Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Тамбовский государственный технический университет»

Многопрофильный колледж ТГТУ

|  |  |
| --- | --- |
| СОГЛАСОВАНО  Главный специалист предприятия    подпись инициалы, фамилия  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_г. | **УТВЕРЖДАЮ**  Заместитель директора по УМР  А.И. Попов  подпись  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г. |

**ОТЧЕТ**

по учебной практике

студента Шилина Дениса Сергеевича

Специальность 09.02.07Информационные системы и программирование

Семестр 6 Курс 3

Группа КСП225

Учебная практика по профессиональному модулю

ПМ.01 «Разработка модулей программного обеспечения для компьютерных систем», 108 часов

Руководитель практики В.В. Конкина

(Ф.И.О., подпись)

**ХАРАКТЕРИСТИКА**

на обучающегося по освоению общих и профессиональных компетенций

в период учебной практики

Студент Шашков Вячеслав Сергеевич

Специальность 09.02.07Информационные системы и программирование

Семестр 6 Курс 3

Группа КСП225

Учебная практика по профессиональному модулю

ПМ.01 «Разработка модулей программного обеспечения для компьютерных систем», 108 часов

За период практики с 02.09.2024 по 30.11.2024 освоены общие и профессиональные компетенции

|  |  |  |  |
| --- | --- | --- | --- |
| № компетенции | Название компетенции | Освоена /не освоена | Уровень освоения компетенций |
| Общие компетенции | | | |
| ОК 1 | Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам | освоен |  |
| ОК 2 | Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности | освоен |  |
| ОК 3 | Планировать и реализовывать собственное профессиональное и личностное развитие | освоен |  |
| ОК 4 | Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами | освоен |  |
| ОК 5 | Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста | освоен |  |
| ОК 6 | Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей, применять стандарты антикоррупционного поведения. | освоен |  |
| ОК 7 | Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях | освоен |  |
| ОК 8 | Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности | освоен |  |
| ОК 9 | Использовать информационные технологии в профессиональной деятельности | освоен |  |
| ОК 10 | Пользоваться профессиональной документацией на государственном и иностранном языке | освоен |  |
| Профессиональные компетенции | | | |
| ПК-1.1 | Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием | освоен |  |
| ПК-1.2 | Разрабатывать программные модули в соответствии с техническим заданием | освоен |  |
| ПК-1.3 | Выполнять отладку программных модулей с использованием специализированных программных средств | освоен |  |
| ПК-1.4 | Выполнять тестирование программных модулей | освоен |  |
| ПК-1.5 | Осуществлять рефакторинг и оптимизацию программного кода | освоен |  |
| ПК-1.6 | Разрабатывать модули программного обеспечения для мобильных платформ | освоен |  |

\*Уровни освоения компетенций: пороговый, базовый, продвинутый

Трудовая дисциплина

(примерная, удовлетворительная, неудовлетворительная)

Заместитель директора по УМР А.И. Попов

(Ф.И.О., подпись)

Руководитель практики В.В. Конкина

(Ф.И.О., подпись)

**ДНЕВНИК**

учебной практики

Студента Шилина Дениса Сергеевича

Специальность 09.02.07Информационные системы и программирование

Семестр 6 Курс 3

Группа КСП225

Период практики с 02.09.2024 по 30.11.2024

Учебная практика по профессиональному модулю

ПМ.01 «Разработка модулей программного обеспечения для компьютерных систем», 108 часов

Руководитель практики В.В. Конкина

(Ф.И.О., подпись)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Дата** | **Выполняемая работа (задание)** | **Оценка** | **Руководитель практики** | **Подпись** |
| **1** | **2** | **3** | **4** | **5** |
| 02.09.2021 | Ознакомление с поставленными задачами, содержанием выполняемых работ |  | В.В. Конкина |  |
| 06.09.2021 | Ознакомление с нормативными документами профильной организации |  | В.В. Конкина |  |
| 07.09.2021 | Прохождение инструктажа по охране труда, технике безопасности, пожарной безопасности, правилам внутреннего трудового распорядка |  | В.В. Конкина |  |
| 09.09.2021 | Ознакомление с организационно-методическими подходами к выполнению профессиональных задач |  | В.В. Конкина |  |
| 13.09.2021 | Уяснение индивидуального задания |  | В.В. Конкина |  |
| 14.09.2021 | Составление спецификации на программный модуль |  | В.В. Конкина |  |
| 16.09.2021-21.09.2021 | Анализ средств разработки |  | В.В. Конкина |  |
| 23.09.2021-05.10.2021 | Разработка алгоритма программного модуля в среде графического редактора |  | В.В. Конкина |  |
| 07.10.2021-08.11.2021 | Разработка кода программного модуля на языке программирования высокого уровня |  | В.В. Конкина |  |
| 09.11.2021-15.11.2021 | Тестирование и отладка программного модуля |  | В.В. Конкина |  |
| 16.11.2021-22.11.2021 | Разработка программной документации |  | В.В. Конкина |  |
| 23.11.2021 –29.11.2021 | Оформление отчета по практике |  | В.В. Конкина |  |
| 30.11.2021 | Промежуточная аттестация |  | В.В. Конкина |  |

**АТТЕСТАЦИОННЫЙ ЛИСТ**

об уровне освоения профессиональных компетенций

в период учебной практики

Студент Шашков Вячеслав Сергеевич

Специальность 09.02.07Информационные системы и программирование

Семестр 6 Курс 3

Группа КСП225

Учебная практика по профессиональному модулю

ПМ.01 «Разработка модулей программного обеспечения для компьютерных систем», 108 часов

За период учебной практики с 02.09.2024 по 30.11.2024

освоены профессиональные компетенции со следующим результатом:

|  |  |  |  |
| --- | --- | --- | --- |
| № компетенции | Название профессиональной компетенции | Освоена /не освоена | Уровень освоения компетенций |
| ПК-1.1 | Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием | освоен |  |
| ПК-1.2 | Разрабатывать программные модули в соответствии с техническим заданием | освоен |  |
| ПК-1.3 | Выполнять отладку программных модулей с использованием специализированных программных средств | освоен |  |
| ПК-1.4 | Выполнять тестирование программных модулей | освоен |  |
| ПК-1.5 | Осуществлять рефакторинг и оптимизацию программного кода | освоен |  |
| ПК-1.6 | Разрабатывать модули программного обеспечения для мобильных платформ | освоен |  |

\*Уровни освоения компетенций: пороговый, базовый, продвинутый

Заместитель директора по УМР А.И. Попов

(Ф.И.О., подпись)

Руководитель практики В.В. Конкина

(Ф.И.О., подпись)

**Аннотированный отчет**

Учебная практика проведена на базе Многопрофильного колледжа ФГБОУ ВО «ТГТУ» в период: с 02.09.2024 по 30.11.2024. За данный период было осуществлено:

* ознакомление с поставленными задачами, содержанием выполняемых работ, с нормативными документами Многопрофильного колледжа. с организационно-методическими подходами к выполнению профессиональных задач;
* прохождение инструктажа по охране труда, технике безопасности, пожарной безопасности, правилам внутреннего трудового распорядка;
* выполнение индивидуального задания – разработка программного модуля программного обеспечения для компьютерных систем, а именно:
* уяснение индивидуального задания;
* составление спецификации на программный модуль;
* анализ и выбор средств реализации программного модуля;
* разработка алгоритма программного модуля в виде блок-схемы в среде графического редактора;
* разработка кода программного модуля на языке программирования высокого уровня *Python*;
* тестирование и отладка программного модуля;
* разработка программной документации.

На заключительном этапе был оформлен отчет по учебной практике в соответствии с Положением об организации практики обучающихся, осваивающих образовательные программы среднего профессионального образования в Тамбовском государственном техническом университете (в редакции изменений, принятых решением Ученого совета ФГБОУ ВО «ТГТУ» от 05.10.2020 г. и ГОСТ СТО ФГБОУ ВО «ТГТУ» 07-2017.

**ПРИЛОЖЕНИЯ**

1 Индивидуальное задание

1. Выбрать предметную область и подробно описать её.
2. Подробно описать работу программы – составить спецификацию.
3. Проанализировать средства реализации (*Python* или *C*#) – привести достоинства и недостатки, сделать вывод о том, какой язык будет использован.
4. Построить блок-схему работы программы.
5. Написать код.
6. Провести тестирование.

2 Спецификация на разрабатываемый программный модуль

Требования к ПМ (пока без привязки к ЯП) и краткое описание его работы

3 Анализ средств реализации программного модуля

Во время разработки программного модуля для управления Фитнес-клубом использовалась среда разработки *Visual* *Studio* *Code*. Мой выбор пал на данную среду разработки из-за универсальности и поддержке различных расширений, которые значительно ускоряют разработку различного ПО.

Основным языком программирования стал *Python*, а так же встроенные в него библиотеки *Tkinter* и *SQLite*3 для реализации *GUI* и взаимодействия с БД.

Программный модуль реализован с использованием графического интерфейса на базе *Tkinter*. Это позволяет создать удобный пользовательский интерфейс с вкладками, в которых реализованы элементы управления для работы с данными о клиентах, абонементах и продажах. Для отображения информации о клиентах и абонементах используются виджеты *Treeview*, которые позволяют удобно представлять данные в виде таблиц. Для каждого типа информации (клиенты, абонементы, продажи) реализованы соответствующие вкладки с возможностью обновления и добавления новых данных.

Работа с базой данных осуществляется через библиотеку *SQLite*3. Во время инициализации приложения создается база данных *fitness*\_*club*.*db* и все необходимые таблицы для хранения информации о клиентах, абонементах и покупках. Для каждой из операций был реализованы отдельные методы, взаимодействующие с базой данных. Это позволяет обеспечивать надежную работы и безопасность данных, сохраняя целостность и актуальность.

Для дополнительных функциональных возможностей, таких как отображение местоположения фитнес-клубов на карте, была использована библиотека *tkintermapview*, что позволяет интегрировать карту и добавлять на нее маркеры и задавать для них координаты.

С точки зрения архитектуры, модуль построен с применением принципов объектно-ориентированного программирования. Был реализован класс *FitnessApp*, управляющий всеми функциями приложения, включая создание и настройку интерфейса, работу с базой данных, обработку событий и отображение статистики.

Так же была продумана составляющая интерфейса: используется словарь *COLORS* для удобной и быстрой настройки цветовой схемы. Цвета были подобраны так, чтобы улучшить восприятие, с применением контрастных цветов для выделения элементов взаимодействия и статусов. Это помогает пользователю ориентироваться в интерфейсе, улучшая взаимодействие с программой.

Таким образом, выбранный подход сочетает в себе простоту реализации и достаточную функциональность, обеспечивая эффективную работу с данными фитнес-клуба и удовлетворяя требования пользователей.

