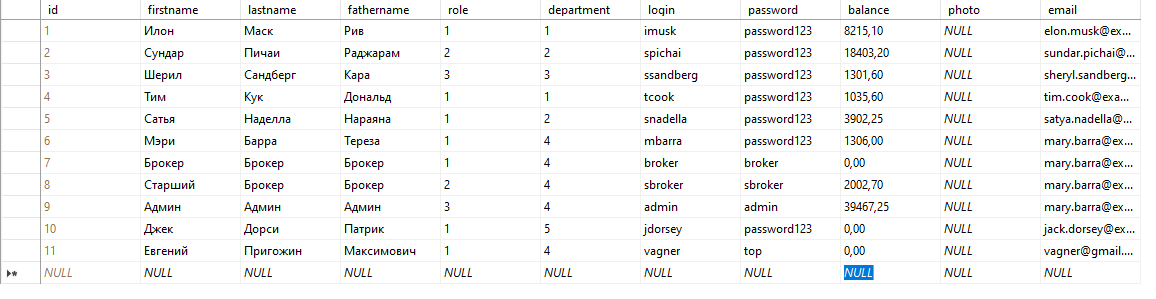


Рис 11 Запись пользователей в базу данных

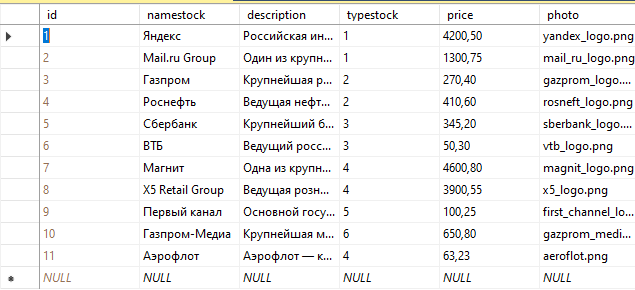


Рис 12 Запись акций в базу данных

**2.3. Организация манипулирования данными**

Старший брокер имеет доступ к своему отделу и может изменять у пользователя статус отдела – добавлять его в свой отдел или удалить из отдела, а на тех пользователях, которые заняты кнопки просто блокируются (см. Рис 13). Также показывается сообщение о успешном добавлении сотрудника (см. Рис 14).

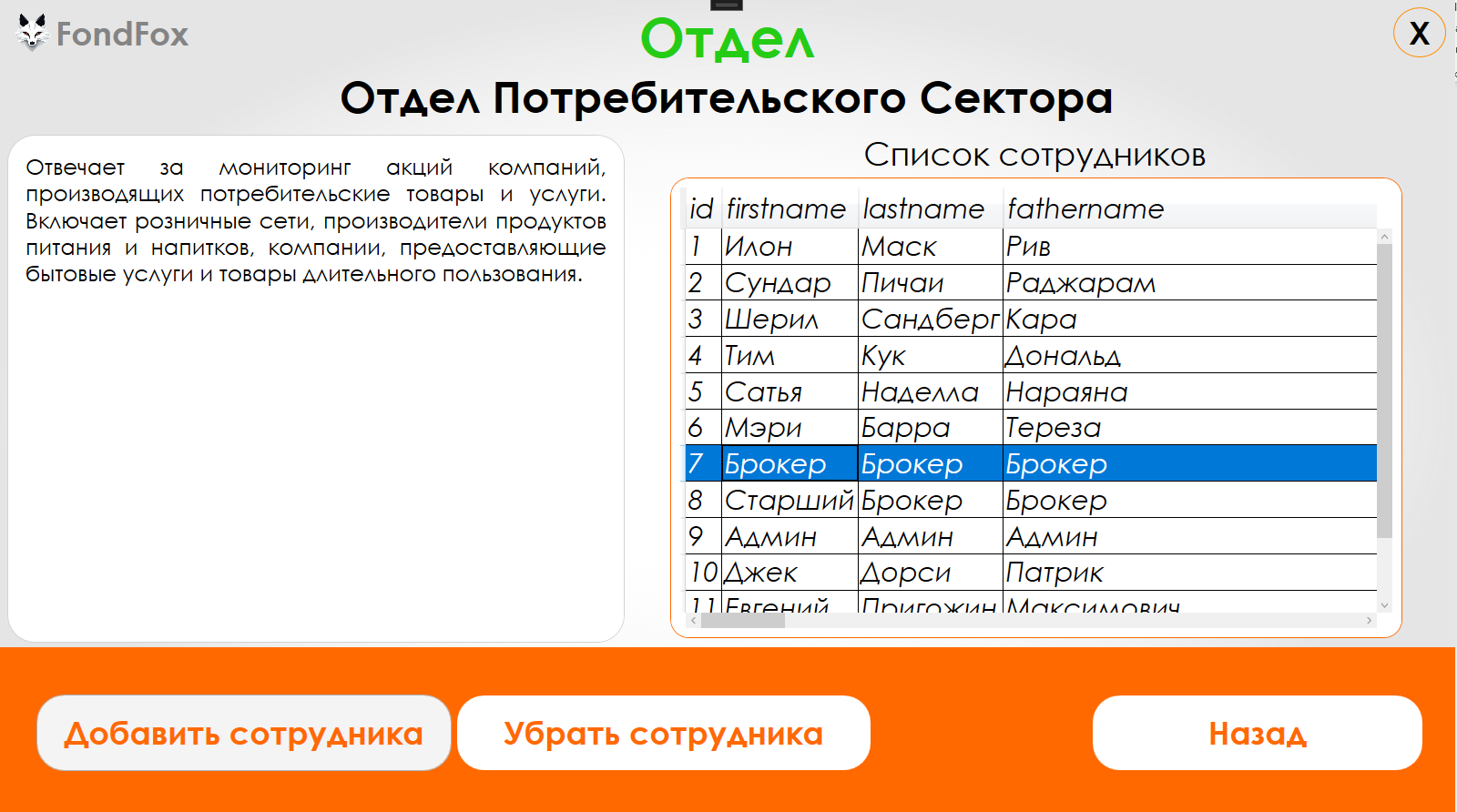


Рис.13 Окно добавления сотрудника в отдел

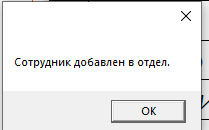


Рис.14 Оповещение о занятости сотрудника

**2.4. Реализация запросов и получение отчетов**

В окне торговой площадки есть страница с новостями, где каждый пользователь, который состоит в отделе может отправить сообщение (см. Рис 15). Сообщение отправляется от имени отдела, так что если у пользователя нет отдела, то он может просто просматривать сообщения, а при попытке отправки вылезет ошибка (см. Рис 16).

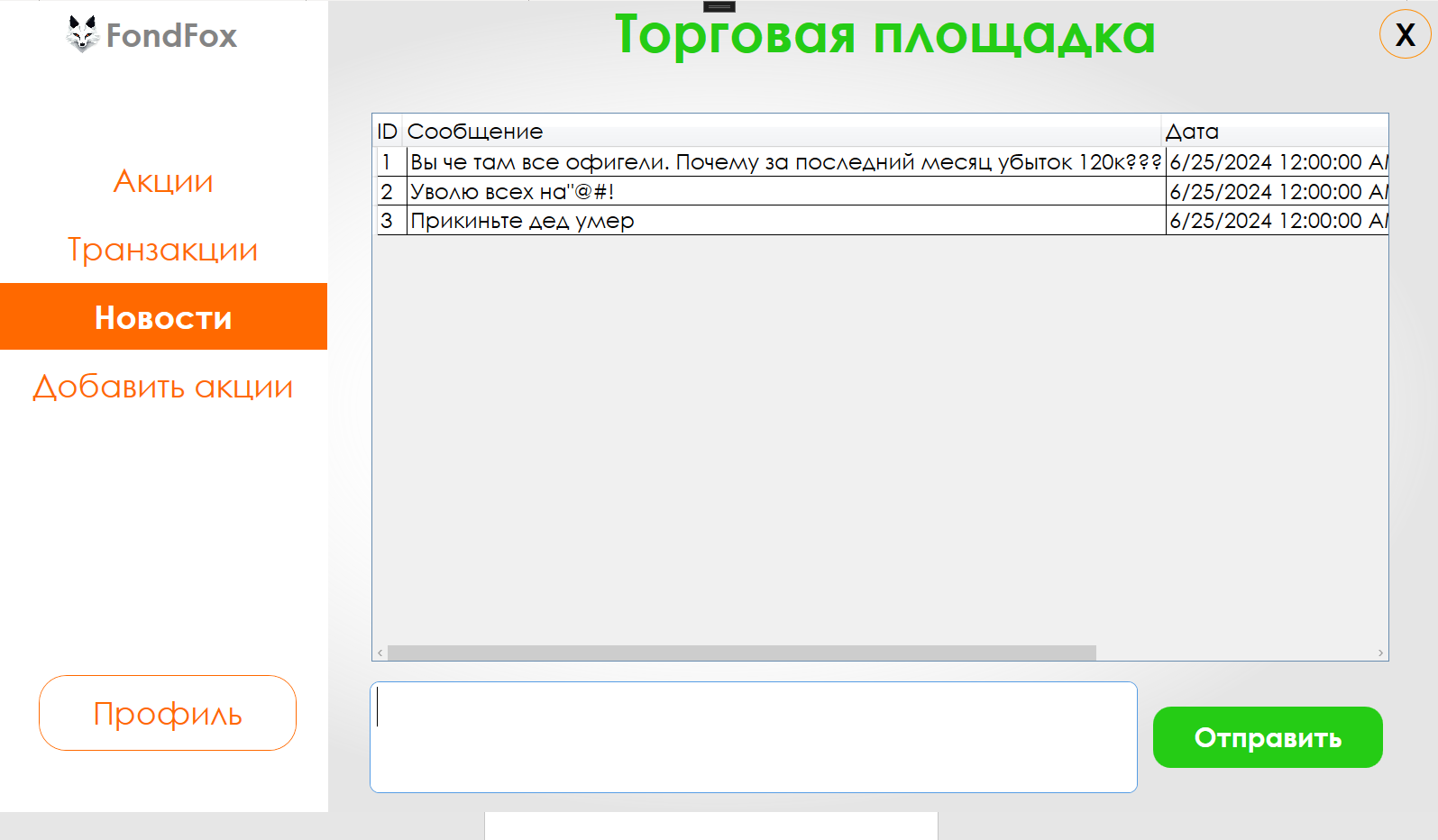


Рис. 15 Страница сообщений.

