| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |
| --- | --- |

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

**О Т Ч Е Т  
по лабораторной работе № 2**

**Название:** Создание прототипа ИС на базе MongoDB

**Дисциплина:** Распределенные базы данных

| Студент | ИУ6-14М |  |  | А.И. Дегтярев | |
| --- | --- | --- | --- | --- | --- |
|  | (Группа) |  | (Подпись, дата) | | (И.О. Фамилия) |
|  |  |  |  | |  |
| Преподаватель |  |  |  | | М.М. Фомин |
|  |  |  | (Подпись, дата) | | (И.О. Фамилия) |

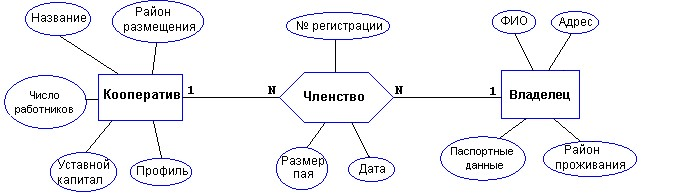
Москва, 2023

**Задание**

В ходе лабораторной работы требуется разработать:

* схемы бизнес-процессов в предметной области;
* схему взаимодействия сущностей;
* тексты скриптов для создания и заполнения коллекций;
* тексты скриптов для поиска и выдачи данных по базовым сущностям;
* скриншоты интерфейсов и заполненных коллекций.

Предметная область для практических заданий: **Кооператив**

****

*Цель работы*: научиться создавать БД в MongoDB.

**1 Схемы бизнес-процессов в предметной области**

В данной лабораторной работе были рассмотрены бизнес-процессы регистрации кооператива в ИС и передачи пая.

Регистрацию кооператива предлагается делать на основе документа решения о создании кооператива, после подтверждения факта внесения регистрации в ЕГРЮЛ, так как в нем представлена информация как о составе учредителей, так и юридическая информация о кооперативе. На рисунке 1.1 представлена верхнеуровневая декомпозиция регистрации кооператива в ИС в виде IDEF0 диаграммы.

По ГК РФ владельцы могут передать свой пай или его часть другим владельцам кооператива если иное не установлено уставом. Передача пая гражданину, не являющемуся членом кооператива допускается лишь с согласия кооператива. На рисунке 1.2 представлена верхнеуровневая декомпозиция передачи пая в виде IDEF0 диаграммы.

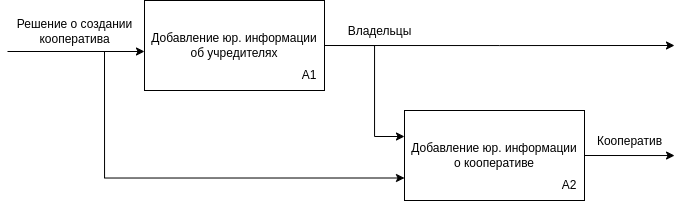
****

Рисунок 1.1 – Декомпозиция процесса регистрации кооператива в ИС

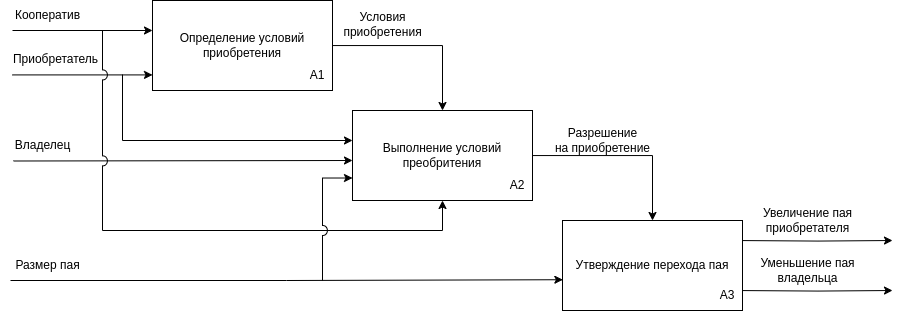
****

Рисунок 1.2 – Декомпозиция процесса передачи пая

**2 Схема взаимодействия сущностей**

Для реализации рассмотренных процессов следует выделить следующие сущности: Владелец, Кооператив и Членство. Схема их взаимодействия соответствует представленной в задании.

Для хранении информации об этих сущностей были созданы 3 коллекции, каждая соответствует своей сущности. Документы коллекции Членства связывают документы Кооператив и Владелец.

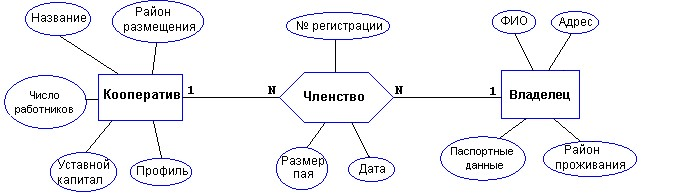
****

Рисунок 2.1 – Схема взаимодействия сущностей

**3 Скрипты для создания и заполнения коллекций**

На листинге 3.1 представлен скрипт заполнения коллекций тестовыми данными в соответствии с таблицей 3.1. В коллекции Членство было создано 2 индекса для полей с внешними ключами для увеличения скорости запросов по поиску владельцев кооператива.

Таблица 3.1 – Размеры паев владельцев кооперативов в тестовых данных

|  | Иванов | Петухов | Кузнецов |
| --- | --- | --- | --- |
| Мир овощей | 101 | – | – |
| Мир пуговиц | 102 | 202 | – |
| Мир ручек | 103 | 203 | 303 |

Листинг 3.1 – Заполнение коллекций тестовыми данными

| from settings import DATABASE\_URI from pymongo import MongoClient  db = MongoClient(host=DATABASE\_URI).get\_default\_database()  o1, o2, o3 = db.owners.insert\_many([{  "name": "Иванов Иван Иванович",  "address": "Январская улица, д 1",  "residence": "Россия",  "passport": "1111222233334444", }, {  "name": "Петухов Петр Петрович",  "address": "Январская улица, д 1",  "residence": "Россия",  "passport": "1111222233334444", }, {  "name": "Кузнецов Кирилл Кириллович",  "address": "Январская улица, д 1",  "residence": "Россия",  "passport": "1111222233334444", }]).inserted\_ids  c1, c2, c3 = db.cooperatives.insert\_many([{  "name": "Мир овощей",  "residence": "Россия",  "workers\_number": 11,  "capital\_amount": 101,  "profile": "Продукты", }, {  "name": "Мир пуговиц",  "residence": "Россия",  "workers\_number": 22,  "capital\_amount": 304,  "profile": "Галантерея", }, {  "name": "Мир ручек",  "residence": "Россия",  "workers\_number": 33,  "capital\_amount": 609,  "profile": "Канцелярия", }]).inserted\_ids  today = "2024-01-01"  db.memberships.insert\_many([{  "number": 1,  "owner\_id": o1, "cooperative\_id": c1,  "amount": 101,  "date": today, }, {  "number": 2,  "owner\_id": o1, "cooperative\_id": c2,  "amount": 102,  "date": today, }, {  "number": 3,  "owner\_id": o2, "cooperative\_id": c2,  "amount": 202,  "date": today, }, {  "number": 4,  "owner\_id": o1, "cooperative\_id": c3,  "amount": 103,  "date": today, }, {  "number": 5,  "owner\_id": o2, "cooperative\_id": c3,  "amount": 203,  "date": today, }, {  "number": 6,  "owner\_id": o3, "cooperative\_id": c3,  "amount": 303,  "date": today, }])  db.memberships.create\_index("owner\_id") db.memberships.create\_index("cooperative\_id") |
| --- |

**4 Скрипты для поиска и выдачи данных по базовым сущностям**

Лабораторная работа была выполнена на языке Python 3.9.18 в виде простого веб сервера на [Flask](https://flask.palletsprojects.com/en/3.0.x/) и библиотеки валидации веб форм [wtforms](https://wtforms.readthedocs.io/en/3.1.x/). Работа с MongoDB происходит с помощью пакета pymongo.

