**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное**

**образовательное учреждение высшего образования**

**«Самарский государственный технический университет»**

**Институт автоматики и информационных технологий**

**Кафедра «Информационно – измерительная техника»**

**Пояснительная записка**

**к курсовой работе Разработка информационной системы для поддержки процесса эксплуатации оборудования КИПиА промышленного предприятия в сфере геофизики.**

**по курсу «Интегрированные технологические**

**системы в приборостроении»**

**Нормоконтроль\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.А. Тюрин**

**Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.В. Мельников**

**Студент Скляров Д. В.**

**Группа 3-ИАИТ-5**

**Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Срок выполнения\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Самара 2023 г.**

# РЕФЕРАТ

Курсовой проект содержит 40 страниц, 11 рисунков, 11 источников, 21 приложение.

CLICKHOUSE, DOCKER, DOCKER-COMPOSE, FASTAPI, HTTP, JWT, MONGO, NGINX, POSTGRESQL, PYDANTIC, PYTHON, SASS, SPA-ПРИЛОЖЕНИЕ, VUE, VUEX, БАЗА ДАННЫХ, КЛИЕНТ-СЕРВЕРНОЕ ВЗАИМОДЕЙСТВИЕ, ОДНОСТРАНИЧНОЕ ПРИЛОЖЕНИЕ

Цель курсовой работы: разработать информационную систему для поддержки процесса эксплуатации оборудования КИПиА промышленного предприятия в сфере геофизики**СОДЕРЖАНИЕ**

[РЕФЕРАТ 2](#_Toc137772689)

[ВВЕДЕНИЕ 5](#_Toc137772690)

[1 АНАЛИЗ ЖИЗНЕННОГО ЦИКЛА ПРИБОРОВ КИПИА НА ПРЕДПРИЯТИИ. 6](#_Toc137772691)

[2 РАЗРАБОТКА СТРУКТУРЫ БАЗЫ ДАННЫХ 8](#_Toc137772692)

[3 РАЗРАБОТКА НЕОБХОДИМЫХ ОТЧЕТОВ И ЭКРАННЫХ ФОРМ 10](#_Toc137772693)

[4 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ РАЗРАБОТАННОЙ СИСТЕМЫ НА НАБОРЕ ТЕСТОВЫХ ДАННЫХ 14](#_Toc137772694)

[ЗАКЛЮЧЕНИЕ 18](#_Toc137772695)

[СПИСОК ЛИТЕРАТУРЫ 19](#_Toc137772696)

[Приложение А Страница с логотипом 20](#_Toc137772697)

[Приложение Б Страница с вакансиями 21](#_Toc137772698)

[Приложение В Информация о компании 22](#_Toc137772699)

[Приложение Г Форма авторизации 23](#_Toc137772700)

[Приложение Д Форма авторизации с ошибкой 24](#_Toc137772701)

[Приложение Е Страница с меню 25](#_Toc137772702)

[Приложение Ё Отчёт со всеми устройствами 26](#_Toc137772703)

[Приложение Ж Отчёт с устройствами по категориям 27](#_Toc137772704)

[Приложение З Отчёт с устройствами по спецификациям 28](#_Toc137772705)

[Приложение И Отчёт с устройствами по текущим статусам 29](#_Toc137772706)

[Приложение К Отчёт с устройствами по местам установки 30](#_Toc137772707)

[Приложение Л Отчёт с устройствами по закреплённым за ними сотрудниками 31](#_Toc137772708)

[Приложение М Отчёт с полной информацией об устройстве 32](#_Toc137772709)

[Приложение Н Форма добавления устройства 33](#_Toc137772710)

[Приложение О Отчёт по плану поверок 34](#_Toc137772711)

[Приложение П Отчёт по приборам, со статусом назначенной поверки 35](#_Toc137772712)

[Приложение Р Отчёт о известных устройствах 36](#_Toc137772713)

[Приложение С Форма добавления нового прибора 37](#_Toc137772714)

[Приложение Т Отчёт о сотрудниках 38](#_Toc137772715)

[Приложение У Форма добавления нового сотрудника 39](#_Toc137772716)

[Приложение Ф Антиплагиат 20](#_Toc137772717)

# ВВЕДЕНИЕ

КИП автоматика — это контрольно-измерительные приборы, которые обеспечивают работоспособность различных систем в быту и на промышленных предприятиях. С их помощью автоматически контролируется энергопотребление, проводится измерение величин и осуществляются другие операции. Эффективное ведение технологических процессов заключается не только в контроле, но и в регистрации и регулировке, которые также выполняются контрольно-измерительными приборами и автоматикой.