4 Разработка блок-схем алгоритма программного модуля

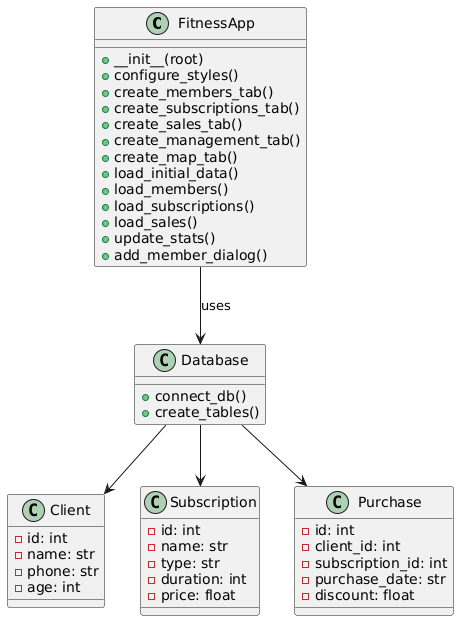


Рисунок 1 – Диаграмм классов

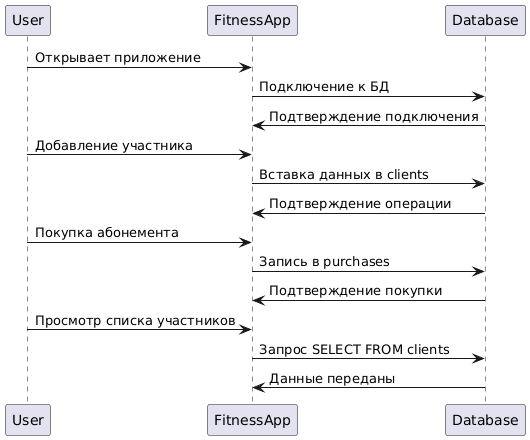


Рисунок 2 – Диаграмм последовательности

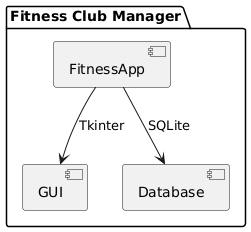


Рисунок 3 – Диаграмм компонентов

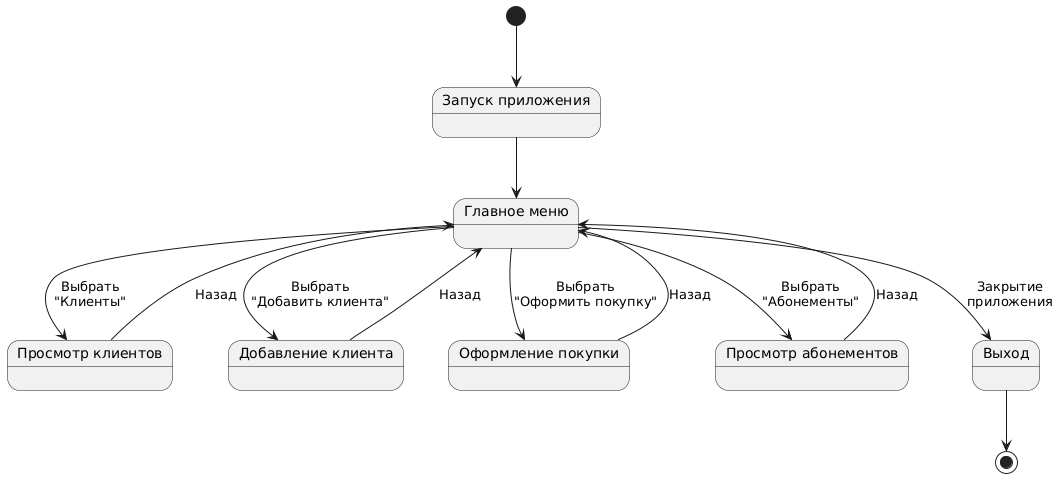


Рисунок 4 – Диаграмм состояний

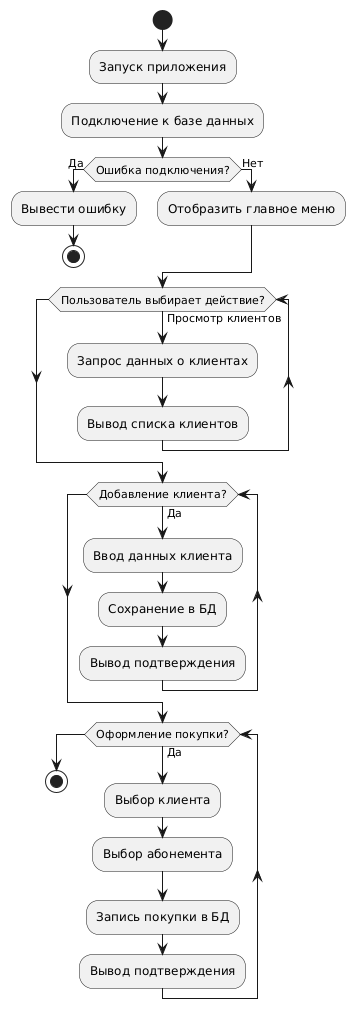


Рисунок 5 – Диаграмм активностей

5 Листинг программного кода

import tkinter as tk

from tkinter import ttk, messagebox

import sqlite3

from datetime import datetime, timedelta

from tkintermapview import TkinterMapView

COLORS = {

"primary": "#2A3D4C",

"secondary": "#A1B2B7",

"background": "#F1F5F9",

"text": "#3C4A55",

"success": "#6DBE45",

"warning": "#FFBB33",

"danger": "#E63946"

}

def connect\_db():

return sqlite3.connect('fitness\_club.db')

def create\_tables():

conn = connect\_db()

cursor = conn.cursor()

cursor.execute('''CREATE TABLE IF NOT EXISTS clients (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

phone TEXT NOT NULL,

age INTEGER NOT NULL)''')

cursor.execute('''CREATE TABLE IF NOT EXISTS subscriptions (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

type TEXT CHECK(type IN ('дневной', 'вечерний', 'безлимитный')),

duration INTEGER NOT NULL,

price REAL NOT NULL,

start\_time TEXT,

end\_time TEXT)''')

cursor.execute('''CREATE TABLE IF NOT EXISTS purchases (

id INTEGER PRIMARY KEY AUTOINCREMENT,

client\_id INTEGER,

subscription\_id INTEGER,

purchase\_date TEXT,

discount REAL DEFAULT 0,

FOREIGN KEY(client\_id) REFERENCES clients(id),

FOREIGN KEY(subscription\_id) REFERENCES subscriptions(id))''')

cursor.execute('''CREATE TABLE IF NOT EXISTS markers (

id INTEGER PRIMARY KEY AUTOINCREMENT,

latitude REAL NOT NULL,

longitude REAL NOT NULL,

text TEXT)''')

conn.commit()

conn.close()

class FitnessApp:

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

self.root = root

self.root.geometry("1280x720")

self.root.title("Fitness Club Manager")

self.root.configure(bg=COLORS["background"])

self.configure\_styles()

self.main\_container = ttk.Frame(root)

self.main\_container.pack(fill="both", expand=True)

self.notebook = ttk.Notebook(self.main\_container)

self.notebook.pack(expand=True, fill="both", padx=20, pady=20)

self.create\_members\_tab()

self.create\_subscriptions\_tab()

self.create\_sales\_tab()

self.create\_management\_tab()

self.create\_map\_tab()

create\_tables()

self.load\_initial\_data()

def configure\_styles(self):

style = ttk.Style()

style.theme\_use("clam")

style.configure(".", background=COLORS["background"], foreground=COLORS["text"])

style.configure("TNotebook", background=COLORS["background"], borderwidth=0)

style.configure("TNotebook.Tab",

font=('Arial', 10, 'bold'),

padding=15,

background=COLORS["secondary"],

foreground="white")

style.map("TNotebook.Tab",

background=[("selected", COLORS["primary"])],

foreground=[("selected", "white")])

style.configure("TButton",

font=('Arial', 10, 'bold'),

padding=8,

borderwidth=2,

relief="flat",

background=COLORS["primary"],

foreground="white")

style.map("TButton",

background=[('active', COLORS["secondary"]), ('disabled', '#D3D3D3')],

relief=[('active', 'sunken'), ('!active', 'flat')])

style.configure("Treeview",

font=('Arial', 10),

rowheight=30,

borderwidth=1,

relief="solid",

fieldbackground=COLORS["background"])

style.configure("Treeview.Heading",

font=('Arial', 11, 'bold'),

background=COLORS["primary"],

foreground="white",

relief="flat")

style.map("Treeview.Heading",

background=[('active', COLORS["secondary"])])

style.configure("Header.TLabel",

font=('Arial', 16, 'bold'),

foreground=COLORS["primary"],

background=COLORS["background"])

style.configure("Secondary.TLabel",

font=('Arial', 12),

foreground=COLORS["text"],

background=COLORS["background"])

style.configure("TEntry",

fieldbackground="white",

bordercolor=COLORS["primary"],

lightcolor=COLORS["primary"],

darkcolor=COLORS["primary"])

style.map("TEntry",

fieldbackground=[("readonly", "#F0F0F0")])

def create\_members\_tab(self):

tab = ttk.Frame(self.notebook)

self.notebook.add(tab, text="Участники")

header\_frame = ttk.Frame(tab)

header\_frame.pack(fill="x", padx=20, pady=10)

ttk.Label(header\_frame, text="Управление участниками", style="Header.TLabel").pack(side="left")

content\_frame = ttk.Frame(tab)

content\_frame.pack(fill="both", expand=True, padx=20, pady=10)

self.members\_tree = ttk.Treeview(content\_frame, columns=("ID", "Имя", "Телефон", "Возраст"), show="headings")

self.members\_tree.pack(fill="both", expand=True, pady=10)

for col in ("ID", "Имя", "Телефон", "Возраст"):

self.members\_tree.heading(col, text=col)

self.members\_tree.column(col, width=120, anchor="center")

btn\_frame = ttk.Frame(content\_frame)

btn\_frame.pack(pady=10)

ttk.Button(btn\_frame, text="Обновить список", command=self.load\_members).pack(side="left", padx=5)

ttk.Button(btn\_frame, text="Добавить участника", command=self.add\_member\_dialog, style="Accent.TButton").pack(side="left", padx=5)

def create\_subscriptions\_tab(self):

tab = ttk.Frame(self.notebook)

self.notebook.add(tab, text="Абонементы")

header\_frame = ttk.Frame(tab)

header\_frame.pack(fill="x", padx=20, pady=10)

ttk.Label(header\_frame, text="Доступные абонементы", style="Header.TLabel").pack(side="left")

content\_frame = ttk.Frame(tab)

content\_frame.pack(fill="both", expand=True, padx=20, pady=10)