На листинге 4.1 представлен скрипт создания Flask приложения с обработчиками следующих путей:

* / — выводит список созданных кооперативов в виде таблицы;
* /cooperatives/register — форма регистрации кооператива позволяет выполнить соответствующий бизнес-процесс;
* /cooperatives/<cooperative\_id>/memberships — выводит список членств определенного кооператива;
* /memberships/<membership\_id>/transfer — форма передачи пая позволяет выполнить соответствующий бизнес-процесс;
* /cooperatives/search — поиск кооператива по имени;
* /owners — выводит список зарегистрированных в ИС владельцев в виде таблицы;
* /owners/create — форма добавления нового владельца;
* /owners/search — поиск владельца по ФИО.

На листинге 4.2 представлено описание форм, которые отвечают за валидацию данных в теле POST-запроса и сохранение этих данных в базу:

* CreateOwnerForm — создание нового владельца;
* RegisterCooperativeForm — регистрация нового кооператива, форма создает документы в коллекции кооперативов и членства. Это происходит в одной форме, так как кооператив не может быть без членов;
* TransferMembershipForm — передача пая, форма создает 2 документа в коллекции членства: с уменьшенным размером пая владельца и новым размером пая получателя.

Листинг 4.1 – Обработчики Flask приложения, app.py

| from bson import ObjectId from flask import Flask, abort, redirect, render\_template, request from flask\_wtf.csrf import CSRFProtect  from db import db from forms import CreateOwnerForm, RegisterCooperativeForm, \  TransferMembershipForm   app = Flask(\_\_name\_\_) app.config.from\_pyfile("settings.py") csrf = CSRFProtect(app)   @app.route("/", methods=["GET"]) def index():  cooperatives = list(db.cooperatives.find(limit=25, sort=[('name', 1)]))  return render\_template("pages/index.html", cooperatives=cooperatives)   @app.route("/cooperatives/register", methods=["GET", "POST"]) def register\_cooperative():  form = RegisterCooperativeForm()  if form.validate\_on\_submit():  form.save()  return redirect('/', code=302)   return render\_template("pages/cooperatives/register.html", form=form)   @app.route("/cooperatives/<cooperative\_id>/memberships", methods=["GET"]) def view\_cooperative\_memberships(cooperative\_id):  try:  cooperative\_id = ObjectId(cooperative\_id)  except:  return abort(400)   cooperative = db.cooperatives.find\_one({"\_id": cooperative\_id})  if not cooperative:  return abort(404)   all\_memberships = list(db.memberships.find({  "cooperative\_id": cooperative\_id  }, sort=[("number", -1)]))   owners\_set = {m.get("owner\_id") for m in all\_memberships}  owners\_filter = {"\_id": {"$in": list(owners\_set)}}  search = request.args.get("q", "")  if search:  owners\_filter["name"] = {  "$regex": search,  "$options": "i",  }   owners = {o["\_id"]: o for o in db.owners.find(owners\_filter)}  showed\_owners = set()   memberships = []  for member in all\_memberships:  owner\_id = member.get('owner\_id')  member["can\_transfer"] = (  owner\_id not in showed\_owners and  member.get("amount", 0) > 0  )  member["owner"] = owners.get(owner\_id)  showed\_owners.add(owner\_id)  if member["owner"]:  memberships.append(member)   return render\_template(  "pages/cooperatives/memberships.html",  cooperative=cooperative,  memberships=memberships,  search=search  )   @app.route("/memberships/<membership\_id>/transfer", methods=["GET", "POST"]) def transfer\_membership(membership\_id):  try:  membership\_id = ObjectId(membership\_id)  except:  return abort(400)   membership = db.memberships.find\_one({"\_id": membership\_id})  if not membership:  return abort(404)   membership["owner"] = db.owners.find\_one({"\_id": membership.get("owner\_id")})  membership["cooperative"] = db.cooperatives.find\_one({"\_id": membership.get("cooperative\_id")})   form = TransferMembershipForm(membership)  if form.validate\_on\_submit():  form.save()  return redirect(f"/cooperatives/{membership['cooperative\_id']}/memberships", code=302)   return render\_template("pages/memberships/transfer.html", form=form)   @app.route("/cooperatives/search", methods=["GET"]) def search\_cooperative():  filter = {}  search = request.args.get("q", "")  if search:  filter["name"] = {  "$regex": search,  "$options": "i",  }  cooperatives = list(db.cooperatives.find(filter, limit=25, sort=[('name', 1)]))   return render\_template("pages/cooperatives/search.html", cooperatives=cooperatives, search=search)   @app.route("/owners", methods=["GET"]) def list\_owners():  owners = list(db.owners.find(limit=25, sort=[('name', 1)]))  return render\_template("pages/owners/index.html", owners=owners)   @app.route("/owners/create", methods=["GET", "POST"]) def create\_owner():  form = CreateOwnerForm()  if form.validate\_on\_submit():  form.save()  return redirect("/owners", code=302)   return render\_template("pages/owners/create.html", form=form)   @app.route("/owners/search", methods=["GET"]) def search\_owner():  filter = {}  search = request.args.get("q", "")  if search:  filter["name"] = {  "$regex": search,  "$options": "i",  }  owners = list(db.owners.find(filter, limit=25, sort=[('name', 1)]))   return render\_template("pages/owners/search.html", owners=owners, search=search) |
| --- |

Листинг 4.1 – Обработчики Flask приложения, app.py

| import datetime from typing import Optional from bson import ObjectId from flask\_wtf import FlaskForm from wtforms import BooleanField, StringField, IntegerField, FormField, \  FieldList, Form, DateField, ValidationError from wtforms.widgets import HiddenInput from wtforms.validators import DataRequired, NumberRange  from db import db from widgets import CooperativeMembershipListWidget, OwnerForeignKeyWidget   def date2str(date: Optional[datetime.datetime]):  if date:  return date.strftime("%Y-%m-%d")   class CreateOwnerForm(FlaskForm):  name = StringField('ФИО', validators=[DataRequired()])  residence = StringField('Район проживания')  address = StringField('Адрес')  passport = StringField('Паспортные данные')   def save(self):  data = {  name: f.data  for name, f in self.\_fields.items()  if f is not None and not isinstance(f.widget, HiddenInput)  }   return db.owners.insert\_one(data)   class RegisterCooperativeForm(FlaskForm):  class MembershipForm(Form):  number = IntegerField('Номер регистрации', validators=[DataRequired(), NumberRange(min=1)])  owner\_id = StringField('Владелец', validators=[DataRequired()], widget=OwnerForeignKeyWidget())  amount = IntegerField('Размер пая', validators=[DataRequired(), NumberRange(min=1)])  date = DateField('Дата', validators=[DataRequired()], default=datetime.datetime.today)   name = StringField('Название', validators=[DataRequired()])  residence = StringField('Район размещения', default="Россия")  workers\_number = IntegerField('Количество работников (человек)', validators=[NumberRange(min=1)], default=10)  profile = StringField('Профиль', default="Разное")  memberships = FieldList(FormField(MembershipForm), 'Членство', min\_entries=1, widget=CooperativeMembershipListWidget())   def save(self):  cooperative\_id = db.cooperatives.insert\_one({  'name': self.name.data,  'profile': self.profile.data,  'workers\_number': self.workers\_number.data,  'residence': self.residence.data,  'capital\_amount': sum(member['amount'] for member in self.memberships.data),  }).inserted\_id   db.memberships.insert\_many([  {  'number': member['number'],  'owner\_id': ObjectId(member['owner\_id']),  'amount': member['amount'],  'date': date2str(member['date']),  'cooperative\_id': cooperative\_id,  }  for member in self.memberships.data  ])   class TransferMembershipForm(FlaskForm):  buyer\_id = StringField('Получатель пая', validators=[DataRequired()], widget=OwnerForeignKeyWidget())  cooperative\_approve = BooleanField('Согласие кооператива было получено?')  amount = IntegerField('Размер пая', validators=[DataRequired(), NumberRange(min=1)])  number = IntegerField('Номер регистрации', validators=[DataRequired(), NumberRange(min=1)])  date = DateField('Дата', validators=[DataRequired()], default=datetime.datetime.today)   def \_\_init\_\_(self, membership, \*args, \*\*kwargs):  super().\_\_init\_\_(\*args, \*\*kwargs)  self.membership = membership  self.cooperative\_id = self.membership.get("cooperative\_id")  self.buyer\_membership = None   def validate\_number(self, field):  number = field.data  last\_membership = db.memberships.find\_one(  {  "cooperative\_id": self.cooperative\_id,  },  sort=[('number', -1)],  )   if last\_membership and last\_membership.get("number") >= number:  raise ValidationError(f"Номер должен быть больше {last\_membership.get('number')}")   def validate\_buyer\_id(self, field):  buyer\_id = ObjectId(field.data)  if buyer\_id == self.membership.get("owner\_id"):  raise ValidationError(f"Владелец и получатель должны отличаться")   self.buyer\_membership = db.memberships.find\_one(  {  "owner\_id": buyer\_id,  "cooperative\_id": self.cooperative\_id,  },  sort=[('number', -1)],  )  if self.buyer\_membership and self.buyer\_membership.get("amount", 0) <= 0:  self.buyer\_membership = None   def validate\_amount(self, field):  owner\_amount = self.membership["amount"]  if field.data > owner\_amount:  raise ValidationError(f"Владелец не может передать пай, больше чем {owner\_amount}")   def validate(self, extra\_validators=None) -> bool:  valid = super().validate(extra\_validators)   if not self.buyer\_membership and not self.cooperative\_approve.data:  self.cooperative\_approve.errors.append(  "Передача пая гражданину, не являющемуся членом кооператива "  "допускается лишь с согласия кооператива"  )  valid = False   return valid   def save(self):  buyer\_amount = self.amount.data  if self.buyer\_membership:  buyer\_amount += self.buyer\_membership.get('amount', 0)   db.memberships.insert\_many([{  'number': self.number.data,  'owner\_id': ObjectId(self.buyer\_id.data),  'amount': buyer\_amount,  'date': date2str(self.date.data),  'cooperative\_id': self.cooperative\_id,  }, {  'number': self.number.data,  'owner\_id': self.membership['owner\_id'],  'amount': self.membership['amount'] - self.amount.data,  'date': date2str(self.date.data),  'cooperative\_id': self.cooperative\_id,  }]) |
| --- |