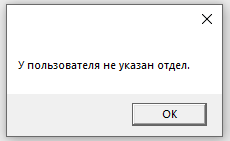


Рис. 16 Ошибка отправки, если сотрудник без отдела

**2.5. Администрирование базы данных**

В БД триггер организуется в таблице «Портфель» и отвечает за то, что во первых автоматически считает сумма умножая цену на количество, а во вторых складывает все сумму одно сотрудника и записывает их ему на баланс в таблицу «Пользователи»:

USE [FondFoxDB]

GO

/\*\*\*\*\*\* Object: Trigger [dbo].[trg\_UpdateBalance] Script Date: 25.06.2024 22:21:43 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Author,,Name>

-- Create date: <Create Date,,>

-- Description: <Description,,>

-- =============================================

ALTER TRIGGER [dbo].[trg\_UpdateBalance]

ON [dbo].[Portfel]

AFTER INSERT, UPDATE, DELETE

AS

BEGIN

SET NOCOUNT ON;

-- Обновить price в таблице Portfel на основе данных из Stocks

UPDATE p

SET p.price = s.price

FROM dbo.Portfel p

INNER JOIN dbo.Stocks s ON p.stock\_id = s.id;

-- Обновить баланс всех пользователей, чьи записи в портфеле изменились

UPDATE u

SET u.balance = ISNULL(p.TotalValue, 0)

FROM dbo.Users u

INNER JOIN (

SELECT p.user\_id, SUM(p.price \* p.cost) AS TotalValue

FROM dbo.Portfel p

GROUP BY p.user\_id

) p

ON u.id = p.user\_id;

-- Установить баланс в 0 для пользователей, у которых нет записей в портфеле

UPDATE u

SET u.balance = 0

FROM dbo.Users u

WHERE NOT EXISTS (

SELECT 1

FROM dbo.Portfel p

WHERE p.user\_id = u.id

);

END;

**2.6 Разработка интерфейса приложения**

При входе в приложение нас встречает окно авторизации, где необходимо ввести логин и пароль. Справа находится курс доллара и йен, который транслируется в реальном времени (см. Рис. 17).

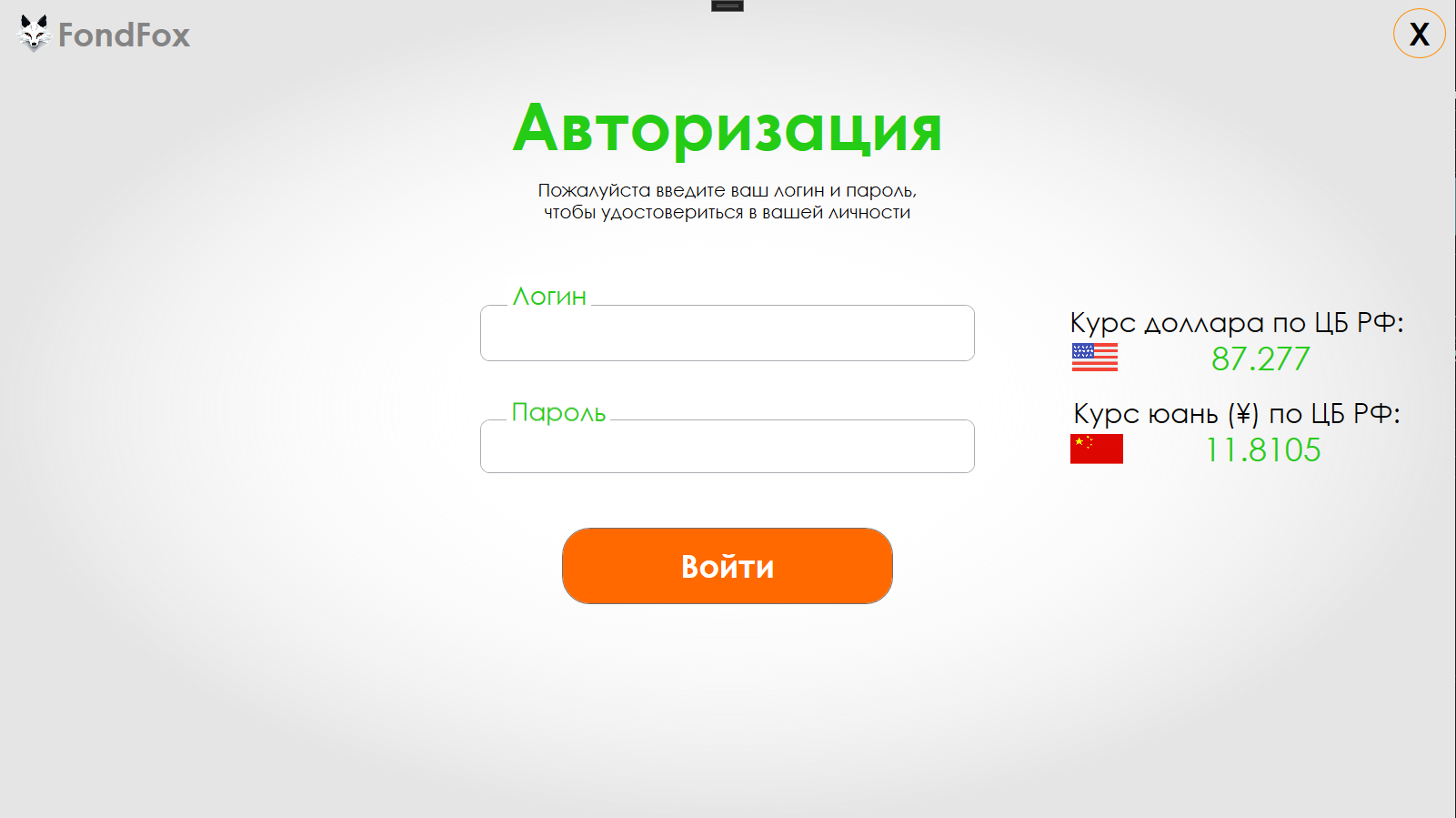


Рис. 17 Окно авторизации

При входе нас встречает главное окно, где написаны все основные данные сотрудника (см. Рис. 18).

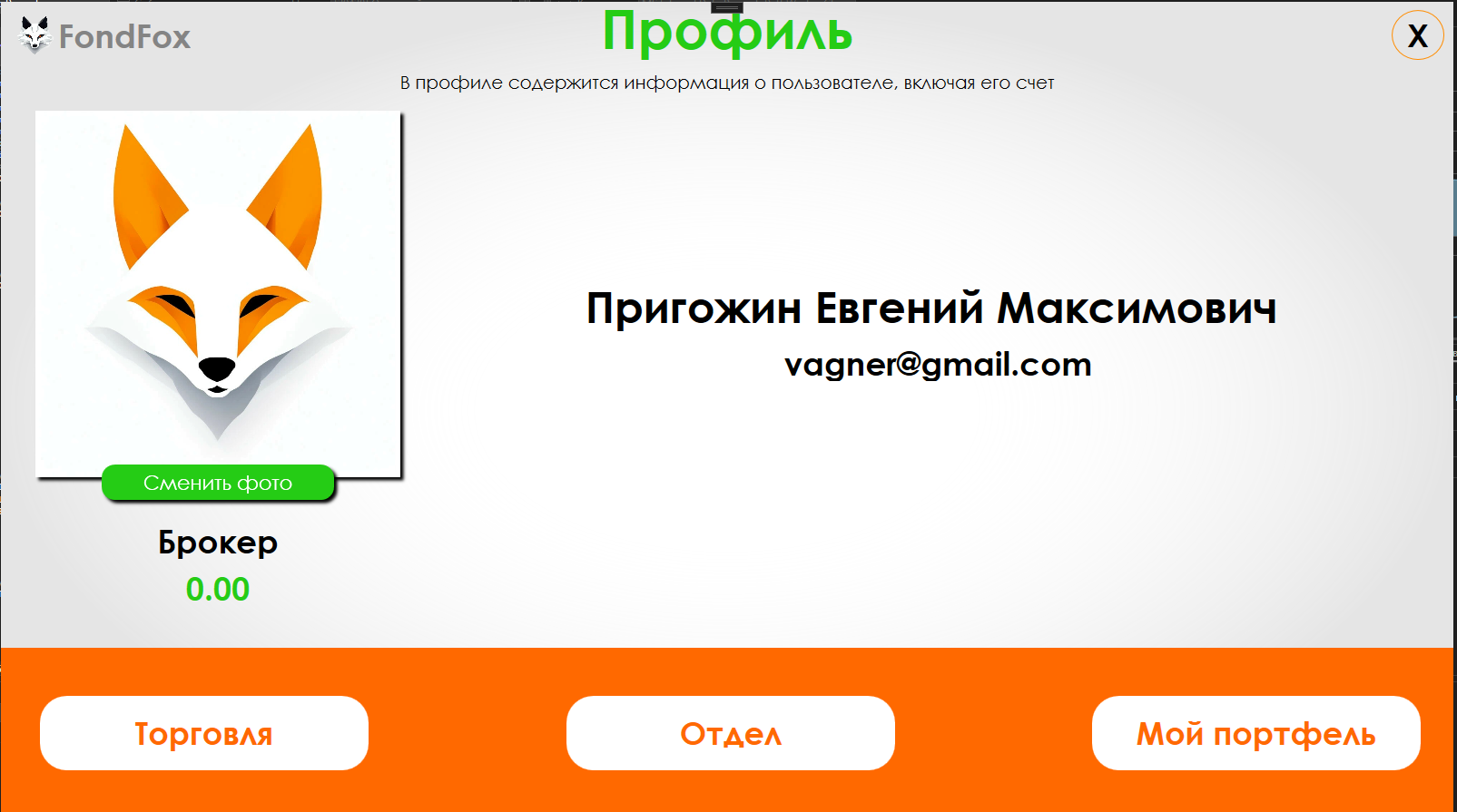


Рис. 18 Окно сотрудника

При нажатии кнопка «Торговля» открывается торговая площадка, где пользователю предоставлен выбор акицй, а также возможность поиска по названию и категории (См. Рис 19).