В процессе эксплуатации данных устройств следует понимать, что они имеют свой жизненный цикл со своими этапами, за которыми стоит следить, для чего как раз и создаются автоматизированные системы контроля устройств КИПиА. Общий подход при создании в таких систем заключается в выделении общих, статусов устройств КИПиА вне зависимости от сфере внедрения устройства, чтобы система имела минимально жизнеспособный вид на этапе внедрения, помимо этого система должна обладать имеет гибким функционалом настройки, благодаря чему внедрение может быть осуществлено в любой сфере.

На данный момент рынок систем ведения поддержки процесса эксплуатации устройств КИПиА не представлен ни в каком виде, в виду низкой ликвидности и специфичности рынка. Если заказчику требуется создать такую систему, то она изначально создаётся под ограниченный набор задач, непосредственно под специфику приборов, с которыми работает заказчик.

Цель курсовой работы является разработка удобной, с точки зрения «развёртки», поддержания и использования, информационной системы для поддержки процесса эксплуатации оборудования КИПиА промышленного предприятия.

# 1 АНАЛИЗ ЖИЗНЕННОГО ЦИКЛА ПРИБОРОВ КИПИА НА ПРЕДПРИЯТИИ.

Каждое устройство в ходе эксплуатации должно пройти через следующие этапы жизненного цикла (в дальнейшем единица жизненного цикла устройства называется – событием):

* Зарегистрирован – обязательное событие с момента которого можно считать, что устройство поставлено на учёт, но также стоит отметить, что данное событие не означает единовременное введение в работу, следовательно учитывать такие события при расчётах амортизации (или подобных характеристиках) не стоит;
* Установлен – обязательное событие, от которого должны производиться расчёты, так как фактически, от этого этапа события устройство будет эксплуатироваться, следовательно его ресурс и смежные характеристики будут снижаться;
* Поверен – каждое измерительное устройство раз в определённый промежуток времени должно проходить сертификации на фактическое соответствие своим характеристикам, таким образом это обязательное событие. Это событие должно обладать переходной точкой для расчётов, от которой будет рассчитывать дата следующей поверки. Также стоит обратить внимание на проблему человеческого фактора в данном вопросе, ответственный за КИПиА попросту может забыть о поверке устройства, в таком случае сам сотрудник и вышестоящее руководство должно выделять такие устройства в приоритетную группу на поверку, таким образом точно все устройства будут поверены.
* Поставлен на поверку – обязательное событие – означает, что человек ответственный поставил устройство КИПиА в приоритет на поверку;
* Снят с эксплуатации – обязательное событие, каждое устройство имеет свой ресурс работы, по истечению которого, его необходимо заменить новым устройством;
* Починен – необязательное событие, но его также стоит учитывать, чтобы иметь более детальную информацию. Благодаря данному статусу теоретически можно провести статистический анализ, с целью выявления некачественных КИПиА.

# 2 РАЗРАБОТКА СТРУКТУРЫ БАЗЫ ДАННЫХ

В ходе работы предстоит ведение различной по структуре хранения и типу информации. Также стоит обратить внимание, что определённого рода структуры будут задействованы в пользовательской работе чаще, чем остальные, к примеру – информация о сотрудниках будет менее важна в данной системе, чем история статусов КИПиА.

Таким образом, для данных о приборах, таких как – спецификация и описание – была выбрана NoSQL база данных – MongoDB – она предоставляет хранение данных в формате JSON ключ-значение, где нет закреплённой структуры, в итоге – информация о спецификации может иметь сколько угодно значение, а описание хранить как обычные строки, так и пред форматированный HTML текст. Сама база данных имеет асинхронный интерфейс, при помощи которого взаимодействовать с базой данных может более одного человека в один промежуток времени.