**5 Скриншоты интерфейсов и заполненных коллекций**

Интерфейс пользователя доступен по соответствующим путям в браузере. Для улучшения внешнего вида была использована библиотека [Simple.css](https://simplecss.org/).

На рисунках 5.1-5.3 представлены шаги создания нового владельца, на 5.4-5.8 – шаги регистрации кооператива, на 5.9-5.11 – шаги передачи пая владельца другому владельцу. На листинге 5.1 представлены заполненные коллекции после указанных шагов.

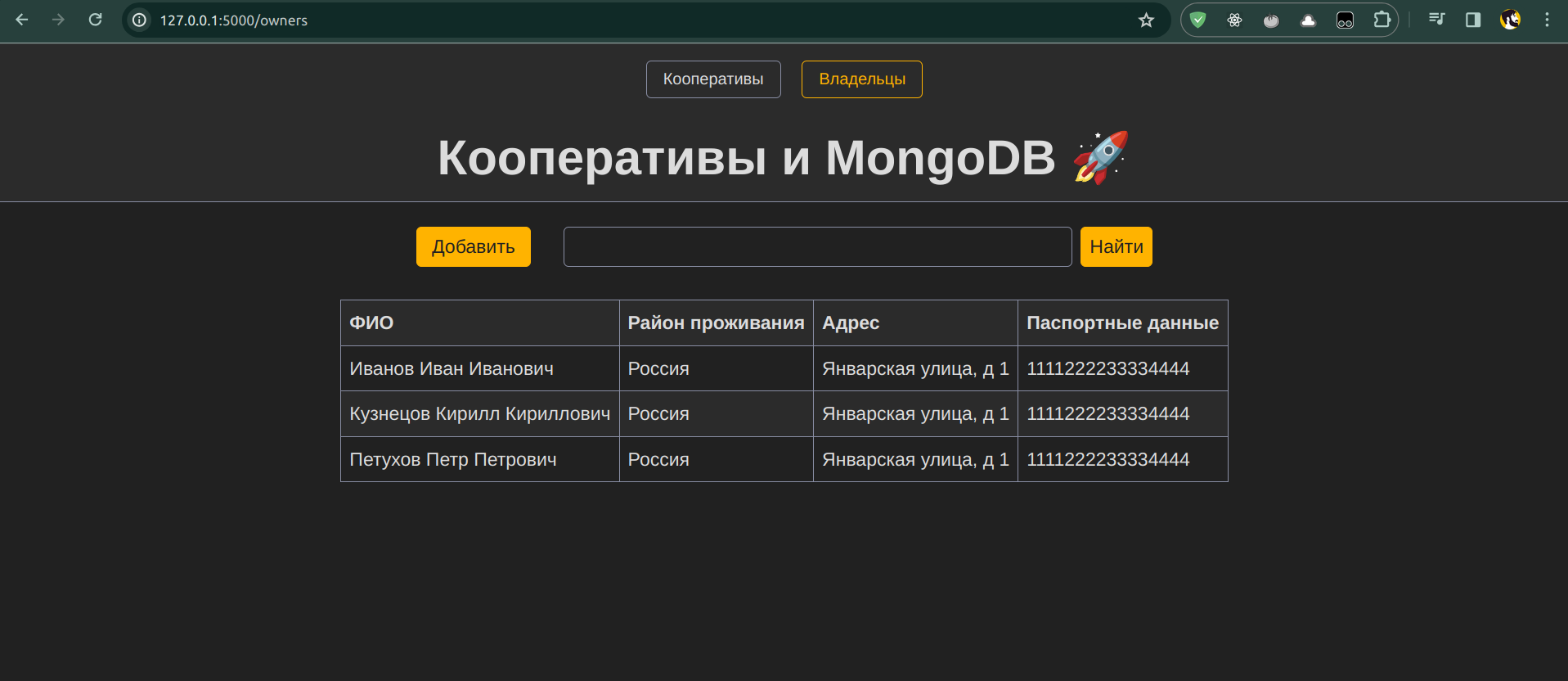


Рисунок 5.1 – Список владельцев с тестовыми данными

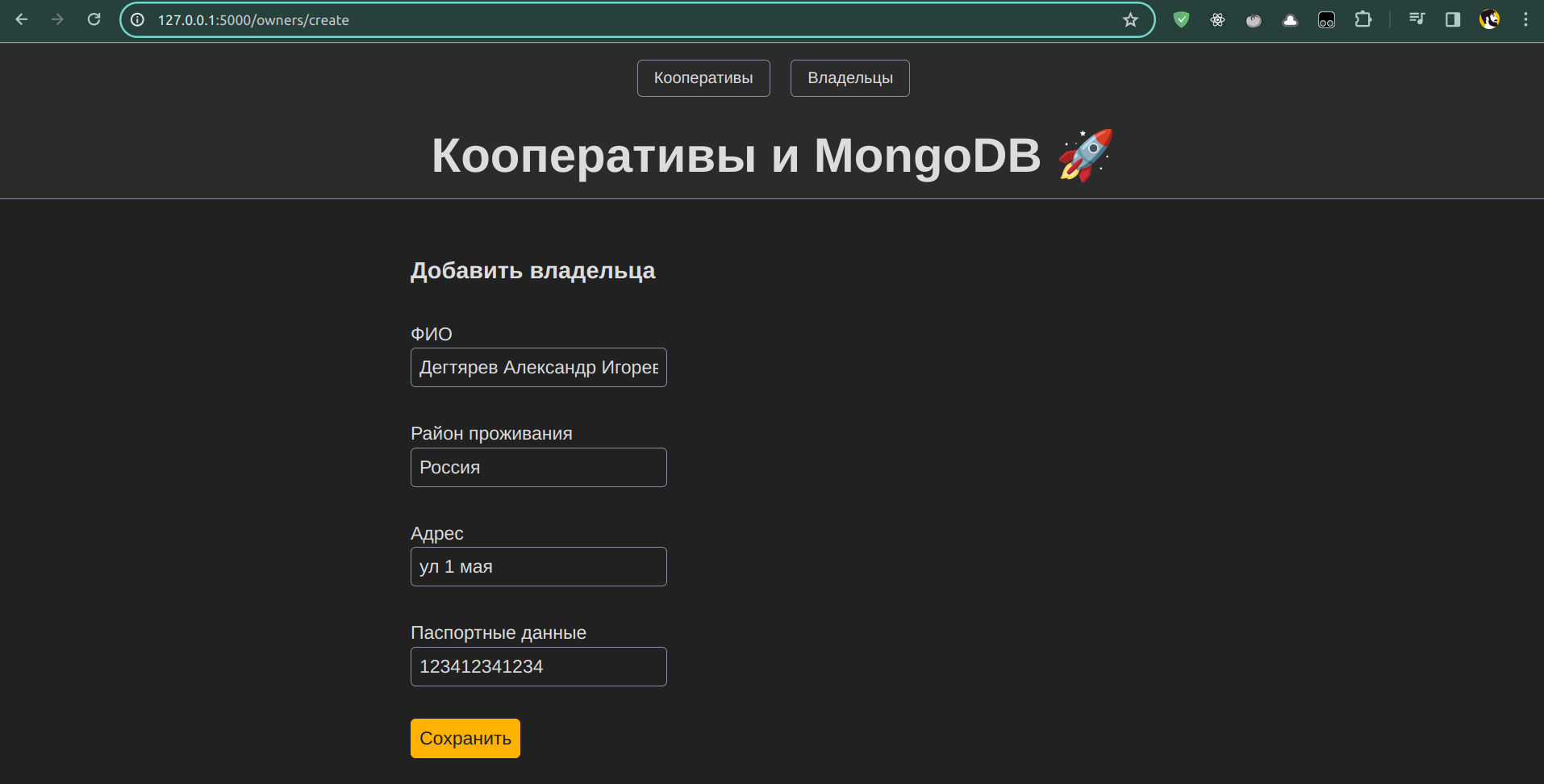


Рисунок 5.2 – Добавление владельца