self.subs\_tree = ttk.Treeview(content\_frame, columns=("ID", "Название", "Тип", "Длительность", "Цена"), show="headings")

self.subs\_tree.pack(fill="both", expand=True, pady=10)

for col in ("ID", "Название", "Тип", "Длительность", "Цена"):

self.subs\_tree.heading(col, text=col)

self.subs\_tree.column(col, width=120, anchor="center")

btn\_frame = ttk.Frame(content\_frame)

btn\_frame.pack(pady=10)

ttk.Button(btn\_frame, text="Обновить список", command=self.load\_subscriptions).pack(side="left", padx=5)

ttk.Button(btn\_frame, text="Добавить абонемент", command=self.add\_subscription\_dialog, style="Accent.TButton").pack(side="left", padx=5)

def create\_sales\_tab(self):

tab = ttk.Frame(self.notebook)

self.notebook.add(tab, text="Продажи")

header\_frame = ttk.Frame(tab)

header\_frame.pack(fill="x", padx=20, pady=10)

ttk.Label(header\_frame, text="Активные абонементы", style="Header.TLabel").pack(side="left")

content\_frame = ttk.Frame(tab)

content\_frame.pack(fill="both", expand=True, padx=20, pady=10)

self.sales\_tree = ttk.Treeview(content\_frame, columns=("ID", "Участник", "Абонемент", "Начало", "Окончание", "Осталось"), show="headings")

self.sales\_tree.pack(fill="both", expand=True, pady=10)

for col in ("ID", "Участник", "Абонемент", "Начало", "Окончание", "Осталось"):

self.sales\_tree.heading(col, text=col)

self.sales\_tree.column(col, width=120, anchor="center")

btn\_frame = ttk.Frame(content\_frame)

btn\_frame.pack(pady=10)

ttk.Button(btn\_frame, text="Обновить данные", command=self.load\_sales).pack(side="left", padx=5)

ttk.Button(btn\_frame, text="Оформить продажу", command=self.create\_sale\_dialog, style="Accent.TButton").pack(side="left", padx=5)

def create\_management\_tab(self):

tab = ttk.Frame(self.notebook)

self.notebook.add(tab, text="Администрирование")

header\_frame = ttk.Frame(tab)

header\_frame.pack(fill="x", padx=20, pady=10)

ttk.Label(header\_frame, text="Статистика клуба", style="Header.TLabel").pack(side="left")

stats\_frame = ttk.Frame(tab)

stats\_frame.pack(fill="both", expand=True, padx=20, pady=20)

stat\_card = ttk.Frame(stats\_frame, style="Card.TFrame")

stat\_card.pack(pady=10, fill="x", padx=20)

ttk.Label(stat\_card, text="Всего участников:", style="Secondary.TLabel").pack(side="left", padx=10, pady=5)

self.total\_members\_label = ttk.Label(stat\_card, text="0", style="Secondary.TLabel", font=('Arial', 14, 'bold'))

self.total\_members\_label.pack(side="left", padx=10)

stat\_card = ttk.Frame(stats\_frame, style="Card.TFrame")

stat\_card.pack(pady=10, fill="x", padx=20)

ttk.Label(stat\_card, text="Активных абонементов:", style="Secondary.TLabel").pack(side="left", padx=10, pady=5)

self.active\_subs\_label = ttk.Label(stat\_card, text="0", style="Secondary.TLabel", font=('Arial', 14, 'bold'))

self.active\_subs\_label.pack(side="left", padx=10)

ttk.Button(stats\_frame, text="Обновить статистику", command=self.update\_stats).pack(pady=20)

def create\_map\_tab(self):

tab = ttk.Frame(self.notebook)

self.notebook.add(tab, text="Локация")

control\_frame = ttk.Frame(tab)

control\_frame.pack(pady=10, padx=20, fill="x")

input\_frame = ttk.Frame(control\_frame)

input\_frame.pack(side="left", padx=10)

ttk.Label(input\_frame, text="Широта:").grid(row=0, column=0, padx=5)

self.lat\_entry = ttk.Entry(input\_frame, width=15)

self.lat\_entry.grid(row=0, column=1, padx=5)

ttk.Label(input\_frame, text="Долгота:").grid(row=0, column=2, padx=5)

self.lon\_entry = ttk.Entry(input\_frame, width=15)

self.lon\_entry.grid(row=0, column=3, padx=5)

ttk.Label(input\_frame, text="Название:").grid(row=0, column=4, padx=5)

self.marker\_text\_entry = ttk.Entry(input\_frame, width=20)

self.marker\_text\_entry.grid(row=0, column=5, padx=5)

ttk.Button(

control\_frame,

text="Добавить точку",

command=self.add\_marker,

style="Accent.TButton"

).pack(side="left", padx=10)

self.map\_widget = TkinterMapView(tab, width=1200, height=600, corner\_radius=15)

self.map\_widget.pack(pady=20, padx=20, fill="both", expand=True)

self.map\_widget.set\_position(55.7558, 37.6173)

self.map\_widget.set\_zoom(15)

marker = self.map\_widget.set\_marker(55.7558, 37.6173, text="Fitness Club")

marker.set\_text("Главный фитнес-клуб")

def load\_initial\_data(self):

self.load\_members()

self.load\_subscriptions()

self.load\_sales()

self.update\_stats()

self.load\_markers()

def load\_members(self):

for row in self.members\_tree.get\_children():

self.members\_tree.delete(row)

conn = connect\_db()

cursor = conn.cursor()

cursor.execute("SELECT id, name, phone, age FROM clients")

for row in cursor.fetchall():

self.members\_tree.insert("", "end", values=row)

conn.close()

def load\_subscriptions(self):

for row in self.subs\_tree.get\_children():

self.subs\_tree.delete(row)

conn = connect\_db()

cursor = conn.cursor()

cursor.execute("SELECT id, name, type, duration, price FROM subscriptions")

for row in cursor.fetchall():

self.subs\_tree.insert("", "end", values=row)

conn.close()

def load\_sales(self):

for row in self.sales\_tree.get\_children():

self.sales\_tree.delete(row)

conn = connect\_db()

cursor = conn.cursor()

cursor.execute('''SELECT p.id, c.name, s.name, p.purchase\_date,

DATE(p.purchase\_date, '+'||s.duration||' days'),

julianday(DATE(p.purchase\_date, '+'||s.duration||' days')) - julianday('now')

FROM purchases p

JOIN clients c ON p.client\_id = c.id

JOIN subscriptions s ON p.subscription\_id = s.id''')

for row in cursor.fetchall():

days\_left = int(float(row[5])) if row[5] else 0

self.sales\_tree.insert("", "end", values=(row[0], row[1], row[2], row[3], row[4], days\_left))

conn.close()

def update\_stats(self):

conn = connect\_db()

cursor = conn.cursor()

cursor.execute("SELECT COUNT(\*) FROM clients")

self.total\_members\_label.config(text=cursor.fetchone()[0])

cursor.execute("SELECT COUNT(\*) FROM purchases")

self.active\_subs\_label.config(text=cursor.fetchone()[0])

conn.close()

def add\_member\_dialog(self):

dialog = tk.Toplevel(self.root)

dialog.title("Добавить участника")

ttk.Label(dialog, text="ФИО:").grid(row=0, column=0, padx=10, pady=5)

name\_entry = ttk.Entry(dialog)

name\_entry.grid(row=0, column=1, padx=10, pady=5)

ttk.Label(dialog, text="Телефон:").grid(row=1, column=0, padx=10, pady=5)

phone\_entry = ttk.Entry(dialog)

phone\_entry.grid(row=1, column=1, padx=10, pady=5)

ttk.Label(dialog, text="Возраст:").grid(row=2, column=0, padx=10, pady=5)

age\_entry = ttk.Entry(dialog)

age\_entry.grid(row=2, column=1, padx=10, pady=5)

def save\_member():

if name\_entry.get() and phone\_entry.get() and age\_entry.get():

conn = connect\_db()

cursor = conn.cursor()

cursor.execute("INSERT INTO clients (name, phone, age) VALUES (?, ?, ?)",

(name\_entry.get(), phone\_entry.get(), age\_entry.get()))

conn.commit()

conn.close()

self.load\_members()

dialog.destroy()

else:

messagebox.showerror("Ошибка", "Заполните все поля")

ttk.Button(dialog, text="Сохранить", command=save\_member).grid(row=3, column=0, columnspan=2, pady=10)

def add\_subscription\_dialog(self):

dialog = tk.Toplevel(self.root)

dialog.title("Добавить абонемент")

ttk.Label(dialog, text="Название:").grid(row=0, column=0, padx=10, pady=5)

name\_entry = ttk.Entry(dialog)

name\_entry.grid(row=0, column=1, padx=10, pady=5)

ttk.Label(dialog, text="Тип:").grid(row=1, column=0, padx=10, pady=5)

type\_combobox = ttk.Combobox(dialog, values=["дневной", "вечерний", "безлимитный"])

type\_combobox.grid(row=1, column=1, padx=10, pady=5)

ttk.Label(dialog, text="Длительность (дней):").grid(row=2, column=0, padx=10, pady=5)

duration\_entry = ttk.Entry(dialog)

duration\_entry.grid(row=2, column=1, padx=10, pady=5)

ttk.Label(dialog, text="Цена:").grid(row=3, column=0, padx=10, pady=5)

price\_entry = ttk.Entry(dialog)

price\_entry.grid(row=3, column=1, padx=10, pady=5)

def save\_subscription():

if all([name\_entry.get(), type\_combobox.get(), duration\_entry.get(), price\_entry.get()]):

conn = connect\_db()

cursor = conn.cursor()

cursor.execute('''INSERT INTO subscriptions

(name, type, duration, price)