Информация о сотрудниках, поставленных на учёт устройствах и набора заранее созданных устройств имеют изначально определённую структуру – следовательно для них наиболее удобоваримым форматом будет SQL ориентированная база данных – PostgreSQL. Такого рода база данных позволит создать связи между учтёнными устройствами и закреплёнными за ними сотрудниками при помощи «многие ко многим» и внешних ключей. Помимо этого, реализация PostgreSQL обладает асинхронным интерфейсом и транзакциями, при помощи них можно быть уверенным в консистентности данных при разнородном взаимодействии большого количества одновременно взаимодействующих пользователей.

Информация о событиях, применяемых в системе, обладает наиболее весомым с количественной точки зрения размером. На одно устройство приходится минимум пять событий, это, не говоря о пользовательских событиях, сами же данные имеют определённую структуру – уникальный идентификатор устройства, идентификатор закреплённого за ним человека и значение статуса события. Помимо этого, данные о событиях нужно собирать как можно скорее, чтобы не ухудшить пользовательское взаимодействие. Наиболее подходящей для таких требований является колоночная база данных – Clickhouse от Яндекса. Поиск в ней происходит поколоночно, а также доступно асинхронное взаимодействие, благодаря чему вносить и получать данные из неё можно быстрее, в сравнении с конкурентами.

Структура базы данных представлена на рисунке 2.1.