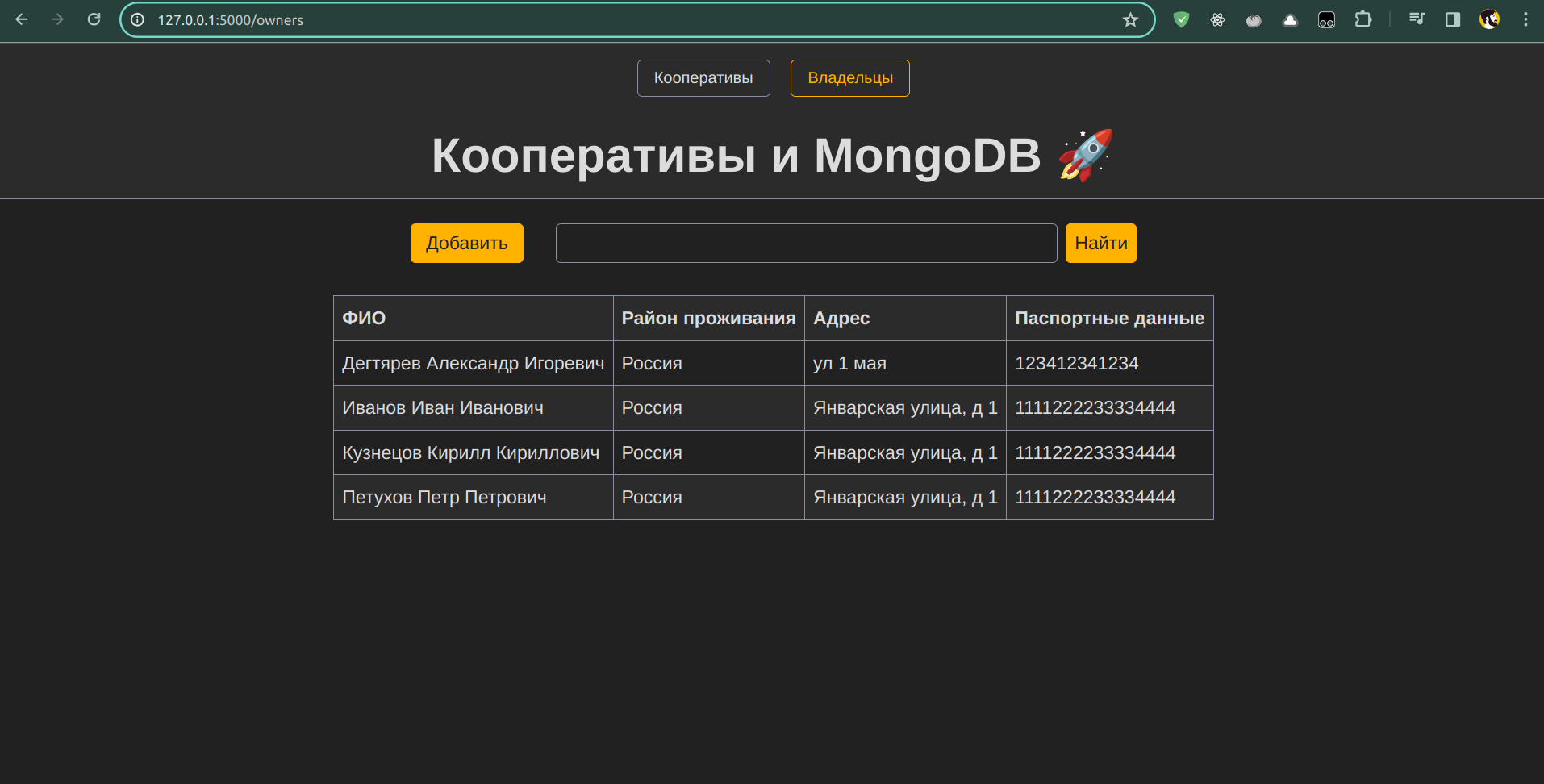


Рисунок 5.3 – Результат добавления

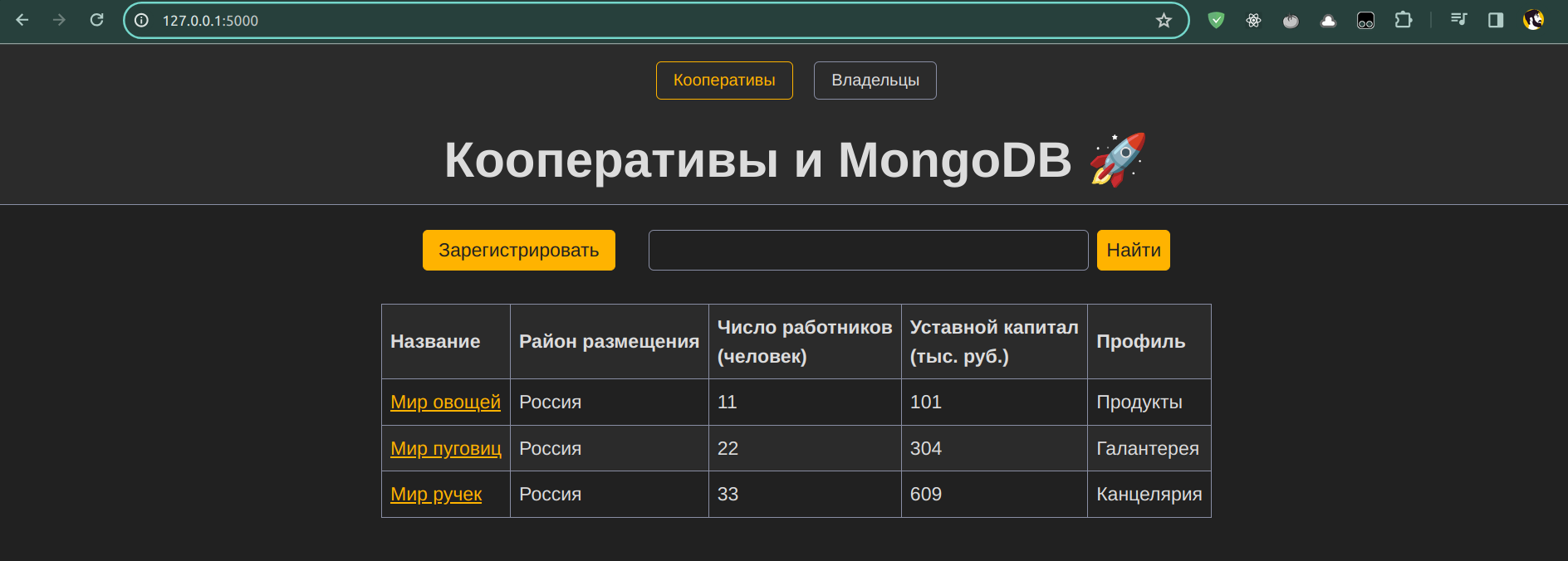
****

Рисунок 5.4 – Список кооперативов с тестовыми данными

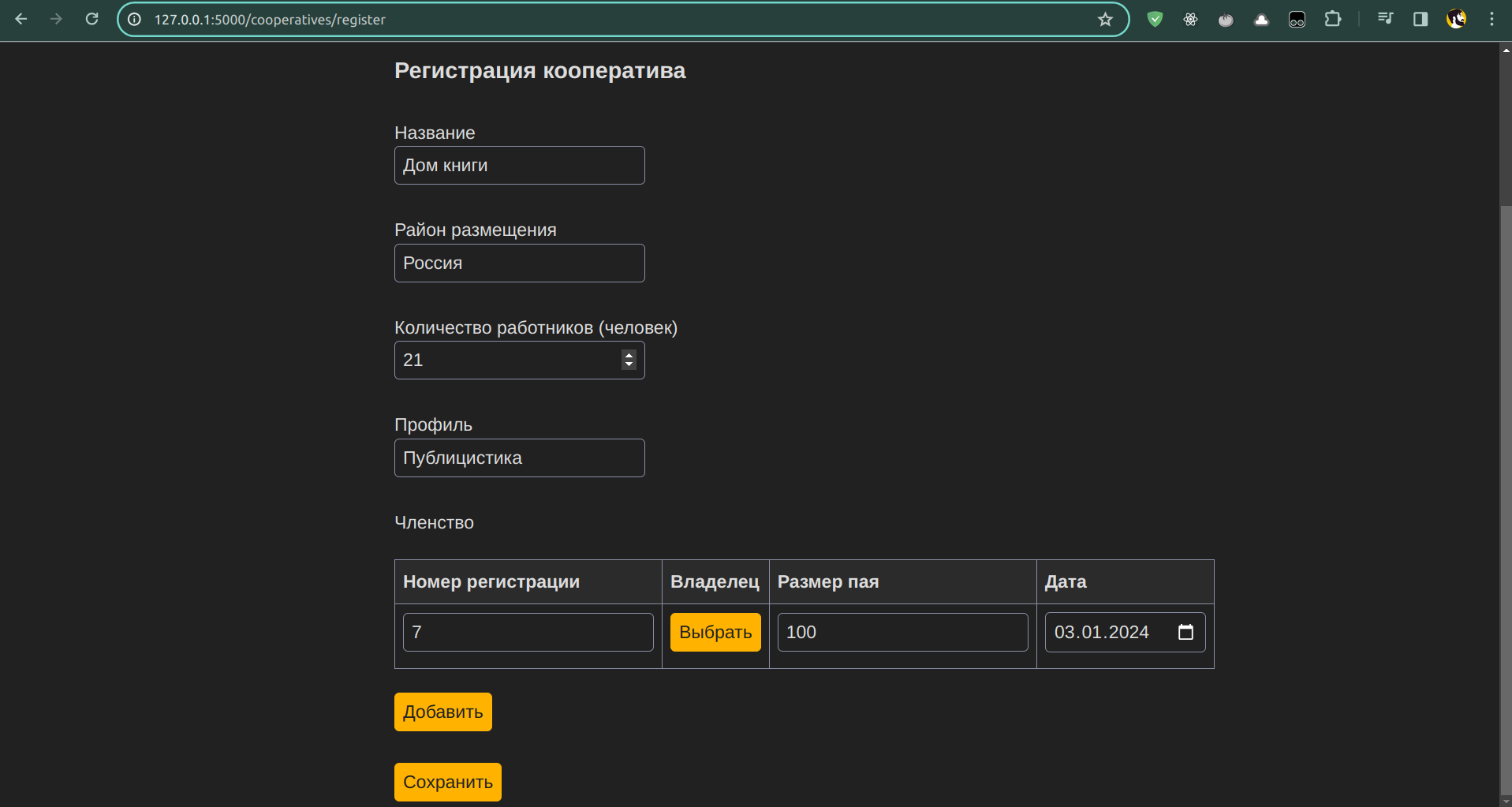
****

Рисунок 5.5 – Заполнение юр. информации о кооперативе

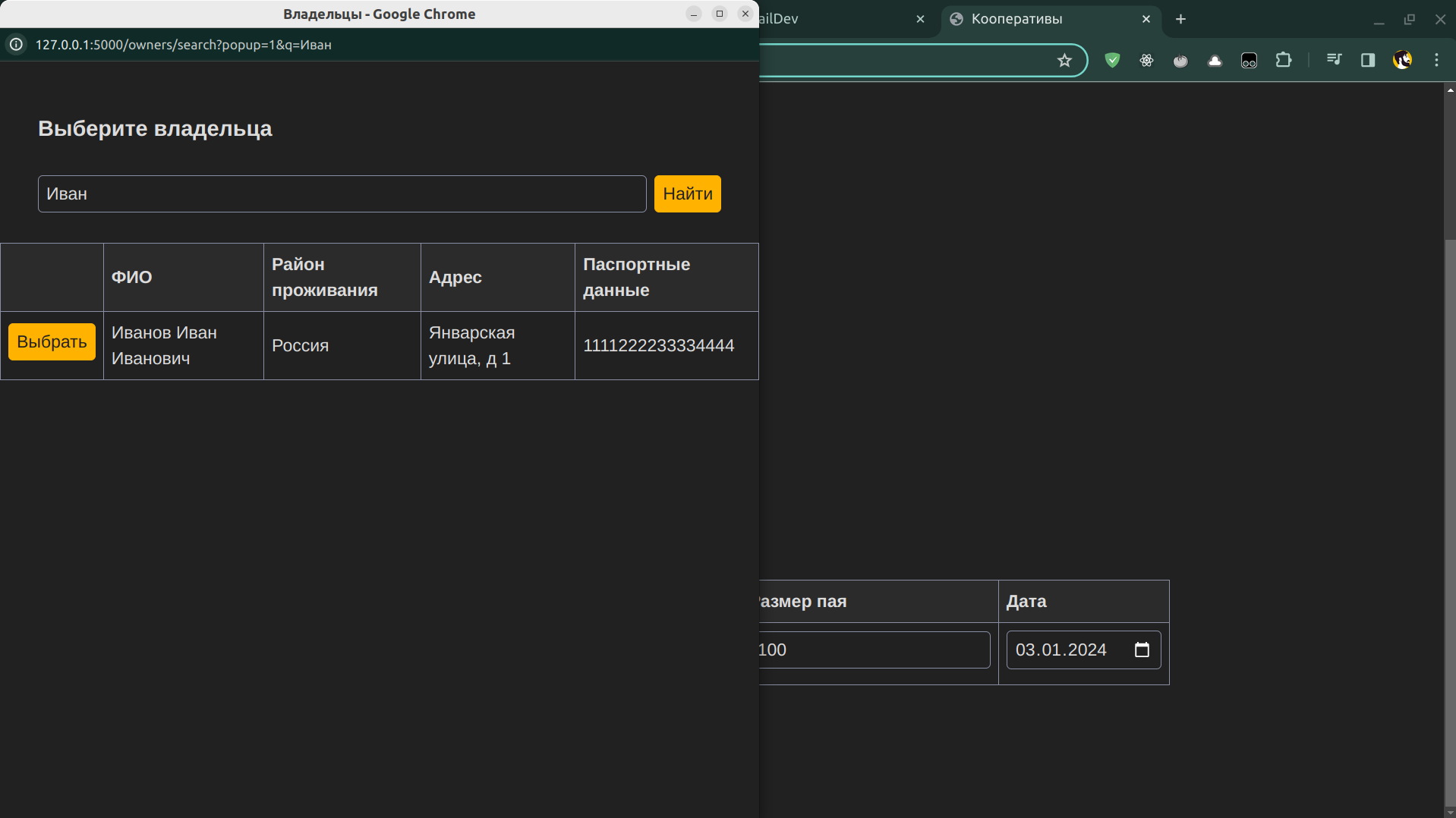