VALUES (?, ?, ?, ?)''',

(name\_entry.get(), type\_combobox.get(),

duration\_entry.get(), price\_entry.get()))

conn.commit()

conn.close()

self.load\_subscriptions()

dialog.destroy()

else:

messagebox.showerror("Ошибка", "Заполните все поля")

ttk.Button(dialog, text="Сохранить", command=save\_subscription).grid(row=4, column=0, columnspan=2, pady=10)

def create\_sale\_dialog(self):

dialog = tk.Toplevel(self.root)

dialog.title("Оформить продажу")

ttk.Label(dialog, text="Участник:").grid(row=0, column=0, padx=10, pady=5)

client\_combobox = ttk.Combobox(dialog)

client\_combobox.grid(row=0, column=1, padx=10, pady=5)

ttk.Label(dialog, text="Абонемент:").grid(row=1, column=0, padx=10, pady=5)

sub\_combobox = ttk.Combobox(dialog)

sub\_combobox.grid(row=1, column=1, padx=10, pady=5)

ttk.Label(dialog, text="Скидка (%):").grid(row=2, column=0, padx=10, pady=5)

discount\_entry = ttk.Entry(dialog)

discount\_entry.grid(row=2, column=1, padx=10, pady=5)

def load\_clients():

conn = connect\_db()

cursor = conn.cursor()

cursor.execute("SELECT name FROM clients")

client\_combobox['values'] = [row[0] for row in cursor.fetchall()]

conn.close()

def load\_subs():

conn = connect\_db()

cursor = conn.cursor()

cursor.execute("SELECT name FROM subscriptions")

sub\_combobox['values'] = [row[0] for row in cursor.fetchall()]

conn.close()

load\_clients()

load\_subs()

def process\_sale():

client = client\_combobox.get()

subscription = sub\_combobox.get()

discount = discount\_entry.get() or 0

if client and subscription:

conn = connect\_db()

cursor = conn.cursor()

cursor.execute("SELECT id FROM clients WHERE name = ?", (client,))

client\_id = cursor.fetchone()[0]

cursor.execute("SELECT id FROM subscriptions WHERE name = ?", (subscription,))

sub\_id = cursor.fetchone()[0]

cursor.execute('''INSERT INTO purchases

(client\_id, subscription\_id, purchase\_date, discount)

VALUES (?, ?, DATE('now'), ?)''',

(client\_id, sub\_id, discount))

conn.commit()

conn.close()

self.load\_sales()

self.update\_stats()

dialog.destroy()

else:

messagebox.showerror("Ошибка", "Выберите участника и абонемент")

ttk.Button(dialog, text="Оформить", command=process\_sale).grid(row=3, column=0, columnspan=2, pady=10)

def add\_marker(self):

lat = self.lat\_entry.get()

lon = self.lon\_entry.get()

text = self.marker\_text\_entry.get()

try:

lat\_float = float(lat)

lon\_float = float(lon)

self.map\_widget.set\_marker(lat\_float, lon\_float, text=text)

conn = connect\_db()

cursor = conn.cursor()

cursor.execute('''INSERT INTO markers (latitude, longitude, text)

VALUES (?, ?, ?)''', (lat\_float, lon\_float, text))

conn.commit()

conn.close()

self.lat\_entry.delete(0, "end")

self.lon\_entry.delete(0, "end")

self.marker\_text\_entry.delete(0, "end")

except ValueError:

messagebox.showerror("Ошибка", "Введите корректные координаты (числа)")

def load\_markers(self):

conn = connect\_db()

cursor = conn.cursor()

cursor.execute("SELECT latitude, longitude, text FROM markers")

for row in cursor.fetchall():

self.map\_widget.set\_marker(row[0], row[1], text=row[2])

conn.close()

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

root = tk.Tk()

app = FitnessApp(root)

root.mainloop()

6 Компиляция программного модуля

Компиляция программного модуля была выполнена с использованием библиотеки *auto*\_*py*\_*to*\_*exe*, которая предоставляет удобный интерфейс для преобразования *Python*-скриптов в исполняемые .*EXE* файлы. Процесс компиляции включал следующие этапы:

1. Описание процесса компиляции:

Для компиляции был использован Python-скрипт, содержащий логику работы приложения. С помощью библиотеки auto-py-to-exe был создан исполняемый файл, который позволяет запускать приложение без необходимости установки интерпретатора Python на целевом устройстве.

Для запуска библиотеки в терминале вводится команда: *python -m auto\_py\_to\_exe*.

1. Настройка компиляции:

В процессе настройки компиляции были заданы следующие параметры:

* *Onefile* (вариант *One* *File*) — сборка всех зависимостей и ресурсов в один исполняемый файл.
* *Console* *Window* (вариант *Window* *Based* (*hide* *the* *console*)) — создание приложения без отображения консоли.
* В разделе *Advanced* -> *General* *Options* -> --*name* было указано название конечного файла: "Панель управления фитнес-клубом".

1. Анализ результата компиляции:

В результате компиляции был получен корректный и полностью рабочий исполняемый файл. При запуске приложения в той же директории автоматически создается база данных (БД), необходимая для работы с данными. Исполняемый файл функционирует в соответствии с ожиданиями, ошибок или сбоев в процессе работы не было выявлено.