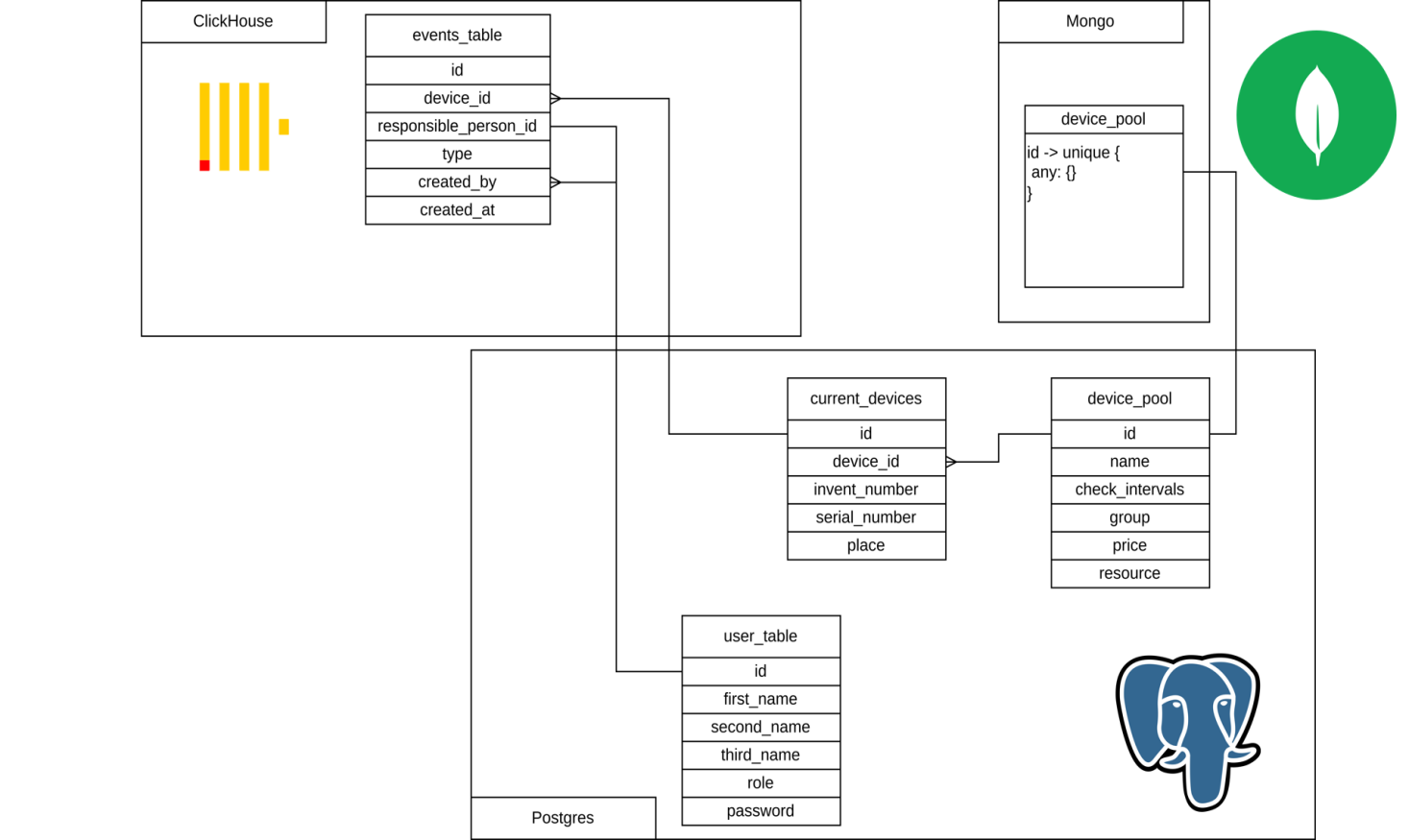


Рисунок 2.1 – Структура базы данных

Также стоит пояснить, что связи таблиц между разными базами данных реализованы на уровне серверного приложения, так что сами базы данных не знают о существовании друг друга, все проверки происходят на уровне контракта самого приложения.

# 3 РАЗРАБОТКА НЕОБХОДИМЫХ ОТЧЕТОВ И ЭКРАННЫХ ФОРМ

Изначально для отрисовки форм создано серверное приложение на языке программирования Python с использованием FastAPI, данные инструменты позволяют создавать набор контрактов, которые предоставляет приложение. Помимо FastAPI используются технологии связывающие объектно-ориентированные модели в языке Python с моделями в Clickhouse, PostgreSQL и MongoDB. В качестве библиотеки для работы с PostgreSQL – был выбран Tortoise, относительно новый инструмент, уступающий лишь синхронной Pony ORM, использующий свой собственный язык запросов, а не классический SQL. Для Clickhouse используется стандартный инструмент – SQLAlchemy, который предоставляет асинхронное взаимодействие при правильно настроенной конфигурации. MongoDB обладает инструментом Motor, который поддерживает сама компания Mongo, и она также создана для асинхронной работы.

Помимо инструментов для работы с базами данных, используются инструменты для их версионирования. Для сущностей в PostgreSQL используется Aerich разработка той же команды, что создаёт Tortoise ORM, а для Clickhouse – Alembic, хоть он не создаёт таблицы в зависимости от их типа, выставленном в коде, но правильно корректирует таблицы в соответствии с файлом миграции.

В ходе разработки серверного приложения были использованы инструменты автоматического тестирования, так как контрактов много и все они асинхронны – ручная отладка приложения здесь заняла бы значительную часть времени, потому в качестве инструмента тестирования был использован pytest с асинхронными плагинами, весь код серверного приложения покрыт авто тестами, благодаря которым можно сразу выявить – какой контракт, после изменения кода, отработал неправильно.

Помимо инструментов авто тестирования был использован инструмент линтирования – flake8 – при помощи него производится проверка на стандарты написания PEP 8, за исключением - E203, W503, W504, F405 – так как они противоречат стандартам использования набора инструментов FastAPI.

Одной из важной задач в ходе этого курсового проекта было – простая «развёртка» приложения, под которой понимается весь процесс настройки и запуска создаваемого приложения на реальном сервере. Потому в данном проекте решено было использовать Docker вместе с Docker-compose. Сами приложения баз данных, само контрактное приложение, а также вспомогательные инструменты «упакованы» в «контейнеры», свободно распространяемые на площадках DockerHub и DockerRegistry. При помощи всего одной консольной команды данные приложения будут выгружены с площадок, настроены в соответствии с файлом docker-compose и в итоге не будут ничем отличатся от приложений установленных в нативном формате. Так помимо всего прочего, вышеупомянутые инструменты предоставляют сетевое взаимодействие в локальном пространстве имён, из-за чего конфигурирование сетевых настроек не требуется.