****

Рисунок 5.6 – Поиск и выбор владельца в ИС

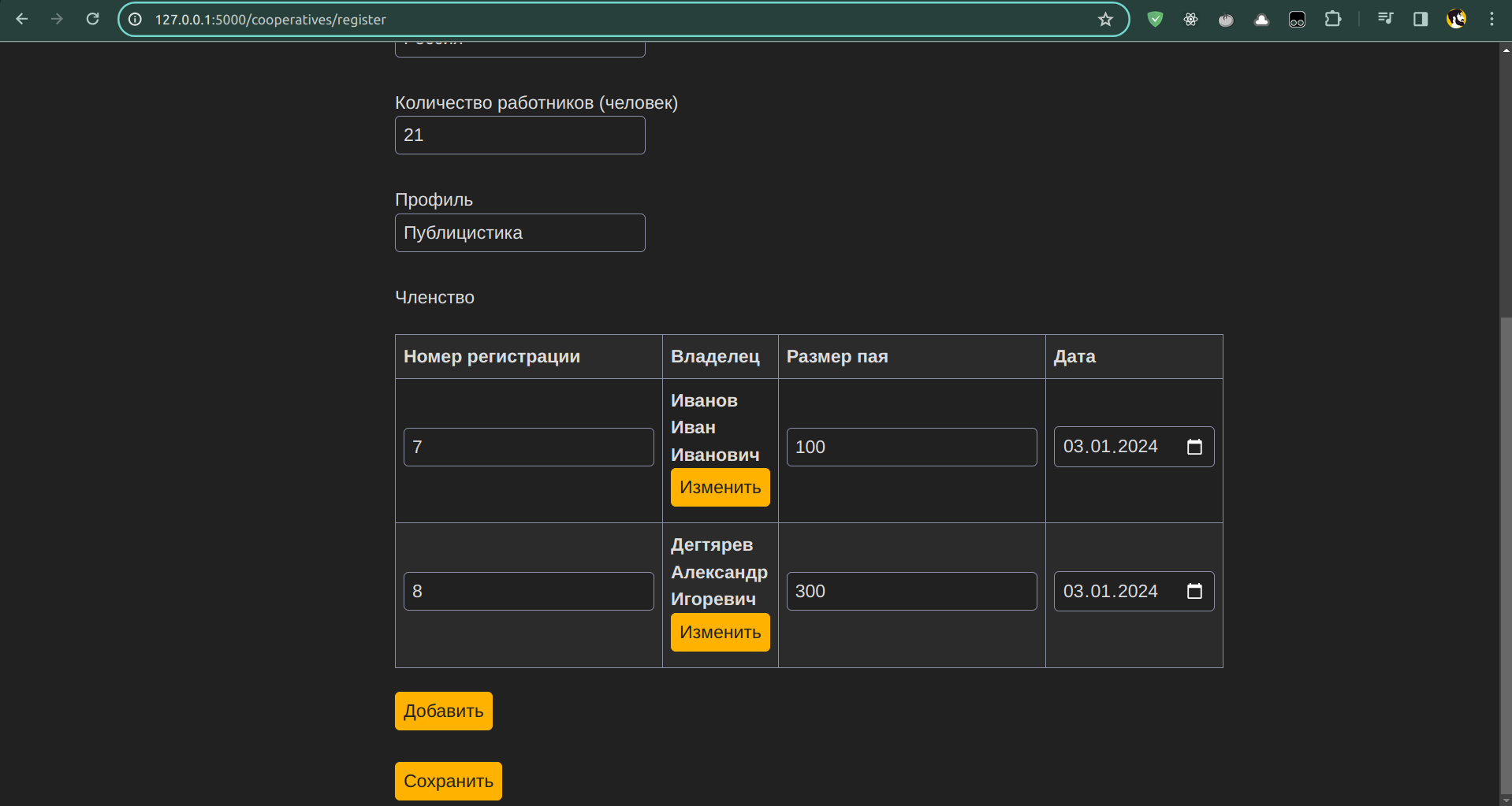
****

Рисунок 5.7 – Заполнение информации о членах кооператива

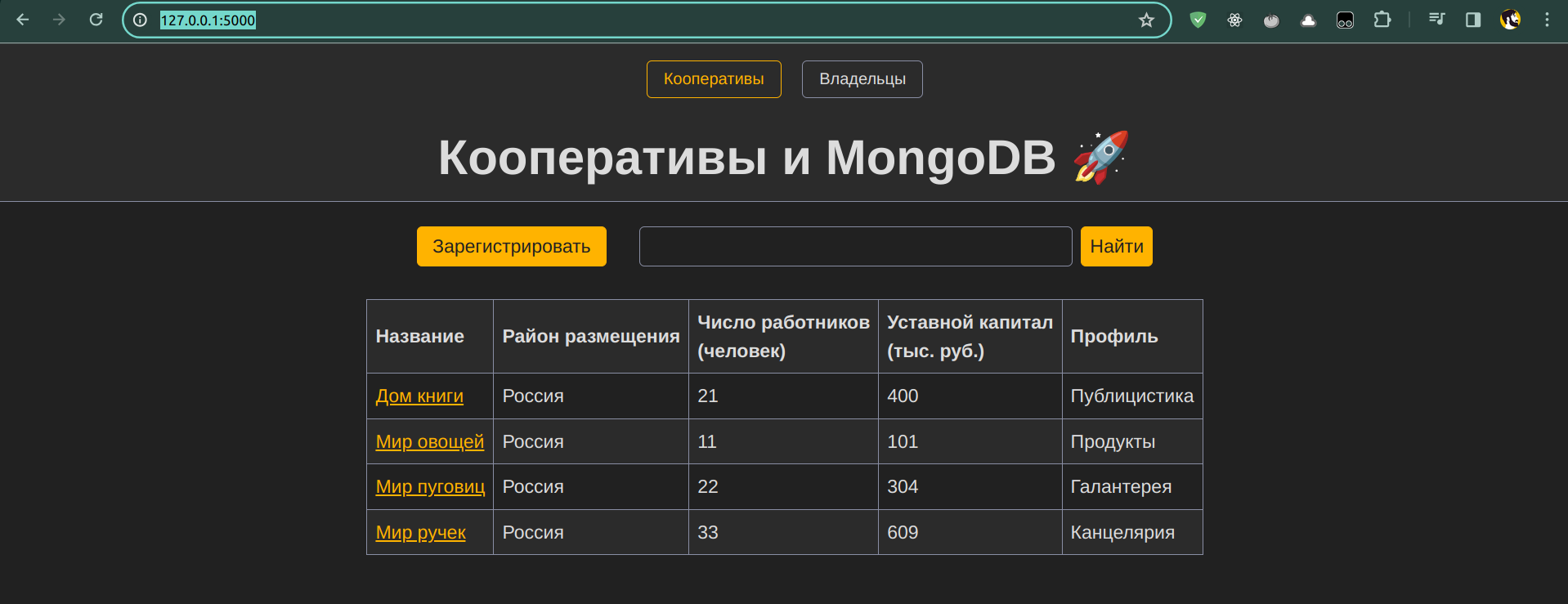
****

Рисунок 5.8 – Результат регистрации кооператива

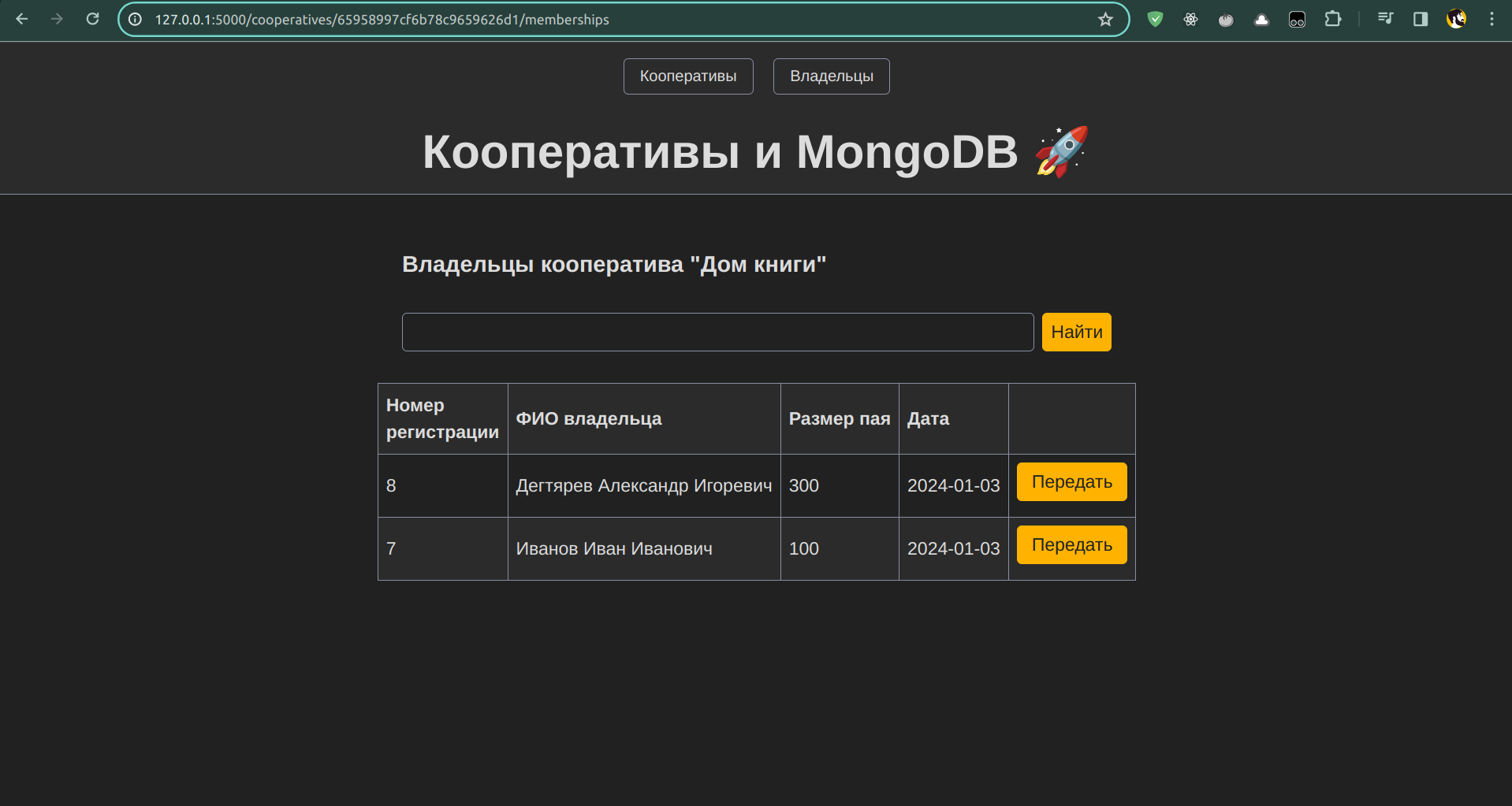
****

Рисунок 5.9 – Список членов кооператива “Дом книги”