7 Тестирование программного модуля

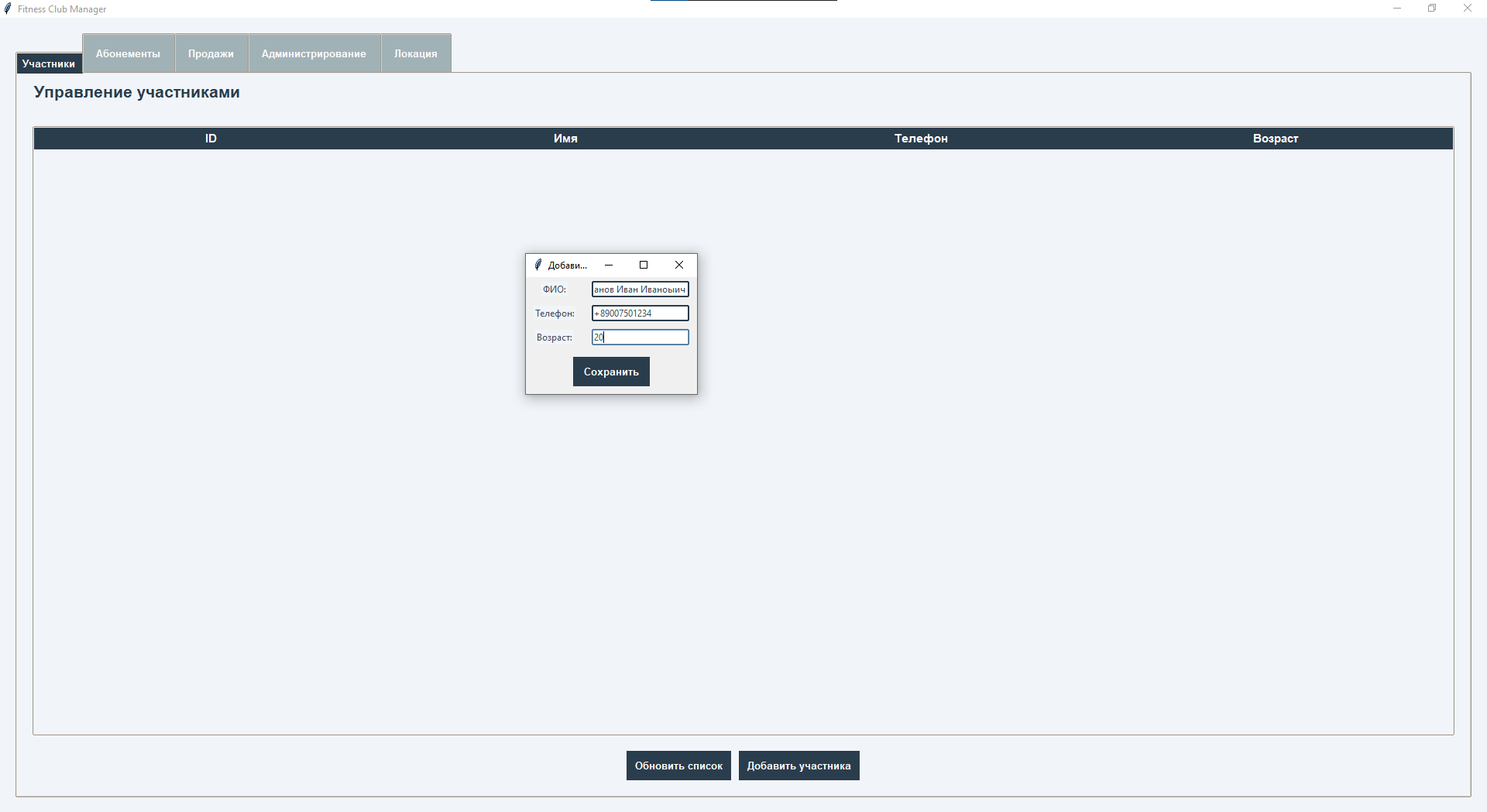


Рисунок 6 – Добавление участника

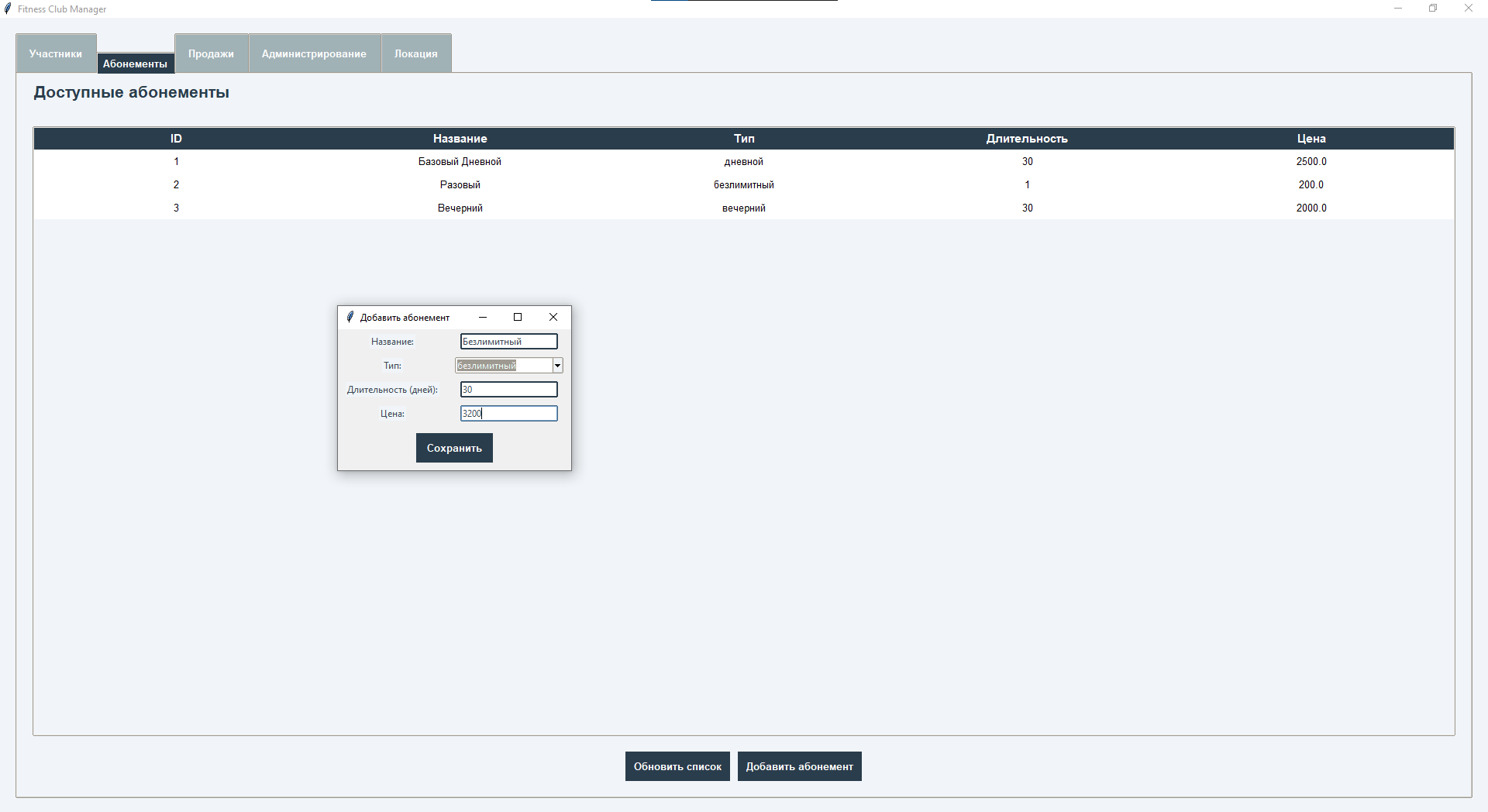


Рисунок 7 – Добавление абонемента

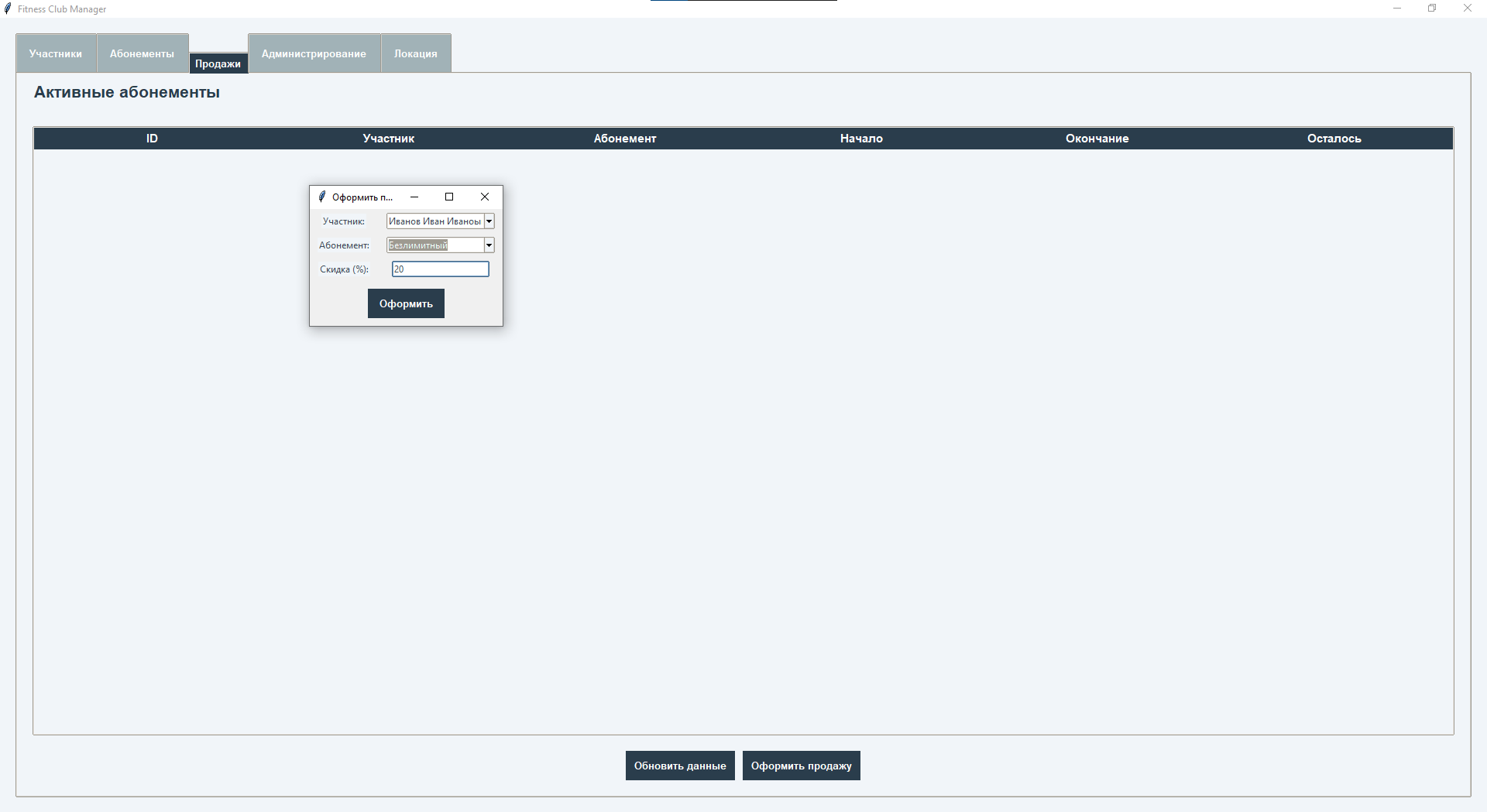


Рисунок 8 – Добавление продажи

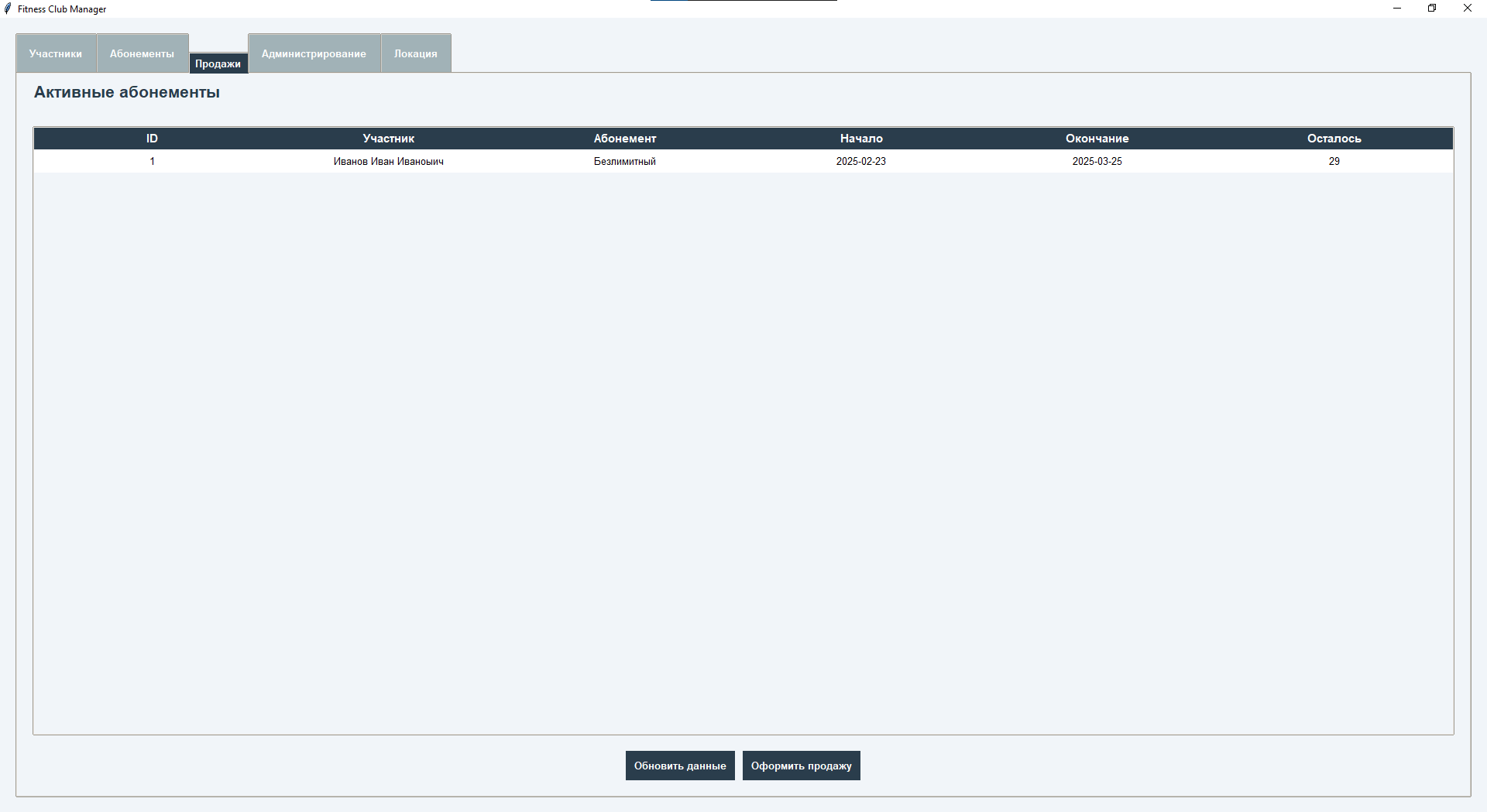


Рисунок 9 – Данные о проданных абонементах

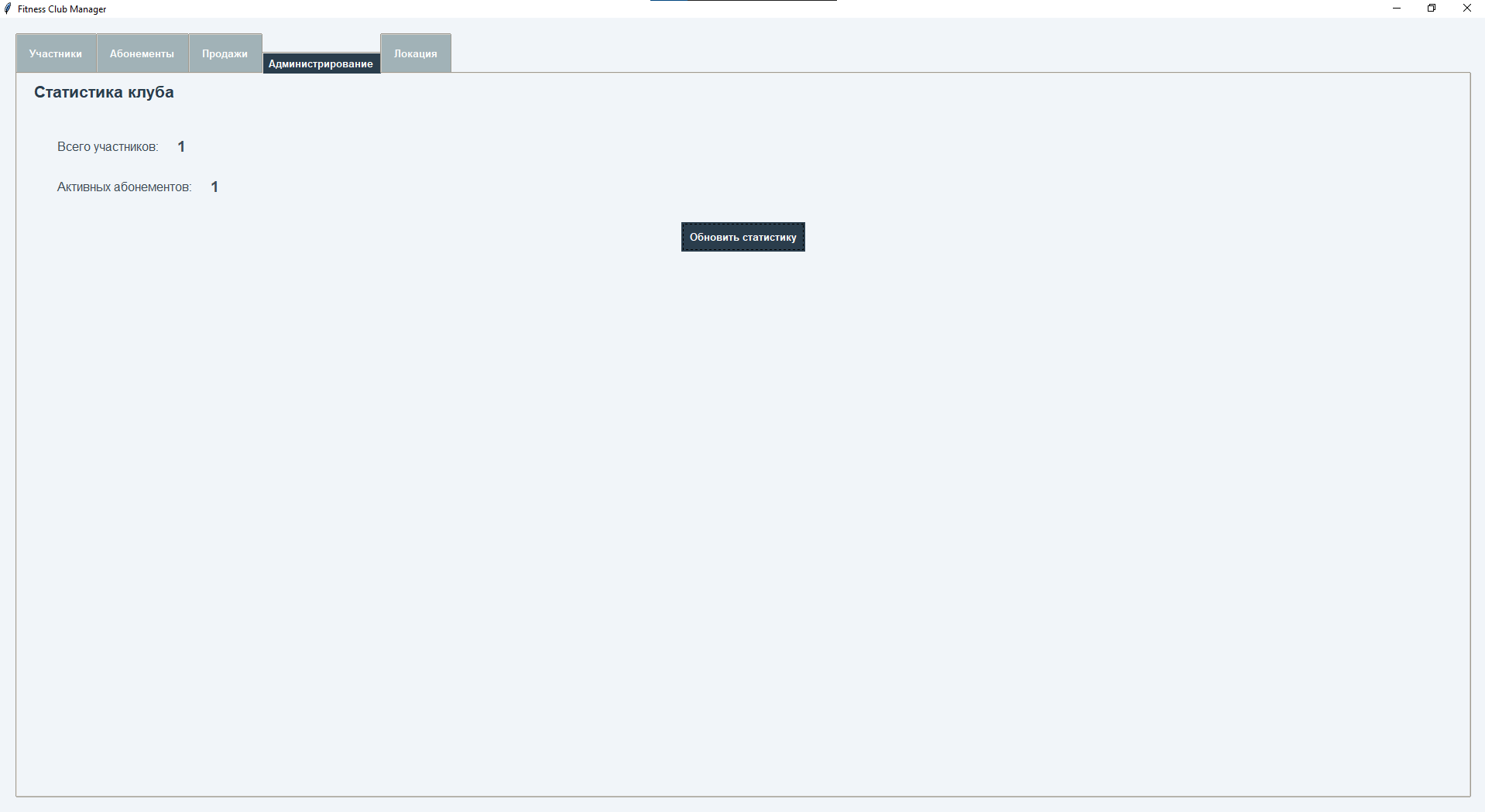


Рисунок 10 – Статистика для администрирования

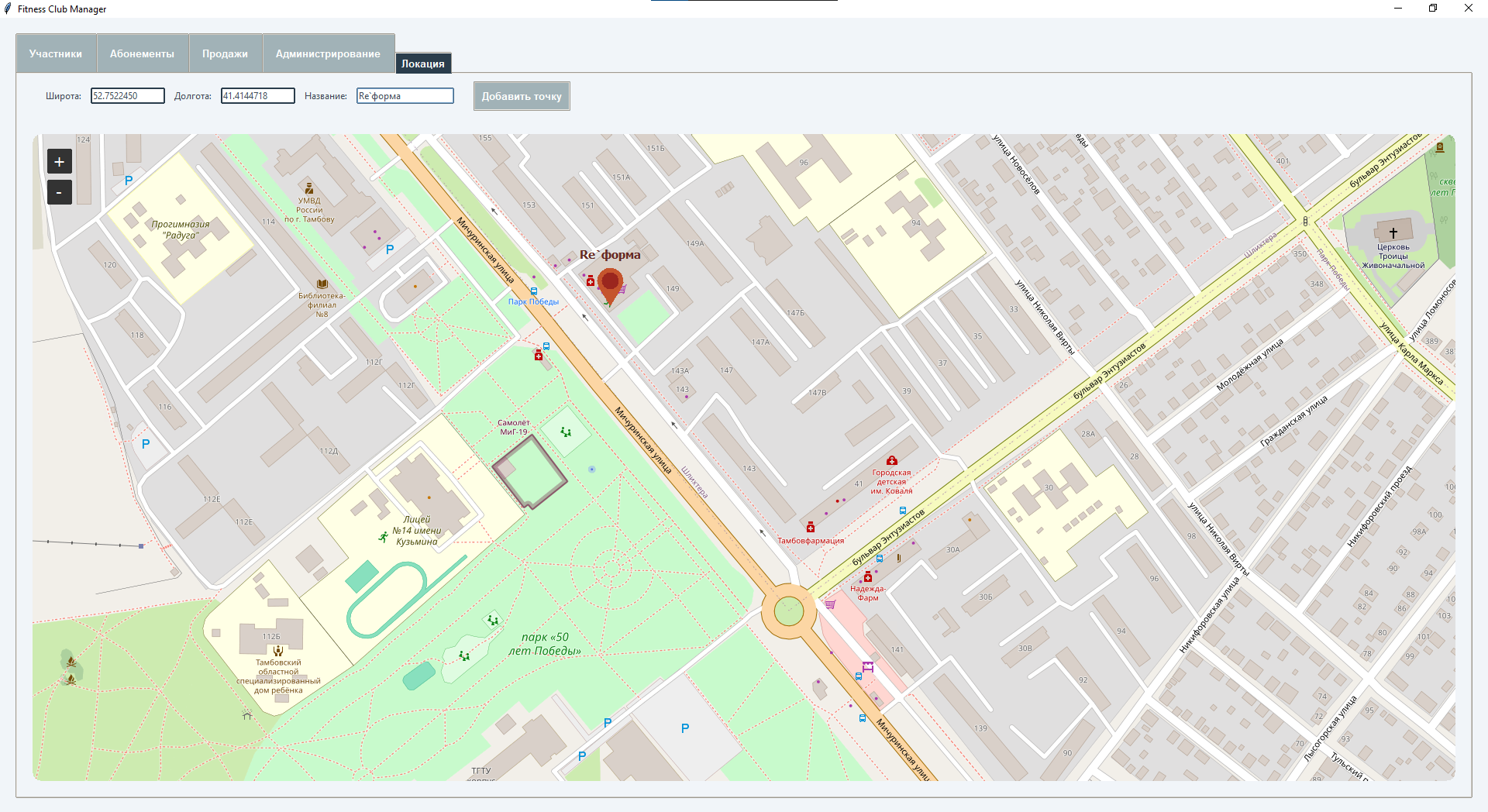




Рисунок 11 – Добавление точки на картах

8 Разработка программной документации