Говоря о дополнительных инструментах, само приложение можно запустить как обычную Python программу, но, хоть оно и будет предоставлять асинхронный доступ (возможно даже для нескольких клиентов), оно не будет использовать все мощности на располагаемом сервере. Данную проблему решает связка двух приложений – Nginx и Gunicorn. Nginx – выполняет балансировку нагрузки между клиентами, а также их обратное проксирование, в это же время Gunicorn – исполняет один и тот же код на всех доступных ядрах процессора, экспонируя мульти поточный порт для Nginx.

Для упрощения разработки внешний визуальной части приложения был использован инструмент OpenAPI – он автоматически генерирует документацию к проекту, в соответствии с каждым реализуемым контрактом, указывая какие входные, выходные параметры, а также коды ошибок с их значениями. Кроме того, данная спецификация свободно конвертируется в коллекции Postman, благодаря чему можно проводить нагрузочное тестирование на пред-финальных этапах запуска на основном вычислительном сервере.

При разработки пользовательской части – принято решение о создании одностраничного приложения на языке программирования Vue, представляющем возможность разрабатывать элементы в качестве компонентов, а также их пере использовать.

Если пользователь не авторизован сайт делится на две части – основная информационная часть(приложение В), где представлен список вакансий (приложение Б), логотип компании со слоганом и информация о компании (приложение А), второй частью – является страница с авторизацией (приложение Г), которая может оповестить пользователя о неправильно введённых данных (приложение Д) – из соображений безопасности, если любые из данных неверные – то на страницу вернётся информация о том, что пользователь не найден, даже если такой есть в базе данных сотрудников.

После успешного входа пользователь будет перенаправлен на страницу с учётной информацией. Также при успешной авторизации уникальный JWT код, выданный сервером, будет сохранён в куки файлах используемого браузера, и при повторном открытии сайта, уже не нужно будет вводить данные – они сохраняется на протяжении 7 суток, а при повторном заходе пере генерируются в соответствии с контрактом серверного приложения и опять сохранятся.

Страница с учётными данными (приложение Е) имеет меню с четырьмя кнопками – «Приборы» - они же приборы на учёте, «БД Приборов» – это набор известных устройств, а также Поверка и Сотрудники – кнопки, отвечающие за соответствующие категории.

На странице с «приборами» можно выбрать создание отчёта по всем приборам (приложение Ё), либо фильтрам с настраиваемыми суб-фильтрами (приложение Ж-Л).

По нажатию на карточку с устройством – можно посмотреть полную информацию о нём (приложение М), а также выставить статус

Нажатие на кнопку «Добавить устройство» - откроет форму, где можно выбрать устройство из известных приборов (приложение Н), закрепить человека за ним, выставить серийный и инвентаризационный номера, а также указать место установки. По умолчанию данному прибору выдастся статус – Зарегистрирован.

Следующий пункт меню – отвечает за поверку – в нём можно посмотреть план поверок от ближайших по дате (приложение О), а также посмотреть устройствам, которым назначена поверка (приложение П).

Пункт меню, отвечающий за «БД Приборов» - показывает информацию о наборе известных устройств (приложение Р), а кнопка добавить устройство – откроет форму, где можно добавить прибор в известные (приложение С).

Кнопка меню «Сотрудники» – откроет отчёт по всем зарегистрированным сотрудникам (приложение Т), а «Добавить сотрудника» – откроет форму, где можно зарегистрировать нового человека в систему (приложение У).