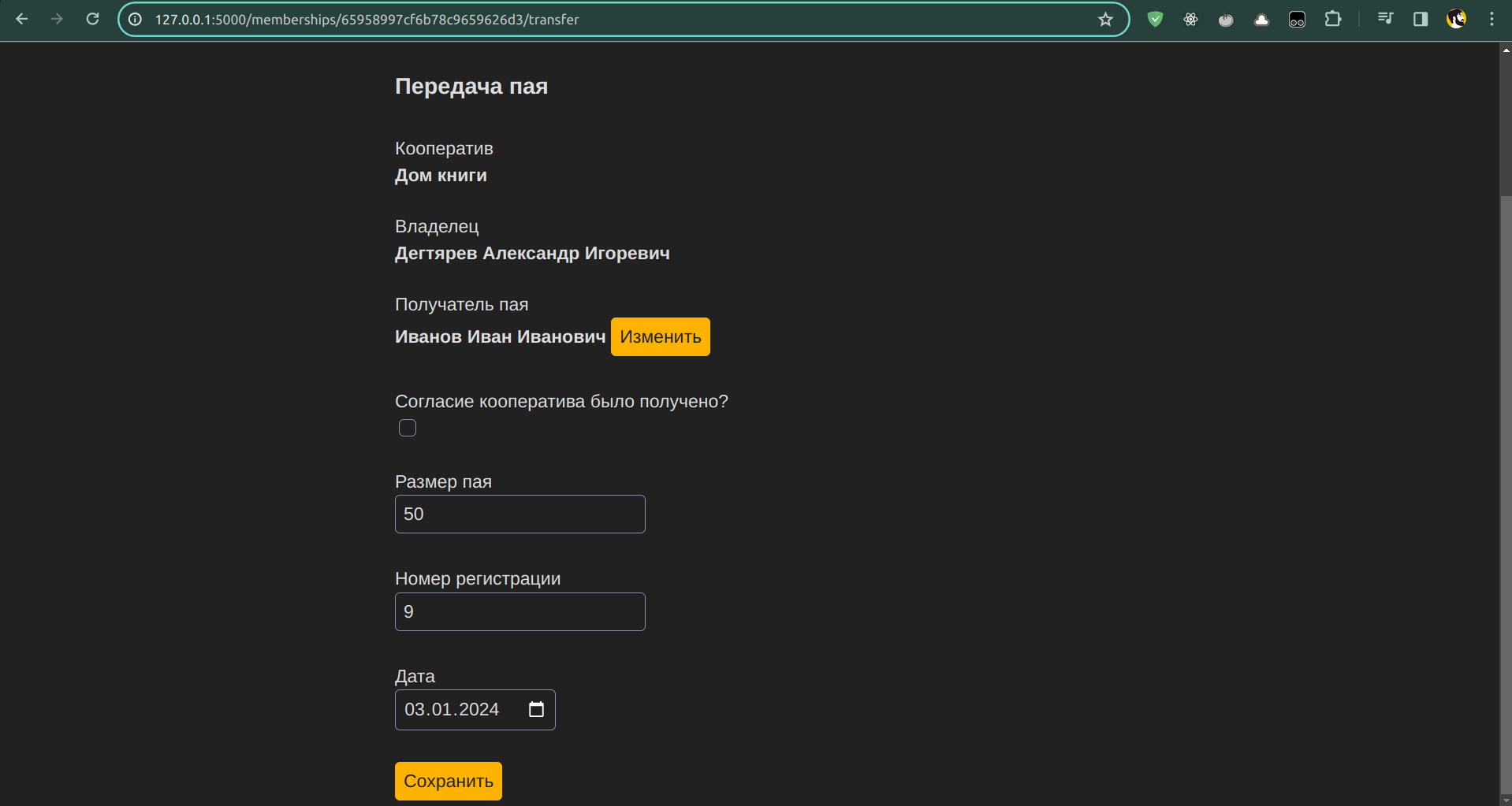
****

Рисунок 5.10 – Интерфейс передачи пая

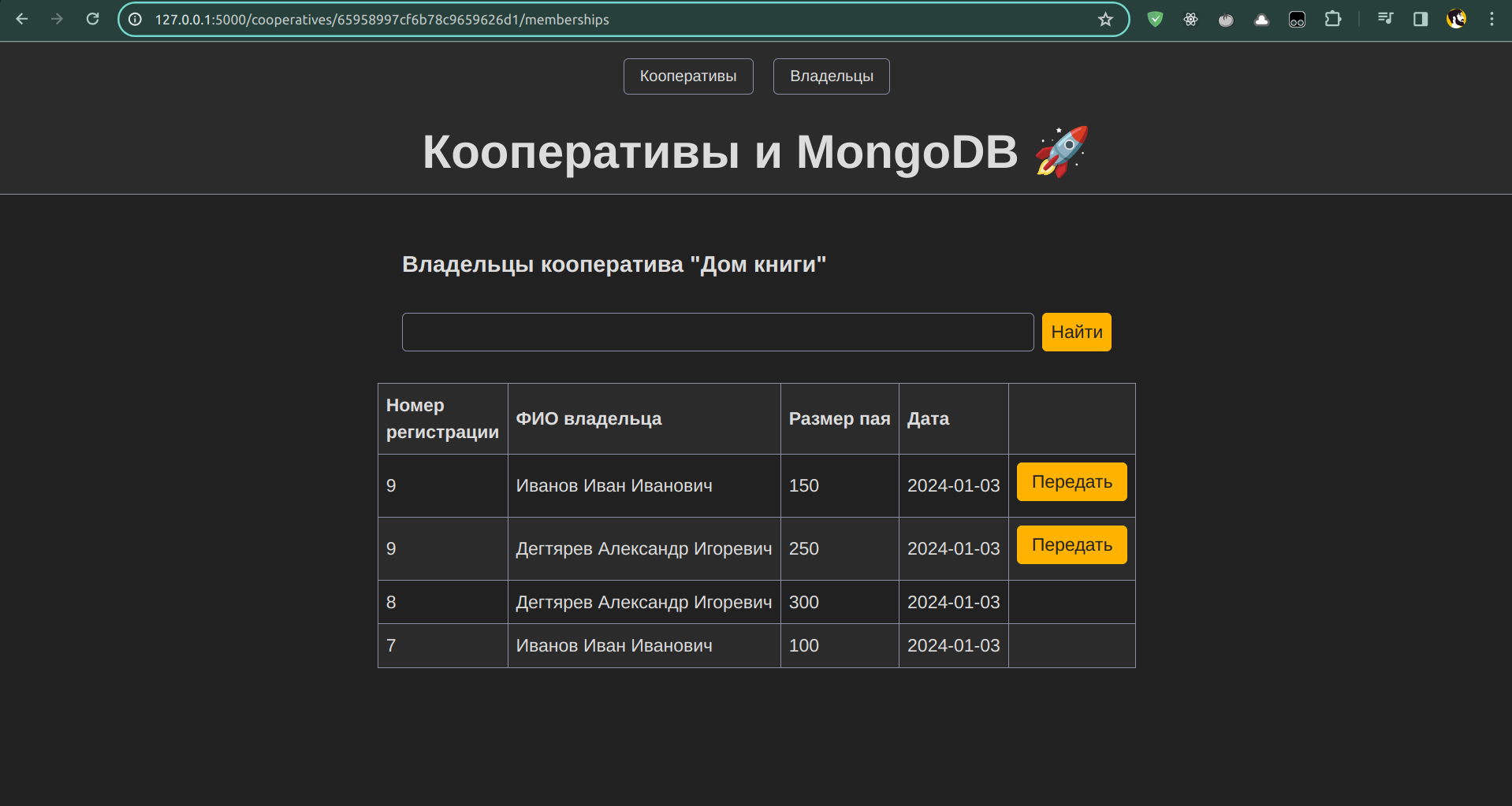
****

Рисунок 5.11 – Результат передачи части пая

Листинг 5.1 – Состояние коллекций после указанных шагов

| lab02> db.cooperatives.find() [  {  \_id: ObjectId("659587844dd874d5dac5ddfe"),  name: 'Мир овощей',  residence: 'Россия',  workers\_number: 11,  capital\_amount: 101,  profile: 'Продукты'  },  {  \_id: ObjectId("659587844dd874d5dac5ddff"),  name: 'Мир пуговиц',  residence: 'Россия',  workers\_number: 22,  capital\_amount: 304,  profile: 'Галантерея'  },  {  \_id: ObjectId("659587844dd874d5dac5de00"),  name: 'Мир ручек',  residence: 'Россия',  workers\_number: 33,  capital\_amount: 609,  profile: 'Канцелярия'  },  {  \_id: ObjectId("65958997cf6b78c9659626d1"),  name: 'Дом книги',  residence: 'Россия',  workers\_number: 21,  profile: 'Публицистика',  capital\_amount: 400  } ] lab02> db.owners.find() [  {  \_id: ObjectId("659587844dd874d5dac5ddfb"),  name: 'Иванов Иван Иванович',  address: 'Январская улица, д 1',  residence: 'Россия',  passport: '1111222233334444'  },  {  \_id: ObjectId("659587844dd874d5dac5ddfc"),  name: 'Петухов Петр Петрович',  address: 'Январская улица, д 1',  residence: 'Россия',  passport: '1111222233334444'  },  {  \_id: ObjectId("659587844dd874d5dac5ddfd"),  name: 'Кузнецов Кирилл Кириллович',  address: 'Январская улица, д 1',  residence: 'Россия',  passport: '1111222233334444'  },  {  \_id: ObjectId("6595882dcf6b78c9659626c8"),  name: 'Дегтярев Александр Игоревич',  residence: 'Россия',  address: 'ул 1 мая',  passport: '123412341234'  } ] lab02> db.memberships.find() [  {  \_id: ObjectId("659587844dd874d5dac5de01"),  number: 1,  owner\_id: ObjectId("659587844dd874d5dac5ddfb"),  cooperative\_id: ObjectId("659587844dd874d5dac5ddfe"),  amount: 101,  date: '2024-01-01'  },  {  \_id: ObjectId("659587844dd874d5dac5de02"),  number: 2,  owner\_id: ObjectId("659587844dd874d5dac5ddfb"),  cooperative\_id: ObjectId("659587844dd874d5dac5ddff"),  