Программный модуль - приложение для управления фитнес-клубом, разработанное с использованием библиотеки *Tkinter* для *GUI* и *SQLite*3 для работы с базой данных. Приложение позволяет управлять участниками клуба, абонементами, продажами, отображает статистику и локацию клуба на карте.

Основные функции модуля:

1. Управление участниками:

* Добавление новых участников (ФИО, телефон, возраст).
* Отображение списка всех участников в таблице.
* Обновление списка участников.

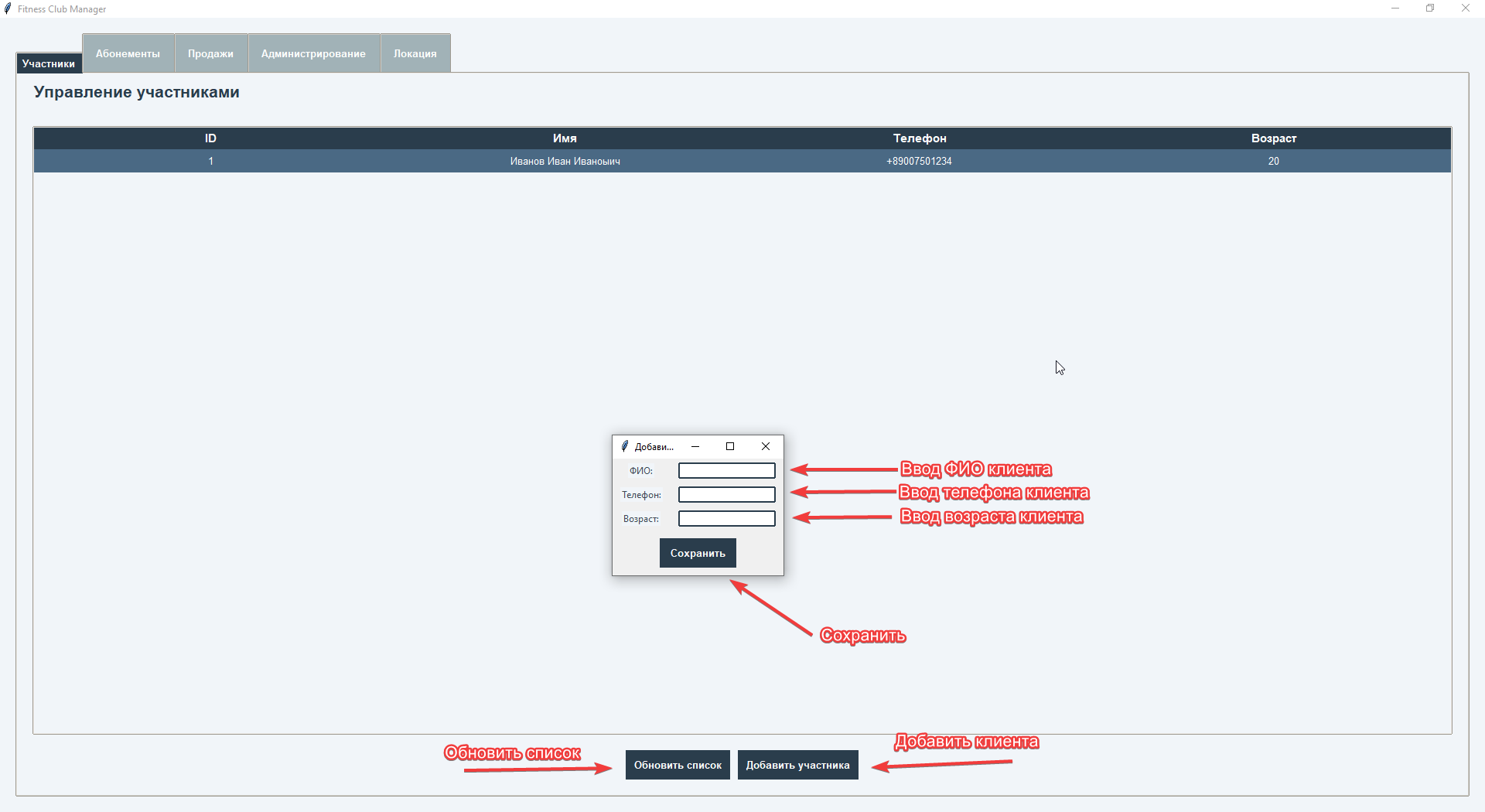


Рисунок 12 – Как добавить участника

1. Управление абонементами:

* Добавление новых абонементов (название, тип, длительность, цена).
* Отображение списка доступных абонементов.
* Обновление списка абонементов.