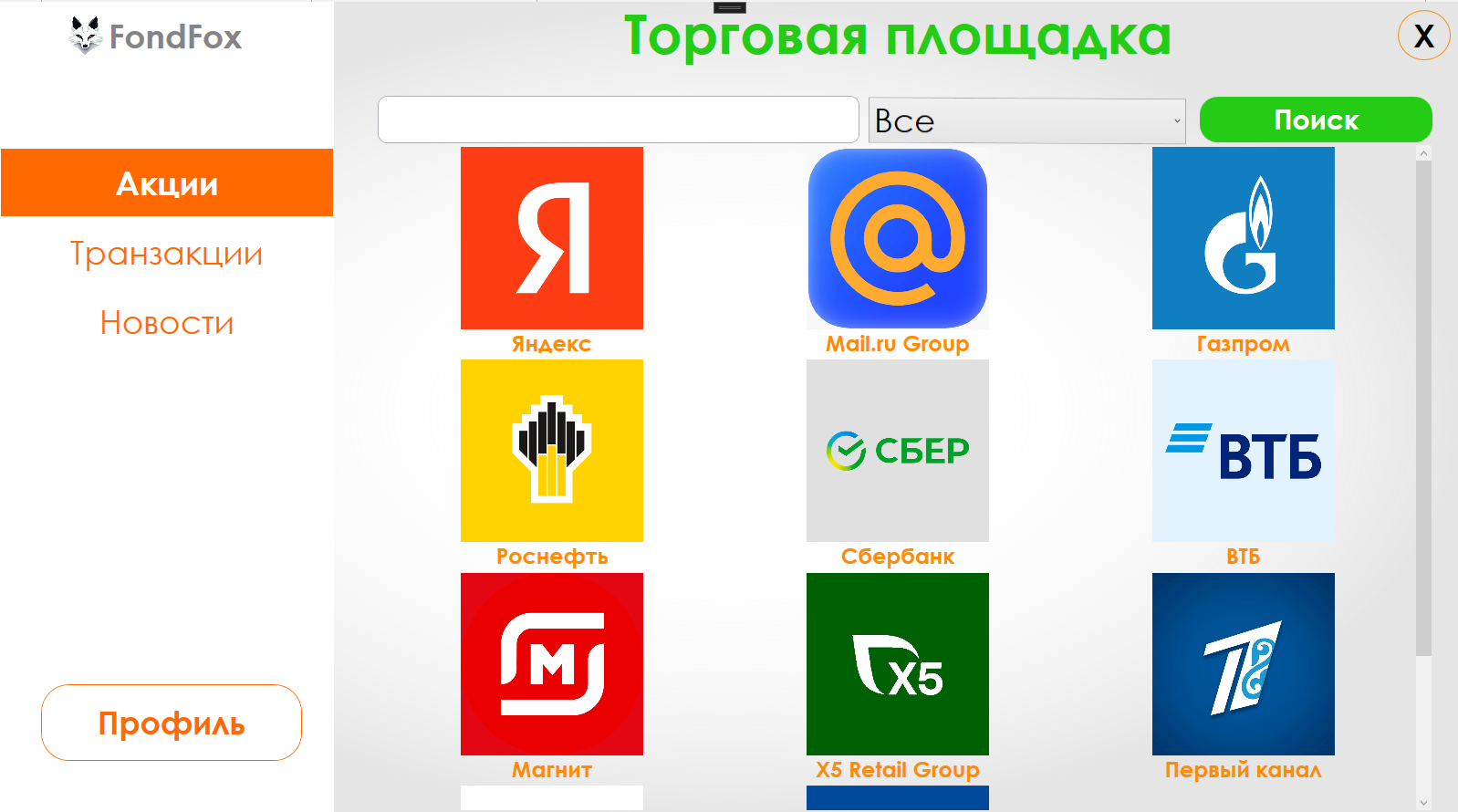


Рис. 19 Окно торговой площадки, вкладка акции

При нажатии кнопки «Транзакция» открывается список всех транзакций за всё время с подробной информацией (См. Рис 20).



Рис. 20 Окно торговой площадки, вкладка транзакций

При нажатии кнопки «Новости» открывается список сообщений сотрудников, а также есть возможность добавить своё (См. Рис 21).

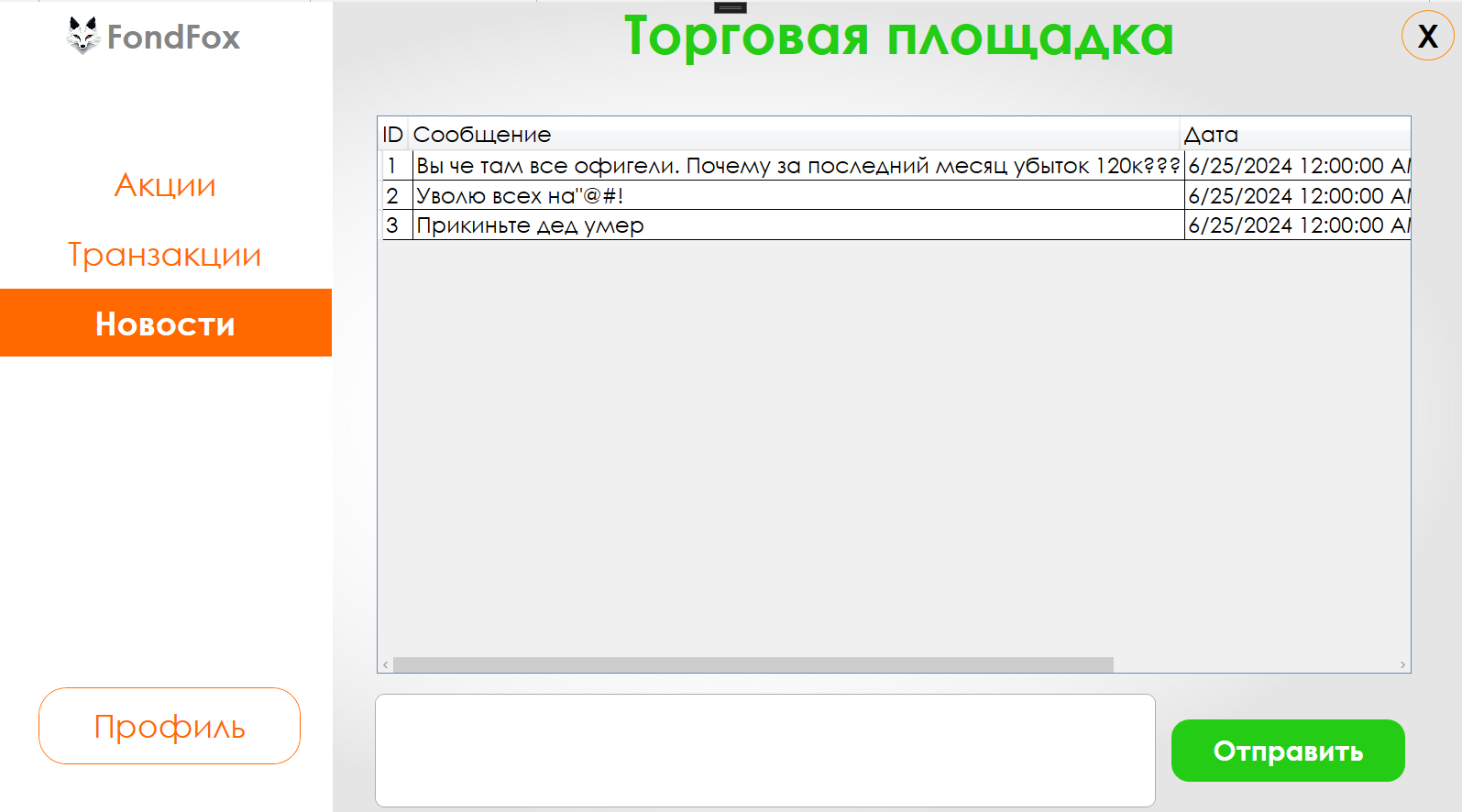


Рис. 21 Окно торговой площадки, вкладка новостей

При нажатии кнопки «Профиль» нас возвращает в профиль.

При нажатии на определенную акцию открывается её окно, где мы можем добавить определенное количество акций к себе указав их количество в тестовом поле и нажав на кнопку «Купить» (См. Рис 22).