amount: 102,  date: '2024-01-01'  },  {  \_id: ObjectId("659587844dd874d5dac5de03"),  number: 3,  owner\_id: ObjectId("659587844dd874d5dac5ddfc"),  cooperative\_id: ObjectId("659587844dd874d5dac5ddff"),  amount: 202,  date: '2024-01-01'  },  {  \_id: ObjectId("659587844dd874d5dac5de04"),  number: 4,  owner\_id: ObjectId("659587844dd874d5dac5ddfb"),  cooperative\_id: ObjectId("659587844dd874d5dac5de00"),  amount: 103,  date: '2024-01-01'  },  {  \_id: ObjectId("659587844dd874d5dac5de05"),  number: 5,  owner\_id: ObjectId("659587844dd874d5dac5ddfc"),  cooperative\_id: ObjectId("659587844dd874d5dac5de00"),  amount: 203,  date: '2024-01-01'  },  {  \_id: ObjectId("659587844dd874d5dac5de06"),  number: 6,  owner\_id: ObjectId("659587844dd874d5dac5ddfd"),  cooperative\_id: ObjectId("659587844dd874d5dac5de00"),  amount: 303,  date: '2024-01-01'  },  {  \_id: ObjectId("65958997cf6b78c9659626d2"),  number: 7,  owner\_id: ObjectId("659587844dd874d5dac5ddfb"),  amount: 100,  date: '2024-01-03',  cooperative\_id: ObjectId("65958997cf6b78c9659626d1")  },  {  \_id: ObjectId("65958997cf6b78c9659626d3"),  number: 8,  owner\_id: ObjectId("6595882dcf6b78c9659626c8"),  amount: 300,  date: '2024-01-03',  cooperative\_id: ObjectId("65958997cf6b78c9659626d1")  },  {  \_id: ObjectId("65958ac1cf6b78c9659626db"),  number: 9,  owner\_id: ObjectId("659587844dd874d5dac5ddfb"),  amount: 150,  date: '2024-01-03',  cooperative\_id: ObjectId("65958997cf6b78c9659626d1")  },  {  \_id: ObjectId("65958ac1cf6b78c9659626dc"),  number: 9,  owner\_id: ObjectId("6595882dcf6b78c9659626c8"),  amount: 250,  date: '2024-01-03',  cooperative\_id: ObjectId("65958997cf6b78c9659626d1")  } ] |
| --- |

**Заключение**

В рамках лабораторная был разработан веб сервер с пользовательским интерфейсом к ИС учета кооперативов и их владельцев с минимальным функционалом для реализации следующих бизнес-процессов: регистрация кооператива и передача пая. Хранение данных было реализовано в MongoDB, доступ к которой был осуществлен через пакет pymongo.