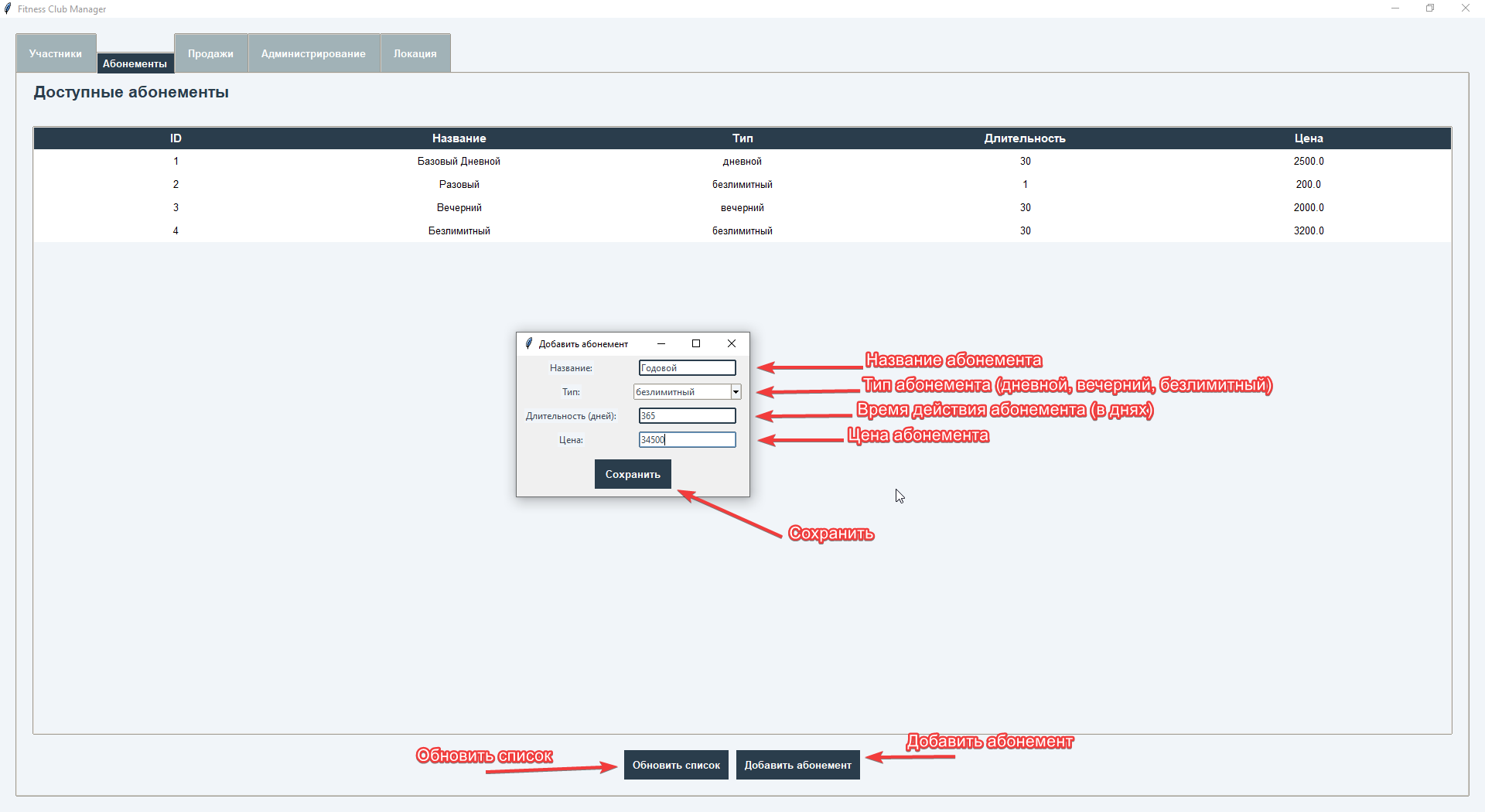


Рисунок 13 – Как добавить абонемент

1. Управление продажами:

* Оформление продажи абонемента участнику с возможностью указания скидки.
* Отображение активных абонементов с информацией о сроке действия.
* Обновление данных о продажах.

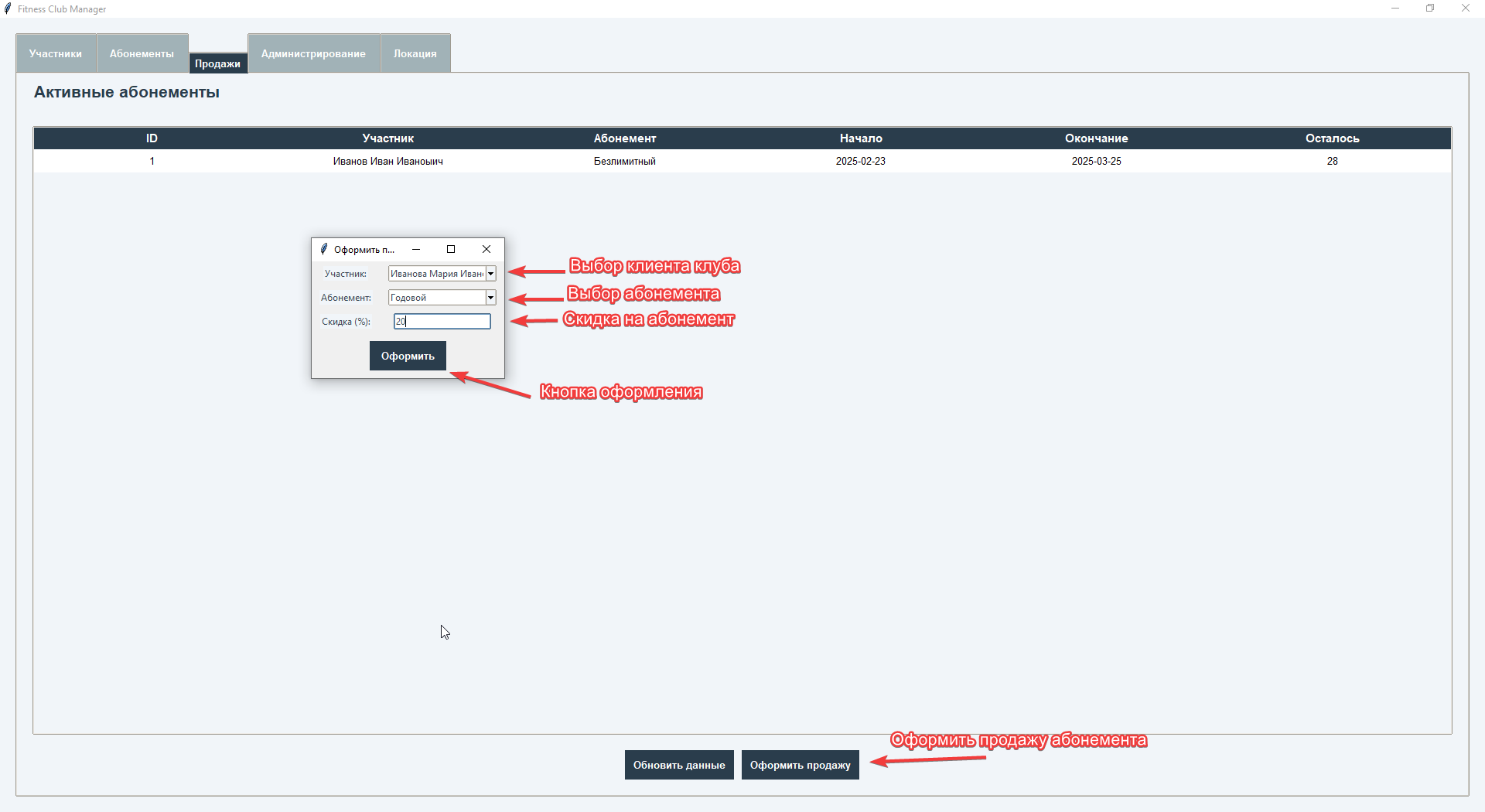


Рисунок 14 – Как добавить продажи

1. Администрирование:

* Отображение статистики по количеству участников и активных абонементов.
* Обновление статистики.

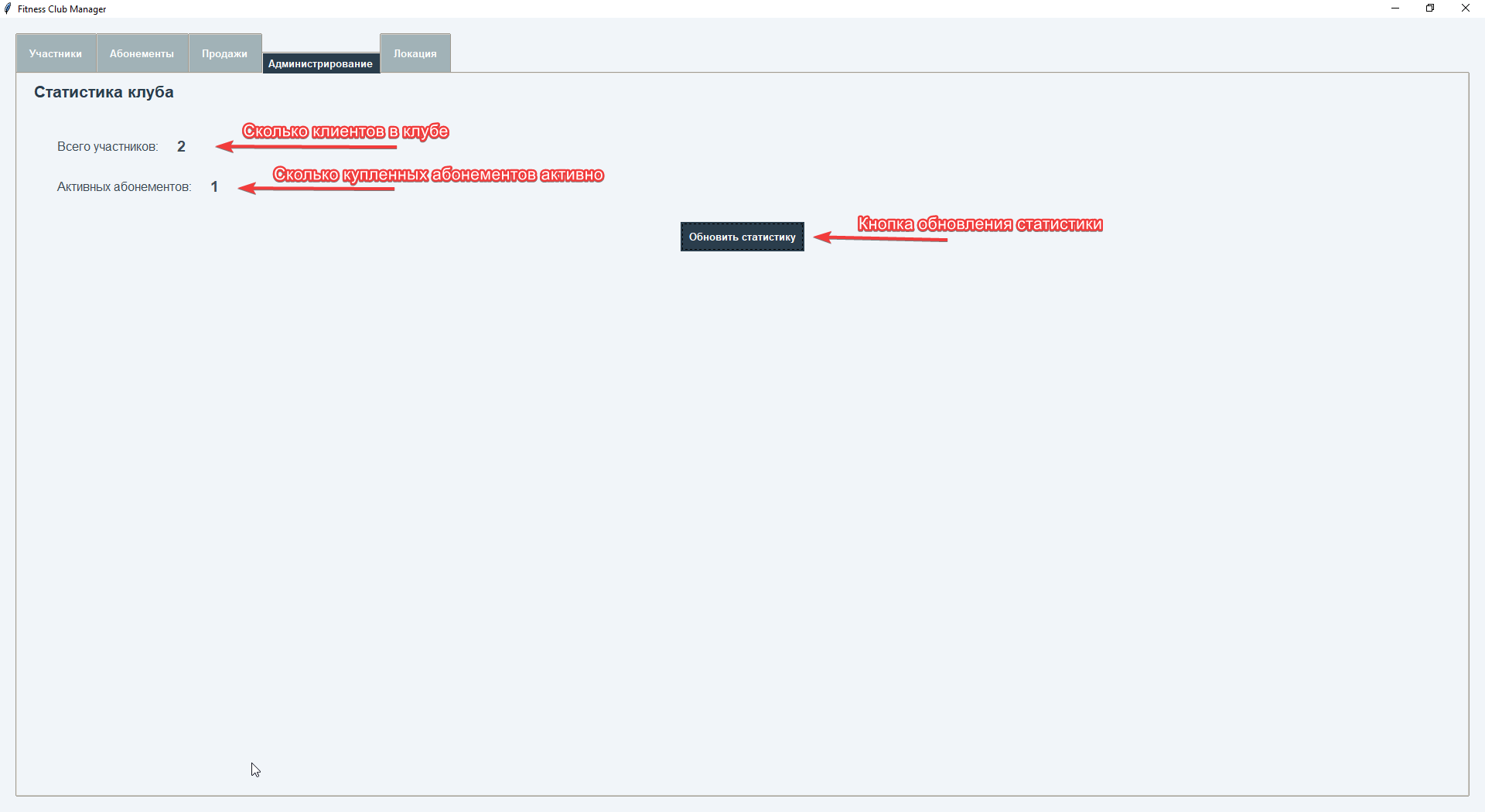


Рисунок 15 – Как обновить статистику

1. Карты:

* Отображение карты с возможностью добавления маркеров с указанием координат и названия.
* Установка маркера для главного фитнес-клуба.