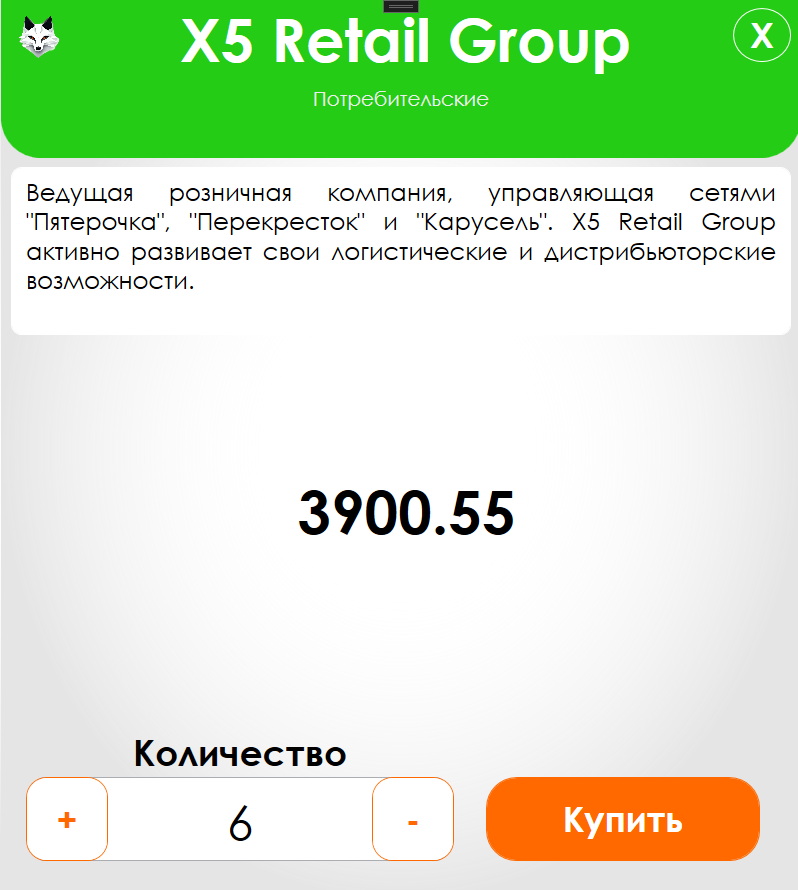


Рис. 22 Окно акции

При нажатии кнопки «X» окно закрывается. Работает на всех окнах.

В профиле мы можем перейти в свой портфель нажав на кнопку «Мой Портфель», у нас открывается окно просмотра всех своих акций (См. Рис 23). Здесь также можно открыть одну из акций и либо продать, либо докупить количество, которое можно вписать в текстовое поле (См. Рис 24).

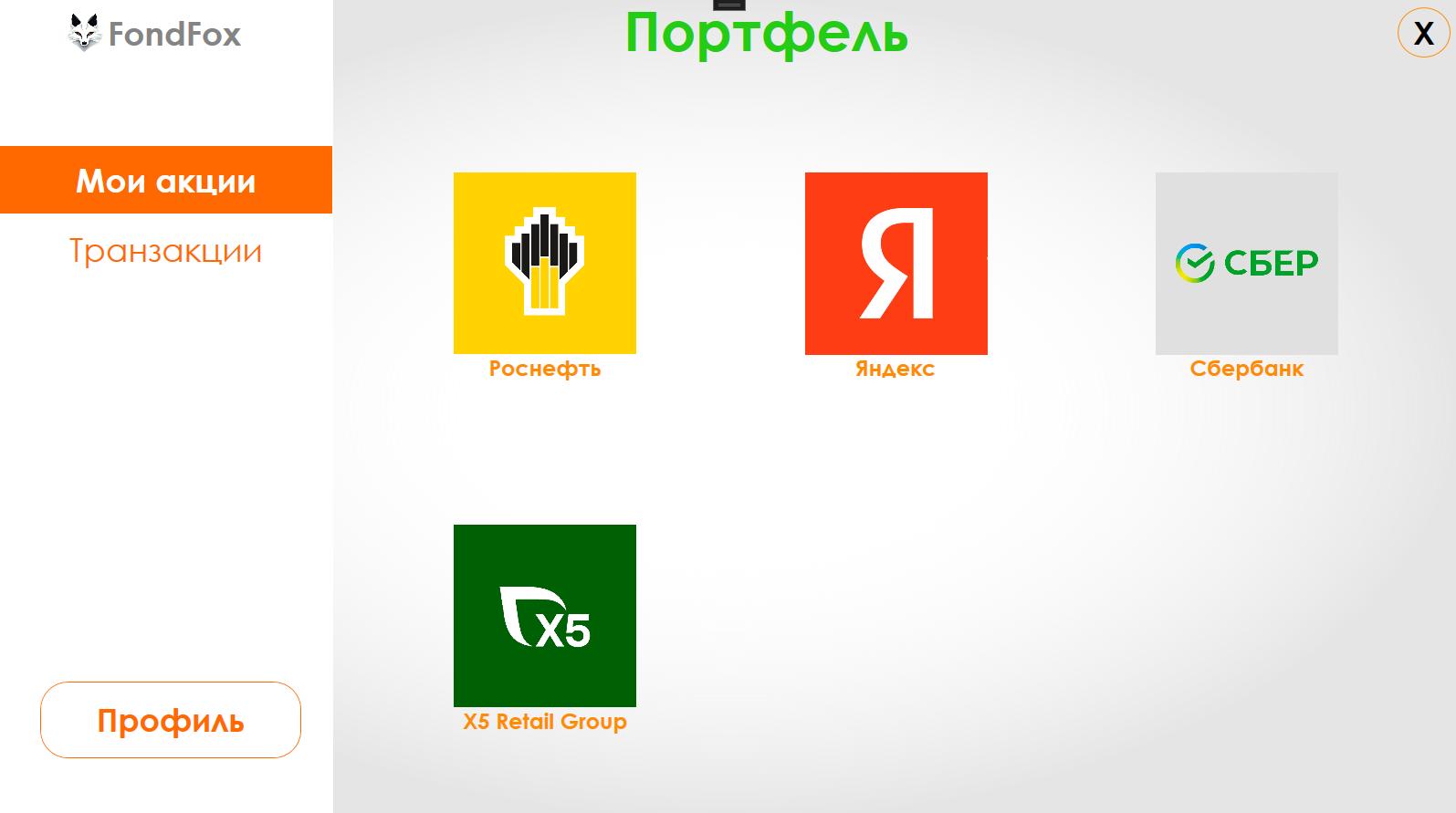


Рис. 23 Окно моего портфеля

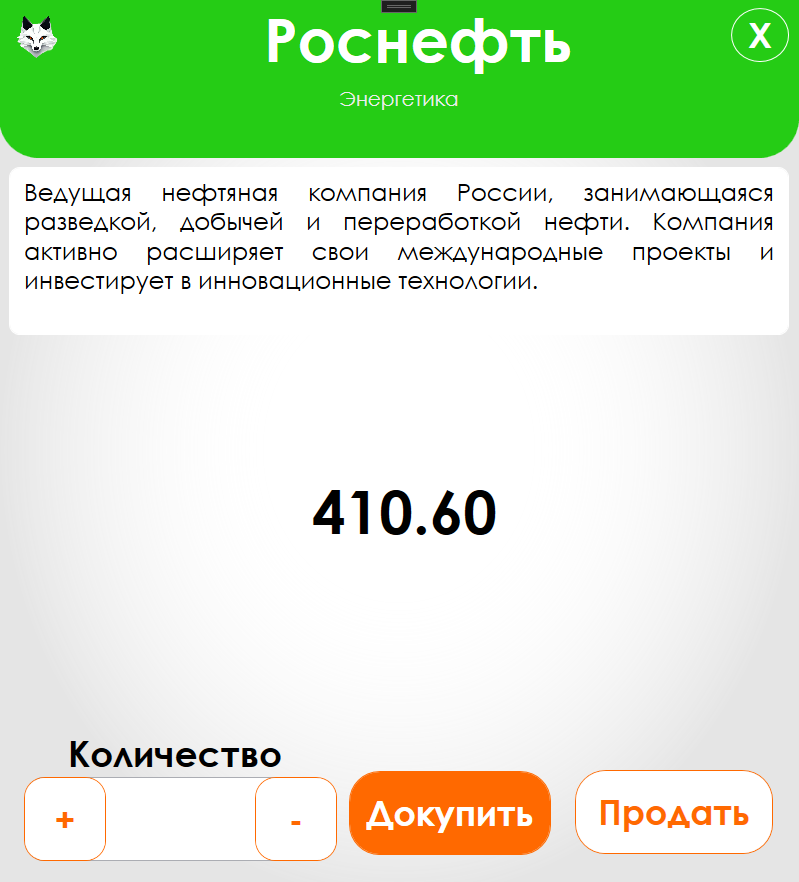


Рис. 24 Окно акции в моём портфеле

Если ввести количество больше, чем сейчас есть акций, то выйдет сообщение об ошибке (см. Рис 25).

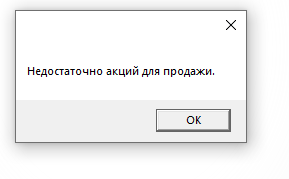


Рис. 25 Ошибка о превышении количества

В окне портфеля также можно переключиться на страницу «Транзакции» и посмотреть историю своих покупок и продаж, а также отфильтровать по «Покупкам» и «Продажам» (см. Рис 26).