# 4 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ РАЗРАБОТАННОЙ СИСТЕМЫ НА НАБОРЕ ТЕСТОВЫХ ДАННЫХ

Тестирование проводим по следующему алгоритму тестирования: добавим сотрудника, добавим новый прибор в известные, поставим этот прибор на учёт, проверим его статус, далее поменяем его статус на «поверен» и опять проверим этот прибор

1. Добавляем сотрудника

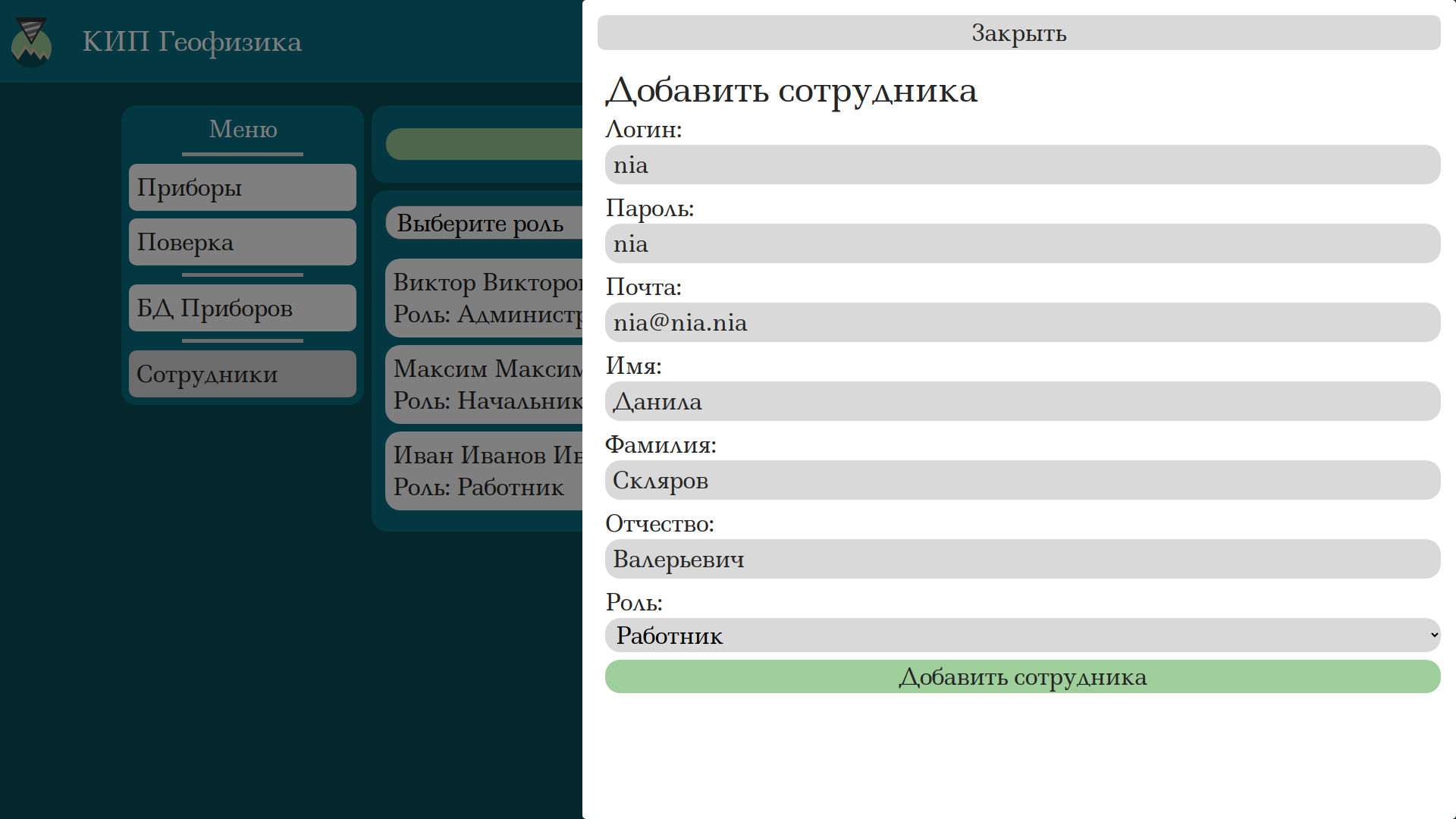


Рисунок 4.1 – Форма добавления сотрудника

1. Получаем ответ

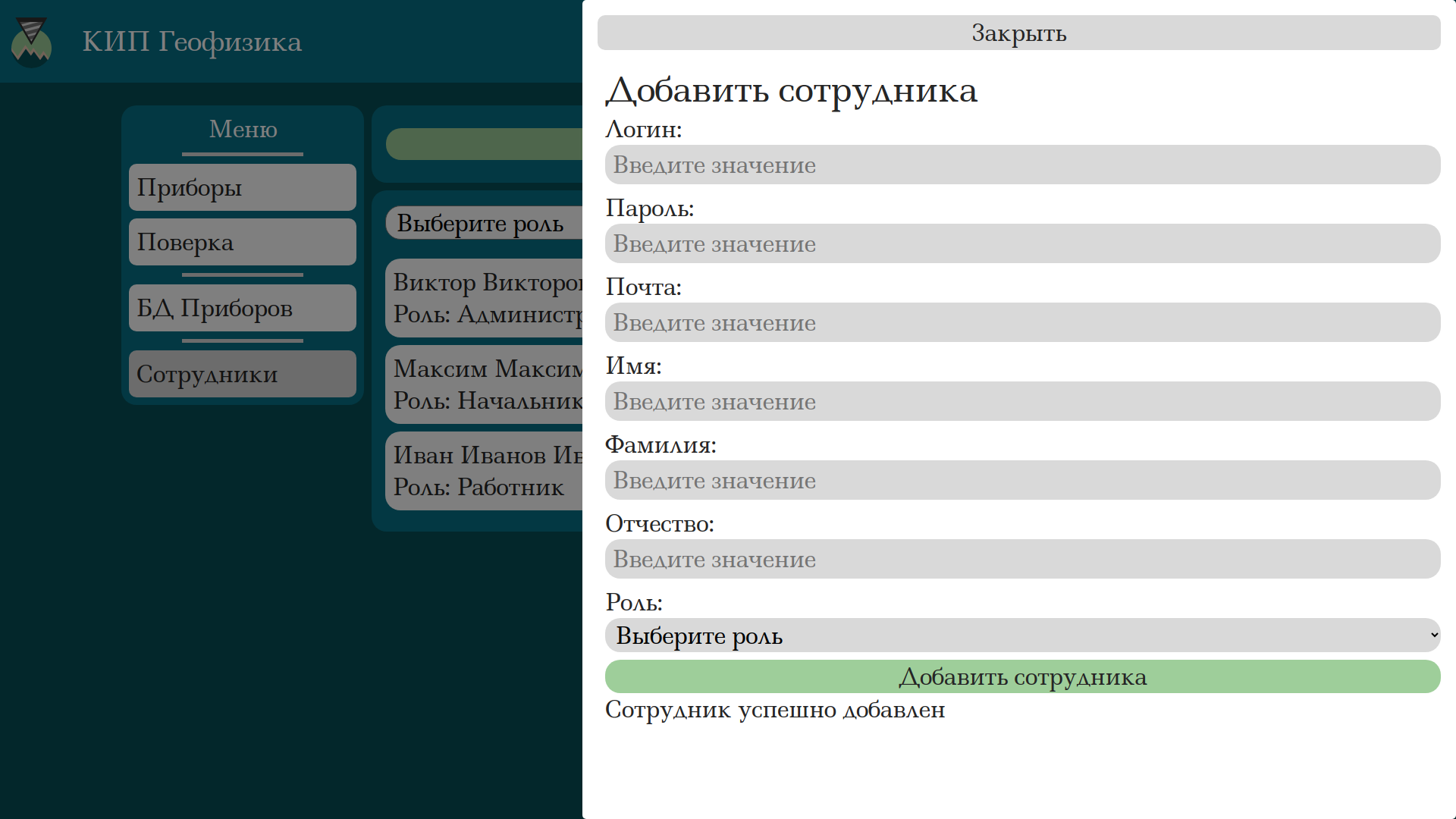