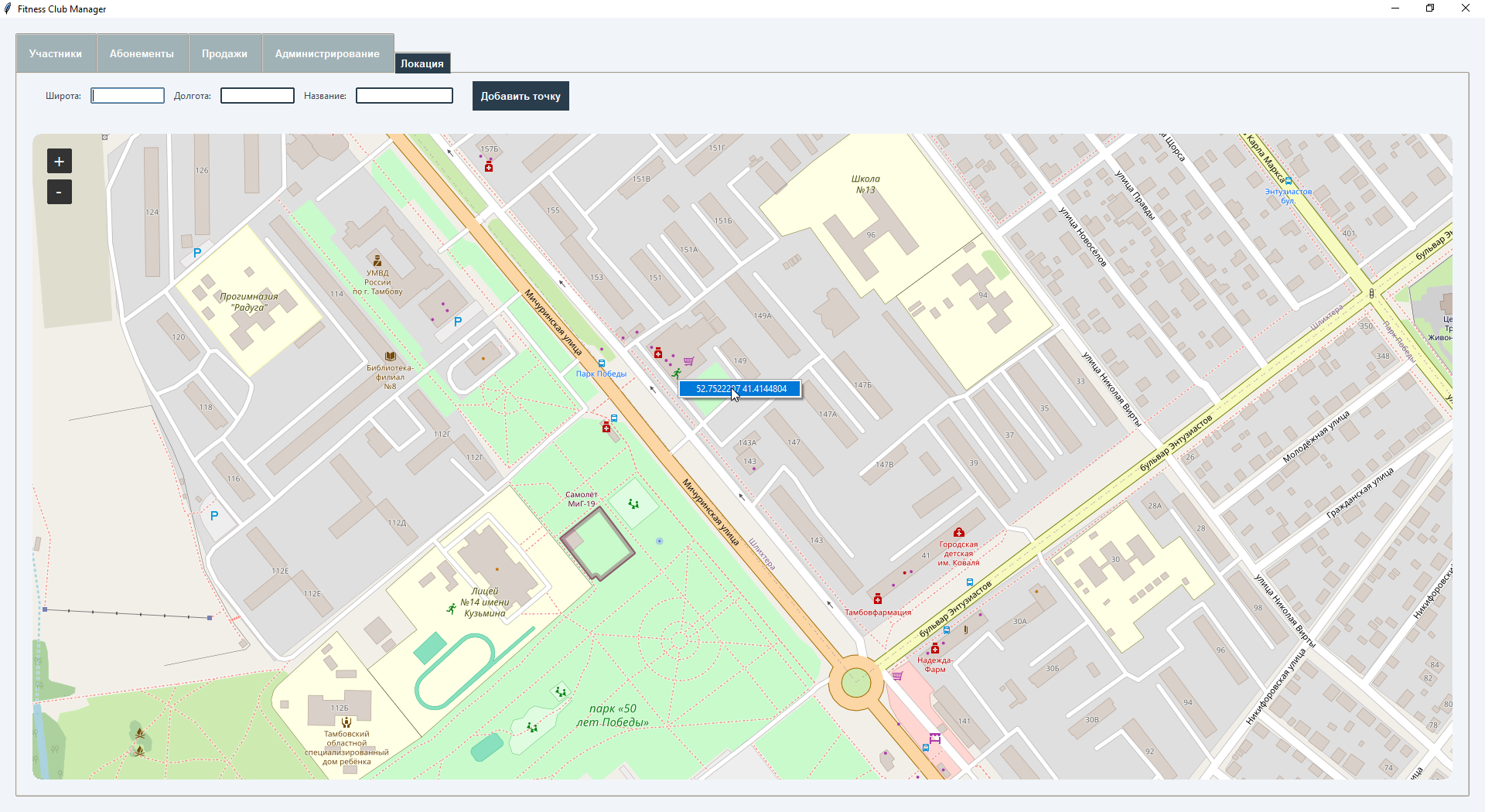


Рисунок 16 – Как получить координаты

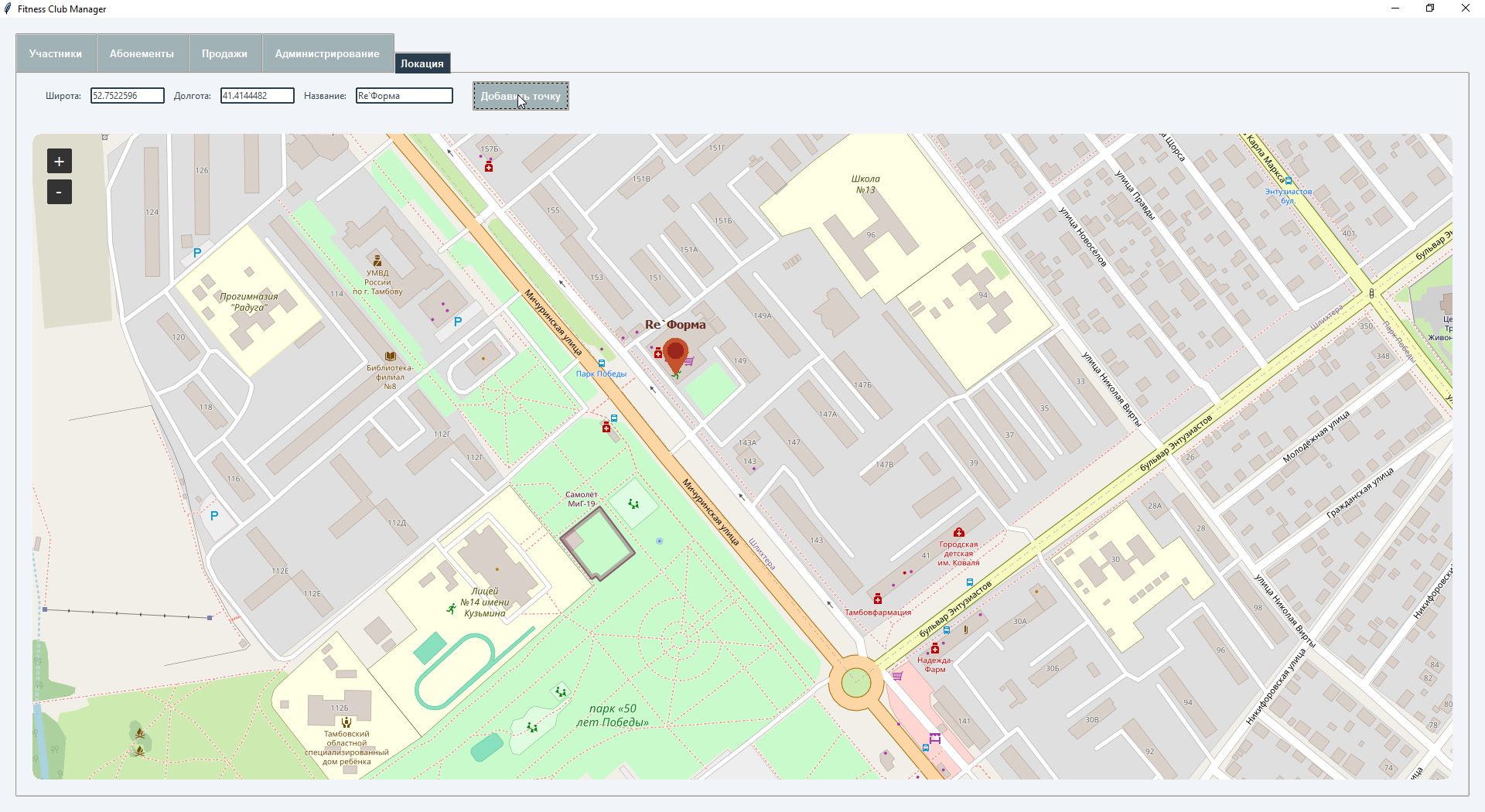


Рисунок 17 – Как добавить маркер

Чтобы добавить маркер необходимо:

1. Нажать по выбранной точке правой кнопкой мыши;
2. Нажать во всплывающем меню по кнопке с координатами;
3. Вставить координаты в соответствующие поля (для вставки необходимо нажать Shift + Insert);
4. Указать название точки;
5. Нажать «Добавить точку».

Работа с интерфейсом:

* Приложение имеет вкладки для каждого из разделов: "Участники", "Абонементы", "Продажи", "Администрирование", "Локация".
* На каждой вкладке доступны кнопки для добавления новых записей и обновления данных.
* Данные отображаются в таблицах, которые автоматически обновляются при добавлении новой информации.

Работа с базой данных:

* При запуске приложения автоматически создаются необходимые таблицы в базе данных, если они отсутствуют.
* Все данные сохраняются в *SQLite*-базе данных *fitness*\_*club*.*db*.
* Для работы с базой данных используются стандартные *SQL*-запросы.

Примеры использования:

1. Добавление участника:

* Перейдите на вкладку "Участники".
* Нажмите кнопку "Добавить участника".
* Заполните поля: ФИО, телефон, возраст.
* Нажмите "Сохранить".

1. Оформление продажи:

* Перейдите на вкладку "Продажи".
* Нажмите кнопку "Оформить продажу".
* Выберите участника и абонемент из выпадающих списков.
* Укажите скидку (если необходимо).
* Нажмите "Оформить".

1. Добавление маркера на карту:

* Перейдите на вкладку "Локация".
* Введите координаты (широта, долгота) и название точки.
* Нажмите "Добавить точку".

Заключение

Программный модуль предоставляет удобный интерфейс для управления основными аспектами работы фитнес-клуба. Все данные сохраняются в базе данных, что обеспечивает их целостность и доступность для дальнейшего анализа.