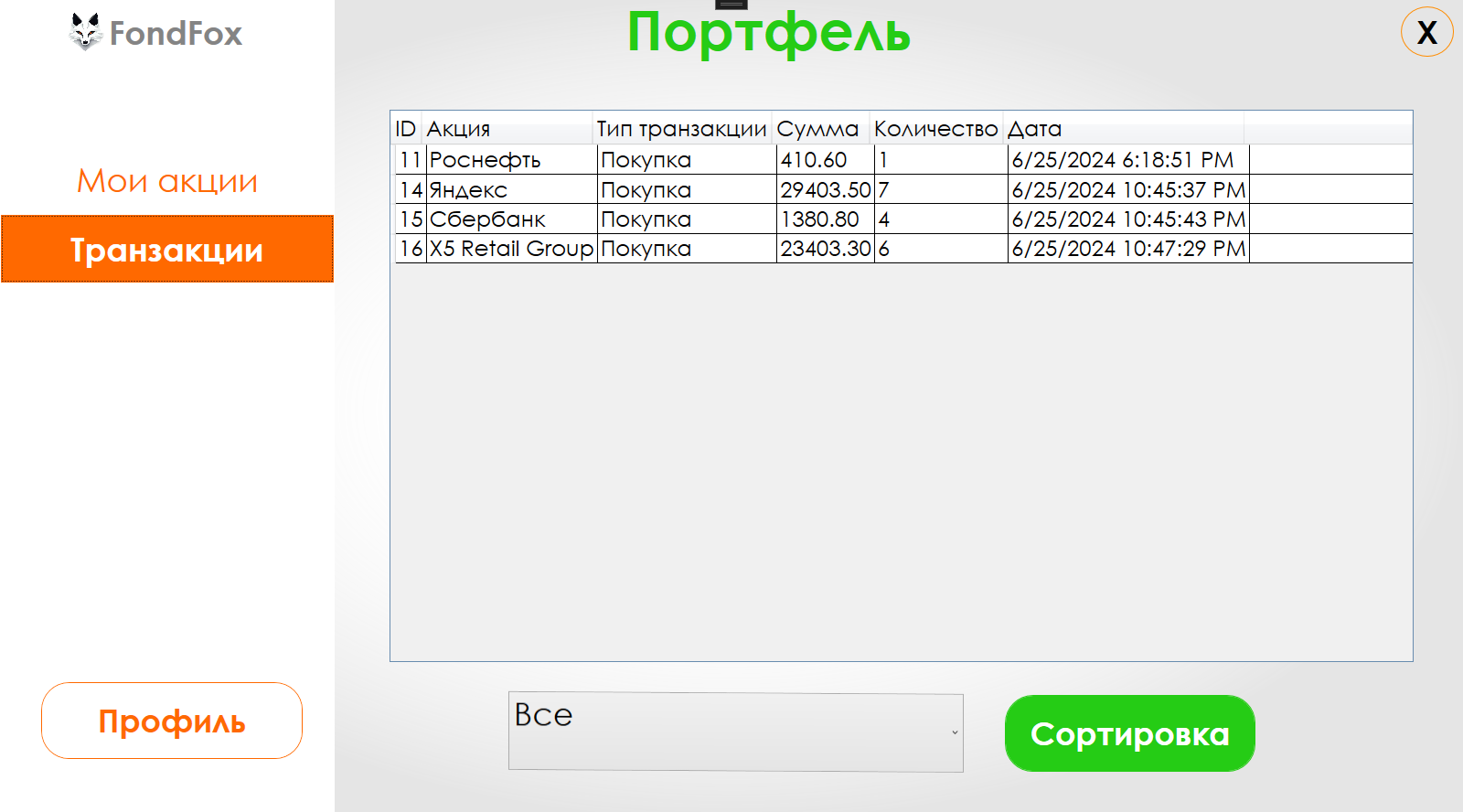


Рис. 26 Транзакции сотрудника

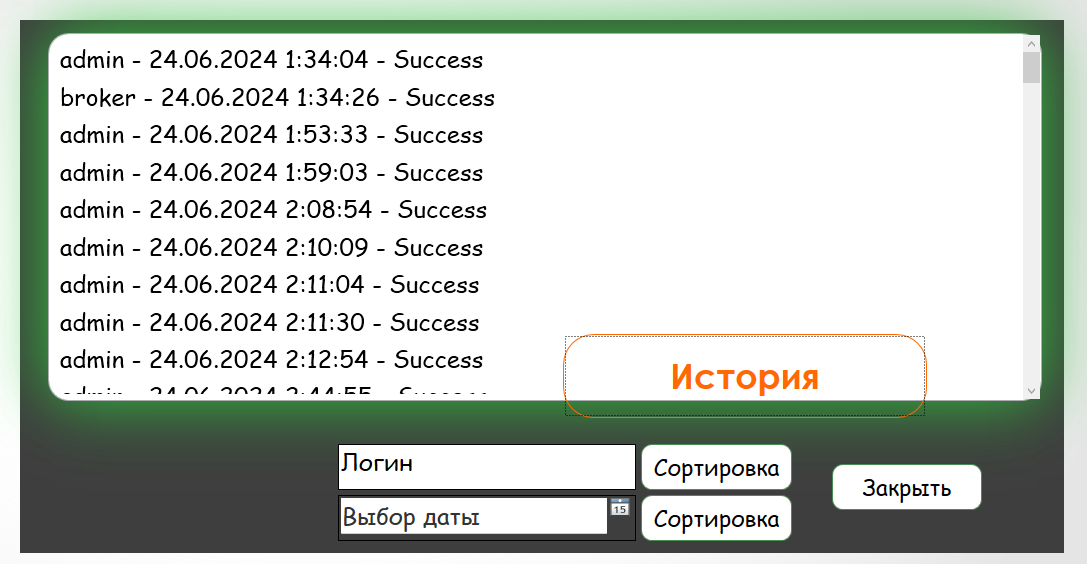
Также за администратора доступна возможность просмотра истории входа всех пользователей. При входе других ролей кнопки скрыты. Доступна в окне «Пользователь», где также возможна сортировка по дате и логину (см. Рис. 27).  


Рис. 27 История входа

**ЗАКЛЮЧЕНИЕ**

В процессе выполнения данного проекта был проведен анализ предметной области фондовой биржи «FondFox», что позволило разработать концептуальную и логическую модели базы данных. На основе этих моделей была создана физическая модель базы данных, использующая СУБД Microsoft SQL Server.

Для работы с базой данных было разработано, протестировано и отлажено приложение на платформе .NET Framework с использованием языка программирования C#. Полученная система обеспечивает разделение прав пользователей и предоставляет следующие функциональности:

* Просмотр информации об акциях и балансах;
* Добавление новых данных об акциях;
* Изменение информации об акциях;
* Удаление данных об акциях.

Приложение является удобным и простым в использовании. Тем не менее, перед началом работы рекомендуется ознакомиться с руководством оператора, разработанным в процессе выполнения проекта.

В итоге можно сделать вывод, что все поставленные задачи проекта были выполнены, а его цель - оптимизация управления акциями и балансами сотрудников на фондовой бирже «FondFox» - достигнута успешно.

# **СПИСКОК ЛИТЕРАТУРЫ**

1. ГОСТ 7.1–84. Библиографическое описание документа. Общие требования и правила составления.
2. ГОСТ 7.9-95. Реферат и аннотация. Общие Требования.
3. Диго С.М. Базы данных: учебное пособие. М.:МЭСИ, 2006. – 157с.
4. Киреева О.А. Базы данных: учебное пособие. Тольятти: Изд-во ТГУС, 2007. – 136с.
5. Кузьмина С.П. Базы данных: учебное пособие. СПб.: СПбГИЭУ, 2006. – 189с.
6. Кумскова И.А. Базы данных. – М.: Логус, 2012. – 462с.

**ПРИЛОЖЕНИЕ**

**Код окна Авторизации:**

using System;

using System.Linq;

using System.Windows;

namespace FondFoxProject

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

FondFoxDBEntities database = new FondFoxDBEntities();

protected RateViewModel viewModel;

public MainWindow()

{

InitializeComponent();

DataContext = viewModel = new RateViewModel();

}

private void entryBTN\_Click(object sender, RoutedEventArgs e)

{

try

{

var user = database.Users.FirstOrDefault(x => x.login == loginTXT.Text && x.password == passwordTXT.Text);

if (user != null)

{

// Добавляем запись в историю входов

var newLoginHistory = new LoginHistory

{

user\_id = user.id,

date = DateTime.Now,

result = true,

};

database.LoginHistory.Add(newLoginHistory);

database.SaveChanges();

// Открываем окно пользователя и передаем данные пользователя

UserWindow userWindow = new UserWindow(user);

userWindow.Show();

this.Close();

}

else

{

// Добавляем запись о неудачной попытке входа

var newLoginHistory = new LoginHistory

{

user\_id = database.Users.FirstOrDefault(u => u.login == loginTXT.Text)?.id ?? -1,

date = DateTime.Now,

result = false,

};

database.LoginHistory.Add(newLoginHistory);

database.SaveChanges();

MessageBox.Show("Пользователь не найден или неверный пароль!");

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка: " + ex.Message);

}

}

private async void OnLoad(object sender, RoutedEventArgs e)

{

await viewModel.OnLoad(sender, e);

}

private void closeBTN\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

}

}

**Код окна Пользователя:**

using Microsoft.Win32;

using System;

using System.Data.Entity;

using System.IO;

using System.Linq;

using System.Windows;

using System.Windows.Media.Imaging;

namespace FondFoxProject

{

public partial class UserWindow : Window

{

private Users \_user;

FondFoxDBEntities database = new FondFoxDBEntities();

public UserWindow(Users user)

{

InitializeComponent();

\_user = user;

LoadUserData();

LoadLoginHistory();

if (\_user.role == 3)

{

infoUsersBTN.Visibility = Visibility.Visible;

}

else

{

infoUsersBTN.Visibility = Visibility.Collapsed;

historyBTN.Visibility = Visibility.Collapsed;

}

}

private void LoadUserData()

{

var roleName = database.Role.FirstOrDefault(r => r.id == \_user.role)?.namerole;

// Fill the UI with user data

usernameLBL.Content = $"{\_user.lastname} {\_user.firstname} {\_user.fathername}";

emailLBL.Content = \_user.email;

roleLBL.Content = roleName;

summLBL.Content = \_user.balance;

// Load the user's photo if available

if (!string.IsNullOrEmpty(\_user.photo))

{

var photoPath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, \_user.photo);

if (File.Exists(photoPath))

{

photo.Source = new BitmapImage(new Uri(photoPath));

}

}

}

private void deportamentBTN\_Click(object sender, RoutedEventArgs e)

{

DepartmentWindow departmentWindow = new DepartmentWindow(\_user);

departmentWindow.Show();

this.Close();

}

private void tradingBTN\_Click(object sender, RoutedEventArgs e)

{

TradingWindow TradingWindow = new TradingWindow(\_user);

TradingWindow.Show();

this.Close();

}

private void brifBTN\_Click(object sender, RoutedEventArgs e)

{

BriefcaseWindow BriefcaseWindow = new BriefcaseWindow(\_user);

BriefcaseWindow.Show();

this.Close();

}

private void infoUsersBTN\_Click(object sender, RoutedEventArgs e)

{

InfousersWindow InfousersWindow = new InfousersWindow(\_user);

InfousersWindow.Show();

this.Close();

}

private void closeBTN\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

private void photoBTN\_Click(object sender, RoutedEventArgs e)

{

}

private void LoadLoginHistory(string sortBy = "", string loginFilter = "", DateTime? dateFilter = null)

{

var loginHistoryQuery = from lh in database.LoginHistory

join u in database.Users on lh.user\_id equals u.id

select new

{

Login = u.login,

Date = lh.date,

Result = lh.result ?? false // Установить значение по умолчанию для nullable boolean

};

// Apply login filter

if (!string.IsNullOrEmpty(loginFilter))

{

loginHistoryQuery = loginHistoryQuery.Where(lh => lh.Login.Contains(loginFilter));

}

// Apply date filter

if (dateFilter.HasValue)

{

loginHistoryQuery = loginHistoryQuery.Where(lh => DbFunctions.TruncateTime(lh.Date) == dateFilter.Value.Date);

}

// Apply sorting

switch (sortBy)

{

case "login":

loginHistoryQuery = loginHistoryQuery.OrderBy(lh => lh.Login);

break;

case "date":

loginHistoryQuery = loginHistoryQuery.OrderBy(lh => lh.Date);

break;

default:

loginHistoryQuery = loginHistoryQuery.OrderBy(lh => lh.Date); // Default sort by date

break;

}

// Load data into ListBox

loginHistoryListBox.Items.Clear();

foreach (var entry in loginHistoryQuery.ToList())

{

loginHistoryListBox.Items.Add($"{entry.Login} - {entry.Date} - {(entry.Result ? "Success" : "Failure")}");

}

}

private void loginSortBTN\_Click(object sender, RoutedEventArgs e)

{

string loginFilter = loginSortTXT.Text;

LoadLoginHistory("login", loginFilter);

}

private void dataSortBTN\_Click(object sender, RoutedEventArgs e)

{

DateTime? dateFilter = dataSortTXT.SelectedDate;

LoadLoginHistory("date", "", dateFilter);

}

private void closeHistory\_Click(object sender, RoutedEventArgs e)

{

historyGrid.Visibility = Visibility.Collapsed;

}

private void historyBTN\_Click(object sender, RoutedEventArgs e)

{

historyGrid.Visibility=Visibility.Visible;

}

}

}

**Код окна Торговой площадки:**

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace FondFoxProject

{

/// <summary>

/// Логика взаимодействия для TradingWindow.xaml

/// </summary>

public partial class TradingWindow : Window

{

FondFoxDBEntities database = new FondFoxDBEntities();

private string selectedImageFileName;

Users \_user;

public TradingWindow(Users user)

{

InitializeComponent();

LoadStockData();

LoadCategories();

LoadOrdersData();

LoadNewsData();

LoadStockTypes();

\_user = user;

if (\_user.role == 3)

lastnewsBTN.Visibility = Visibility.Visible;

else

lastnewsBTN.Visibility = Visibility.Collapsed;

}

private void LoadOrdersData()

{

var ordersData = (from o in database.Orders

join u in database.Users on o.user\_id equals u.id

join s in database.Stocks on o.stock\_id equals s.id

join t in database.TypeOrder on o.typeorder equals t.id

select new

{

OrderId = o.id,

UserLogin = u.login,

StockName = s.namestock,

TypeOrder = t.nametype,

Price = o.price,

Cost = o.cost,

Date = o.date

}).ToList();

ordersDataGrid.ItemsSource = ordersData;

ordersDataGrid.AutoGenerateColumns = false;

// Очистим колонки и добавим новые

ordersDataGrid.Columns.Clear();

ordersDataGrid.Columns.Add(new DataGridTextColumn { Header = "ID", Binding = new Binding("OrderId") });

ordersDataGrid.Columns.Add(new DataGridTextColumn { Header = "Логин", Binding = new Binding("UserLogin") });

ordersDataGrid.Columns.Add(new DataGridTextColumn { Header = "Акция", Binding = new Binding("StockName") });

ordersDataGrid.Columns.Add(new DataGridTextColumn { Header = "Тип транзакции", Binding = new Binding("TypeOrder") });

ordersDataGrid.Columns.Add(new DataGridTextColumn { Header = "Сумма", Binding = new Binding("Price") });

ordersDataGrid.Columns.Add(new DataGridTextColumn { Header = "Количество", Binding = new Binding("Cost") });

ordersDataGrid.Columns.Add(new DataGridTextColumn { Header = "Дата", Binding = new Binding("Date") });

}

private void LoadNewsData()

{

var newsData = (from n in database.News

join d in database.Department on n.department equals d.id

select new

{

Id = n.id,

Message = n.message,

Date = n.date,

Department = d.namedepartment

}).ToList();

newsDataGrid.ItemsSource = newsData;

}

private void AddNews()

{

if (string.IsNullOrEmpty(messageTXT.Text))

{

MessageBox.Show("Введите сообщение.");

return;

}

if (\_user.department.HasValue)

{

int userDepartmentId = \_user.department.Value;

News news = new News()

{

message = messageTXT.Text,

date = DateTime.Now,

department = userDepartmentId

};

database.News.Add(news);

database.SaveChanges();

MessageBox.Show("Новость успешно добавлена.");

LoadNewsData(); // Обновляем список новостей

}

else

{

MessageBox.Show("У пользователя не указан отдел.");

}

}

private void LoadStockData()

{

var stockData = database.Stocks.Select(s => new

{

Id = s.id,

NameStock = s.namestock,

Description = s.description,

TypeStock = s.typestock,

Price = s.price,

Photo = s.photo

}).ToList();

var stocks = stockData.Select(s => new Stocks

{

id = s.Id,

namestock = s.NameStock,

description = s.Description,

typestock = s.TypeStock,

price = s.Price,

photo = GetImagePath(s.Photo)

}).ToList();

stocksListView.ItemsSource = stocks;

}

private void LoadCategories()

{

var categories = database.TypeStock.ToList();

// Добавляем категорию "Все"

categories.Insert(0, new TypeStock { id = -1, name = "Все" });

categoryCMB.ItemsSource = categories;

categoryCMB.DisplayMemberPath = "name";

categoryCMB.SelectedValuePath = "id";

categoryCMB.SelectedValue = -1; // Устанавливаем категорию "Все" по умолчанию

}

private string GetImagePath(string photoFileName)

{

string baseDirectory = AppDomain.CurrentDomain.BaseDirectory;

string imagesFolder = "..\\..\\ImageStock\\";

string fullPath = System.IO.Path.Combine(baseDirectory, imagesFolder, photoFileName);

if (!System.IO.File.Exists(fullPath))

{

// Если файл не существует, можно задать путь к изображению по умолчанию

return System.IO.Path.Combine(baseDirectory, imagesFolder, "default.png");

}

return new Uri(fullPath).AbsoluteUri;

}

private void strockBTN\_Click(object sender, RoutedEventArgs e)

{

UpdateButtonStyles(sender as Button);

Page1.Visibility = Visibility.Visible;

Page2.Visibility = Visibility.Collapsed;

Page3.Visibility = Visibility.Collapsed;

Page4.Visibility = Visibility.Collapsed;

}

private void strocktimeBTN\_Click(object sender, RoutedEventArgs e)

{

UpdateButtonStyles(sender as Button);

Page2.Visibility = Visibility.Visible;

Page1.Visibility = Visibility.Collapsed;

Page3.Visibility = Visibility.Collapsed;

Page4.Visibility = Visibility.Collapsed;

}

private void orderBTN\_Click(object sender, RoutedEventArgs e)

{

UpdateButtonStyles(sender as Button);

Page3.Visibility = Visibility.Visible;

Page2.Visibility = Visibility.Collapsed;

Page1.Visibility = Visibility.Collapsed;

Page4.Visibility = Visibility.Collapsed;

}

private void lastnewsBTN\_Click(object sender, RoutedEventArgs e)

{

UpdateButtonStyles(sender as Button);

Page4.Visibility = Visibility.Visible;

Page2.Visibility = Visibility.Collapsed;

Page3.Visibility = Visibility.Collapsed;

Page1.Visibility = Visibility.Collapsed;

}

private void UpdateButtonStyles(Button activeButton)

{

var buttons = new Button[] { strockBTN, strocktimeBTN, orderBTN, lastnewsBTN, profileBTN };

foreach (var button in buttons)

{

if (button == activeButton)

{

button.Background = new SolidColorBrush((Color)ColorConverter.ConvertFromString("#FFFF6900"));

button.Foreground = Brushes.White;

button.FontWeight = FontWeights.Bold;

}

else

{

button.Background = Brushes.White;

button.Foreground = new SolidColorBrush((Color)ColorConverter.ConvertFromString("#FFFF6900"));

button.FontWeight = FontWeights.Normal;

}

}

}

private void profileBTN\_Click(object sender, RoutedEventArgs e)

{

UserWindow userWindow = new UserWindow(\_user);

userWindow.Show();

this.Close();

}

private void closeBTN\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

private void stocksListView\_MouseDoubleClick(object sender, MouseButtonEventArgs e)

{

var selectedStock = stocksListView.SelectedItem as Stocks;

if (selectedStock != null)

{

StockWindow stockWindow = new StockWindow(selectedStock, \_user);

stockWindow.ShowDialog();

}

}

private void searchBTN\_Click(object sender, RoutedEventArgs e)

{

string nameFilter = namestockTXT.Text.ToLower();

int? categoryFilter = categoryCMB.SelectedValue as int?;

var filteredStocks = database.Stocks.AsQueryable();

if (!string.IsNullOrEmpty(nameFilter))

{

filteredStocks = filteredStocks.Where(stock => stock.namestock.ToLower().Contains(nameFilter));

}

if (categoryFilter.HasValue)

{

filteredStocks = filteredStocks.Where(stock => stock.typestock == categoryFilter.Value);

}

var stockData = filteredStocks.Select(s => new

{

Id = s.id,

NameStock = s.namestock,

Description = s.description,

TypeStock = s.typestock,

Price = s.price,

Photo = s.photo

}).ToList();

var stocks = stockData.Select(s => new Stocks

{

id = s.Id,

namestock = s.NameStock,

description = s.Description,

typestock = s.TypeStock,

price = s.Price,

photo = GetImagePath(s.Photo)

}).ToList();

stocksListView.ItemsSource = stocks;

}

private void messegeBTN\_Click(object sender, RoutedEventArgs e)

{

AddNews();

}

private void imageBTN\_Click(object sender, RoutedEventArgs e)

{

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "Image files (\*.png;\*.jpeg;\*.jpg)|\*.png;\*.jpeg;\*.jpg";

if (openFileDialog.ShowDialog() == true)

{

string baseDirectory = AppDomain.CurrentDomain.BaseDirectory;

string imagesFolder = "..\\..\\ImageStock\\";

string destinationFolder = System.IO.Path.Combine(baseDirectory, imagesFolder);

if (!Directory.Exists(destinationFolder))

{

Directory.CreateDirectory(destinationFolder);

}

string fileName = System.IO.Path.GetFileName(openFileDialog.FileName);

string destinationPath = System.IO.Path.Combine(destinationFolder, fileName);

try

{

File.Copy(openFileDialog.FileName, destinationPath, true);

selectedImageFileName = fileName;

MessageBox.Show("Изображение успешно загружено.");

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при копировании файла: {ex.Message}");

}

}

}

private void LoadStockTypes()

{

var stockTypes = database.TypeStock.ToList();

newtypeTXT.ItemsSource = stockTypes;

newtypeTXT.DisplayMemberPath = "name";

newtypeTXT.SelectedValuePath = "id";

}

private void AddStockBTN\_Click(object sender, RoutedEventArgs e)

{

string name = NEWnamestockTXT.Text;

string description = NEWdescriptionTXT.Text;

string priceText = NEWpriceTXT.Text;

int? typeId = newtypeTXT.SelectedValue as int?;

if (string.IsNullOrEmpty(name) || string.IsNullOrEmpty(description) || string.IsNullOrEmpty(priceText) || !typeId.HasValue)

{

MessageBox.Show("Заполните все поля.");

return;

}

if (!decimal.TryParse(priceText, out decimal price))

{

MessageBox.Show("Введите корректную стоимость.");

return;

}

if (string.IsNullOrEmpty(selectedImageFileName))

{

MessageBox.Show("Пожалуйста, выберите изображение.");

return;

}

Stocks newStock = new Stocks()

{

namestock = name,

description = description,

price = price,

typestock = typeId.Value,

photo = selectedImageFileName // Сохраняем только имя файла изображения

};

try

{

database.Stocks.Add(newStock);

database.SaveChanges();

MessageBox.Show("Новая акция успешно добавлена.");

LoadStockData(); // Обновляем список акций

// Очищаем поля после добавления акции

NEWnamestockTXT.Text = string.Empty;

NEWdescriptionTXT.Text = string.Empty;

NEWpriceTXT.Text = string.Empty;

newtypeTXT.SelectedIndex = -1;

selectedImageFileName = null;

}

catch (System.Data.Entity.Validation.DbEntityValidationException dbEx)

{

StringBuilder errorMessages = new StringBuilder();

foreach (var validationErrors in dbEx.EntityValidationErrors)

{

foreach (var validationError in validationErrors.ValidationErrors)

{

errorMessages.AppendLine($"{validationError.PropertyName}: {validationError.ErrorMessage}");

}

}

MessageBox.Show(errorMessages.ToString(), "Ошибка валидации данных");

}

catch (Exception ex)

{

MessageBox.Show($"Произошла ошибка: {ex.Message}");

}

}

}

}

**Код окна Акции:**using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace FondFoxProject

{

/// <summary>

/// Логика взаимодействия для StockWindow.xaml

/// </summary>

public partial class StockWindow : Window

{

FondFoxDBEntities database = new FondFoxDBEntities();

private Stocks \_stock;

private Users \_user;

public StockWindow(Stocks stocks, Users users)

{

InitializeComponent();

\_stock = stocks;

\_user = users;

LoadStockDetails();

}

private void LoadStockDetails()

{

var catigoriname = database.TypeStock.FirstOrDefault(r => r.id == \_stock.typestock)?.name;

namestockLBL.Content = \_stock.namestock;

descriptionTXT.Text = \_stock.description;

priceLBL.Content = \_stock.price;

catigoryLBL.Text = catigoriname;

}

private void closeBTN\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

private void plusBTN\_Click(object sender, RoutedEventArgs e)

{

int currentValue = 0;

if (int.TryParse(costTXT.Text, out currentValue))

{

currentValue += 1;

costTXT.Text = currentValue.ToString();

}

else

{

costTXT.Text = "1";

}

}

private void minusBTN\_Click(object sender, RoutedEventArgs e)

{

int currentValue = 0;

if (int.TryParse(costTXT.Text, out currentValue))

{

currentValue -= 1;

costTXT.Text = currentValue.ToString();

}

else

{

costTXT.Text = "0";

}

}

private void costTXT\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

Regex regex = new Regex("[^0-9]+"); // Регулярное выражение для нецифровых символов

e.Handled = regex.IsMatch(e.Text);

}

private void buyBTN\_Click(object sender, RoutedEventArgs e)

{

try

{

// Проверка ввода количества

if (!int.TryParse(costTXT.Text, out int cost) || cost <= 0)

{

MessageBox.Show("Введите корректное количество.");

return;

}

// Проверка ввода цены

if (!decimal.TryParse(priceLBL.Content.ToString(), out decimal price))

{

MessageBox.Show("Ошибка при получении цены.");

return;

}

// Создание нового заказа

Orders order = new Orders

{

user\_id = \_user.id,

stock\_id = \_stock.id,

typeorder = 1,

price = price \* Convert.ToDecimal(costTXT.Text),

cost = cost,

date = DateTime.Now,

};

Portfel portfel = new Portfel

{

user\_id = \_user.id,

stock\_id = \_stock.id,

cost = cost,

};

// Добавление заказа в базу данных

database.Orders.Add(order);

database.Portfel.Add(portfel);

database.SaveChanges();

// Сообщение об успешном оформлении заказа

MessageBox.Show("Акции добавлены в портфель.");

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при добавлении акций: " + ex.Message);

}

}

}

}

**Код окна отделов:**using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

namespace FondFoxProject

{

public partial class DepartmentWindow : Window

{

private Users \_user;

private Department \_department;

FondFoxDBEntities database = new FondFoxDBEntities();

List<Users> users;

private Users selectedUser;

public DepartmentWindow(Users user)

{

InitializeComponent();

\_user = user;

users = database.Users.ToList();

userDataGrid.ItemsSource = users;

if (\_user.role == 2)

{

addUserBTN.IsEnabled = true;

deleteUserBTN.IsEnabled = true;

}

LoadUserData();

}

private void LoadUserData()

{

\_department = database.Department.FirstOrDefault(r => r.id == \_user.department);

var otdelName = \_department?.namedepartment;

var descriptionName = \_department?.description;

nameotdelLBL.Content = otdelName;

descriptionLBL.Text = descriptionName;

}

private void backBTN\_Click(object sender, RoutedEventArgs e)

{

UserWindow userWindow = new UserWindow(\_user);

userWindow.Show();

this.Close();

}

private void closeBTN\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

private void userDataGrid\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

selectedUser = userDataGrid.SelectedItem as Users;

if (selectedUser != null)

{

addUserBTN.IsEnabled = selectedUser.department == null;

deleteUserBTN.IsEnabled = selectedUser.department == \_department?.id;

}

}

private void addUserBTN\_Click(object sender, RoutedEventArgs e)

{

if (selectedUser != null && selectedUser.department == null)

{

selectedUser.department = \_department.id;

database.SaveChanges();

MessageBox.Show("Сотрудник добавлен в отдел.");

userDataGrid.ItemsSource = database.Users.ToList();

}

else

{

MessageBox.Show("Сотрудник уже находится в отделе или не выбран.");

}

}

private void deleteUserBTN\_Click(object sender, RoutedEventArgs e)

{

if (selectedUser != null && selectedUser.department == \_department.id)

{

selectedUser.department = null;

database.SaveChanges();

MessageBox.Show("Сотрудник удален из отдела.");

userDataGrid.ItemsSource = database.Users.ToList();

}

else

{

MessageBox.Show("Сотрудник не находится в данном отделе или не выбран.");

}

}

}

}

**Код окна добавления пользователя:**using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace FondFoxProject

{

/// <summary>

/// Логика взаимодействия для InfousersWindow.xaml

/// </summary>

public partial class InfousersWindow : Window

{

FondFoxDBEntities database = new FondFoxDBEntities();

Users \_user;

private int roleid = 0;

public InfousersWindow(Users users)

{

InitializeComponent();

LoadUsers();

\_user = users;

}

private void backBTN\_Click(object sender, RoutedEventArgs e)

{

UserWindow userWindow = new UserWindow(\_user);

userWindow.Show();

this.Close();

}

// Метод для загрузки пользователей и отображения в DataGrid

private void LoadUsers()

{

try

{

var users = database.Users.ToList(); // Получаем список всех пользователей из базы данных

datagridUSER.ItemsSource = users; // Устанавливаем этот список в качестве источника данных для DataGrid

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке пользователей: {ex.Message}");

}

}

private void closeBTN\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

private void addUserBTN\_Click(object sender, RoutedEventArgs e)

{

if (stbrokerCHECK.IsChecked == true)

roleid = 2;

else

roleid = 1;

// Проверка на наличие цифр в полях имени

if (firstnameTXT.Text.Any(char.IsDigit) ||

secondnameTXT.Text.Any(char.IsDigit) ||

fathernameTXT.Text.Any(char.IsDigit))

{

MessageBox.Show("Поля имени не могут содержать цифры.");

return;

}

// Проверка на уникальность логина

string login = loginTXT.Text.Trim();

var existingUser = database.Users.FirstOrDefault(u => u.login == login);

if (existingUser != null)

{

MessageBox.Show("Пользователь с таким логином уже существует. Пожалуйста, выберите другой логин.");

return;

}

// Создание нового пользователя и сохранение в базе данных

Users newUser = new Users()

{

firstname = firstnameTXT.Text,

lastname = secondnameTXT.Text,

fathername = fathernameTXT.Text,

email = emailTXT.Text,

login = login,

password = passwordTXT.Text,

role = roleid

};

database.Users.Add(newUser);

database.SaveChanges();

MessageBox.Show("Пользователь успешно добавлен.");

}

private void deleteUserBTN\_Click(object sender, RoutedEventArgs e)

{

// Получаем id пользователя для удаления

if (!int.TryParse(idTXT.Text, out int userId))

{

MessageBox.Show("Введите корректный ID пользователя для удаления.");

return;

}

// Находим пользователя по id

var userToDelete = database.Users.FirstOrDefault(u => u.id == userId);

if (userToDelete == null)

{

MessageBox.Show($"Пользователь с ID {userId} не найден.");

return;

}

// Запрашиваем подтверждение перед удалением

MessageBoxResult result = MessageBox.Show($"Вы уверены, что хотите удалить пользователя {userToDelete.firstname}?",

"Подтверждение удаления пользователя",

MessageBoxButton.YesNo,

MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

try

{

// Удаляем пользователя из базы данных

database.Users.Remove(userToDelete);

database.SaveChanges();

MessageBox.Show($"Пользователь {userToDelete.firstname} удалён.");

LoadUsers();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при удалении пользователя: {ex.Message}");

}

}

}

}

}

**Код окна Акции в портфеле:**using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace FondFoxProject

{

/// <summary>

/// Логика взаимодействия для StocksPortfilWindow.xaml

/// </summary>

public partial class StocksPortfilWindow : Window

{

FondFoxDBEntities database = new FondFoxDBEntities();

private Stocks \_stock;

private Users \_user;

public StocksPortfilWindow(Stocks stocks, Users users)

{

InitializeComponent();

\_stock = stocks;

\_user = users;

LoadStockDetails();

}

private void LoadStockDetails()

{

var catigoriname = database.TypeStock.FirstOrDefault(r => r.id == \_stock.typestock)?.name;

namestockLBL.Content = \_stock.namestock;

descriptionTXT.Text = \_stock.description;

priceLBL.Content = \_stock.price;

catigoryLBL.Text = catigoriname;

}

private void closeBTN\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

private void plusBTN\_Click(object sender, RoutedEventArgs e)

{

int currentValue = 0;

if (int.TryParse(costTXT.Text, out currentValue))

{

currentValue += 1;

costTXT.Text = currentValue.ToString();

}

else

{

costTXT.Text = "1";

}

}

private void minusBTN\_Click(object sender, RoutedEventArgs e)

{

int currentValue = 0;

if (int.TryParse(costTXT.Text, out currentValue))

{

currentValue -= 1;

costTXT.Text = currentValue.ToString();

}

else

{

costTXT.Text = "0";

}

}

private void costTXT\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

Regex regex = new Regex("[^0-9]+"); // Регулярное выражение для нецифровых символов

e.Handled = regex.IsMatch(e.Text);

}

private void buyBTN\_Click(object sender, RoutedEventArgs e)

{

try

{

// Проверка ввода количества

if (!int.TryParse(costTXT.Text, out int cost) || cost <= 0)

{

MessageBox.Show("Введите корректное количество.");

return;

}

// Проверка ввода цены

if (!decimal.TryParse(priceLBL.Content.ToString(), out decimal price))

{

MessageBox.Show("Ошибка при получении цены.");

return;

}

// Создание нового заказа

Orders order = new Orders

{

user\_id = \_user.id,

stock\_id = \_stock.id,

typeorder = 1,

price = price \* Convert.ToDecimal(costTXT.Text),

cost = cost,

date = DateTime.Now,

};

Portfel portfel = new Portfel

{

user\_id = \_user.id,

stock\_id = \_stock.id,

cost = cost,

};

// Добавление заказа в базу данных

database.Orders.Add(order);

database.Portfel.Add(portfel);

database.SaveChanges();

// Сообщение об успешном оформлении заказа

MessageBox.Show("Акции добавлены в портфель.");

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при добавлении акций: " + ex.Message);

}

}

private void outBTN\_Click(object sender, RoutedEventArgs e)

{

try

{

// Проверка ввода количества

if (!int.TryParse(costTXT.Text, out int cost) || cost <= 0)

{

MessageBox.Show("Введите корректное количество.");

return;

}

// Проверка наличия акции в портфеле пользователя

var userStock = database.Portfel.FirstOrDefault(p => p.user\_id == \_user.id && p.stock\_id == \_stock.id);

if (userStock == null || userStock.cost < cost)

{

MessageBox.Show("Недостаточно акций для продажи.");

return;

}

// Проверка ввода цены

if (!decimal.TryParse(priceLBL.Content.ToString(), out decimal price))

{

MessageBox.Show("Ошибка при получении цены.");

return;

}

// Создание нового заказа

Orders order = new Orders

{

user\_id = \_user.id,

stock\_id = \_stock.id,

typeorder = 2, // Тип операции "продажа"

price = price \* cost,

cost = cost,

date = DateTime.Now,

};

// Обновление количества акций в портфеле

userStock.cost -= cost;

if (userStock.cost == 0)

{

database.Portfel.Remove(userStock);

}

// Добавление заказа в базу данных

database.Orders.Add(order);

database.SaveChanges();

// Сообщение об успешной продаже

MessageBox.Show("Акции проданы.");

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при продаже акций: " + ex.Message);

}

}

}

}

**Код класса для отображения курса доллара онлайн:**using System;

using System.Linq;

using System.Net;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Threading;

using System.Xml.Linq;

namespace FondFoxProject

{

public class RateViewModel : PropertyChangedBase

{

protected Rate curRate;

protected Rate curRate2;

protected DispatcherTimer timer;

public RateViewModel()

{

// Создаем курс валюты

curRate = new Rate();

curRate2 = new Rate();

// Создаем таймер, которые каждые 10 секунд получает новый курс доллара

timer = new DispatcherTimer();

timer.Tick += new EventHandler(timer\_Tick);

timer.Interval = new TimeSpan(0, 0, 3);

timer.Start();

}

// Обработчик, который вызывается каждые 10 секуд для получения курса доллара

private async void timer\_Tick(object sender, EventArgs e)

{

try

{

CurRate.Value = await GetRate();

CurRate2.Value = await GetRate2();

}

catch

{

MessageBox.Show("Отсутствует подключение к интернету!");

}

}

// Событие случается при загрузке окна программы

public async Task OnLoad(object sender, RoutedEventArgs e)

{

CurRate.Value = await GetRate();

CurRate2.Value = await GetRate2();

}

// Запрос курса доллара с сайта

protected async Task<double> GetRate()

{

using (WebClient client = new WebClient())

{

var xml = await client.DownloadStringTaskAsync(new Uri("https://www.cbr-xml-daily.ru/daily.xml"));

XDocument xdoc = XDocument.Parse(xml);

var el = xdoc.Element("ValCurs").Elements("Valute");

string dollar = el.Where(x => x.Attribute("ID").Value == "R01235").Select(x => x.Element("Value").Value).FirstOrDefault();

if (!string.IsNullOrWhiteSpace(dollar))

return Convert.ToDouble(dollar);

}

return 0;

}

// Запрос курса юаня с сайта

protected async Task<double> GetRate2()

{

using (WebClient client = new WebClient())

{

var xml = await client.DownloadStringTaskAsync(new Uri("https://www.cbr-xml-daily.ru/daily.xml"));

XDocument xdoc = XDocument.Parse(xml);

var el = xdoc.Element("ValCurs").Elements("Valute");

string yan = el.Where(x => x.Attribute("ID").Value == "R01375").Select(x => x.Element("Value").Value).FirstOrDefault();

if (!string.IsNullOrWhiteSpace(yan))

return Convert.ToDouble(yan);

}

return 0;

}

// Свойство в котором хранится курс доллара

public Rate CurRate

{

get { return curRate; }

set

{

curRate = value;

OnPropertyChanged("CurRate");

}

}

// Свойство в котором хранится курс юань

public Rate CurRate2

{

get { return curRate2; }

set

{

curRate = value;

OnPropertyChanged("CurRate2");

}

}

}

}