Рисунок 4.2 – Ответ на добавление сотрудника

1. Добавляем прибор

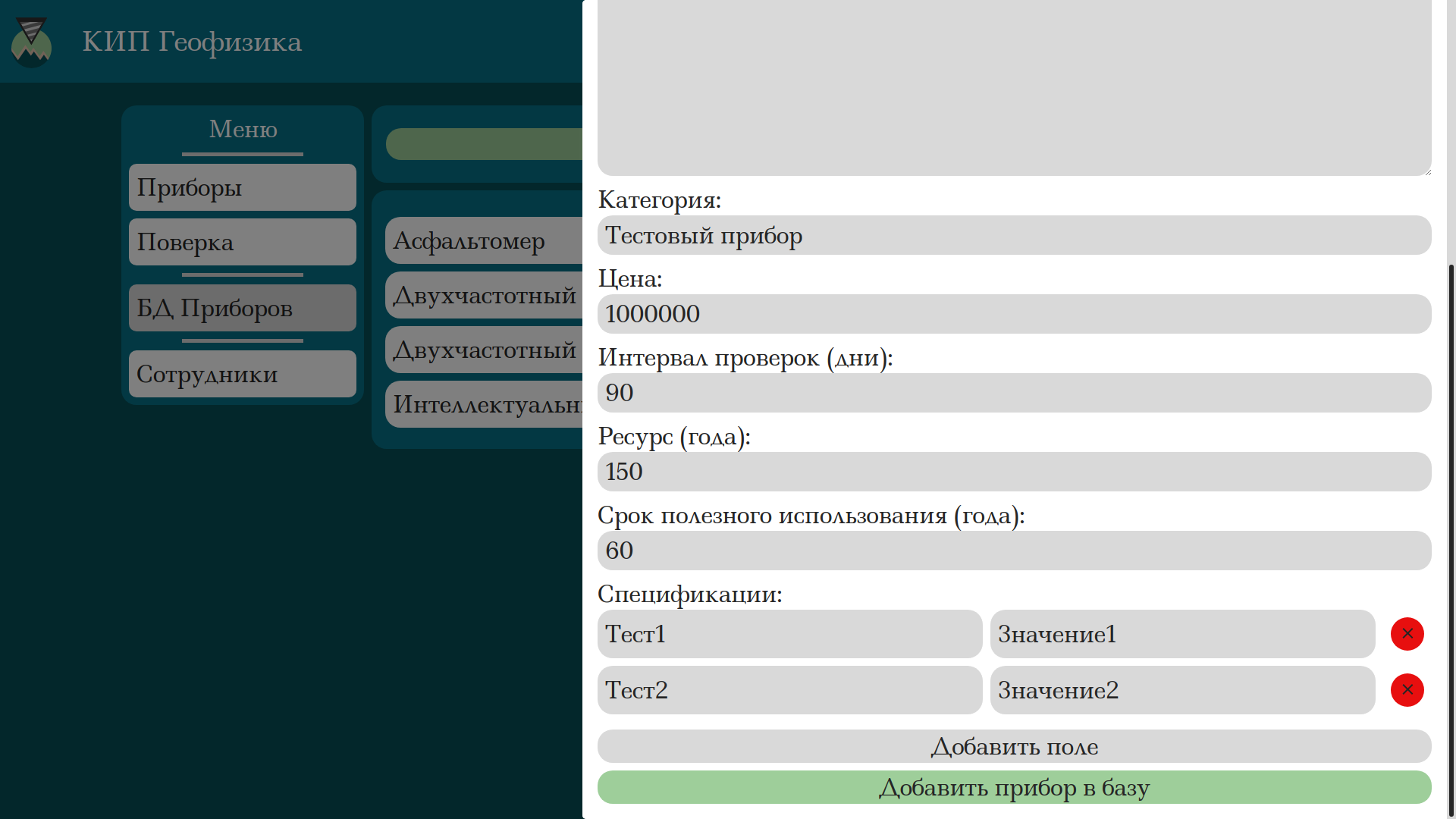


Рисунок 4.3 – Форма добавления прибора в базу приборов

1. Получаем ответ



Рисунок 4.4 – Ответ на добавление прибора в базу

1. Проверяем наличие прибора в базе приборов

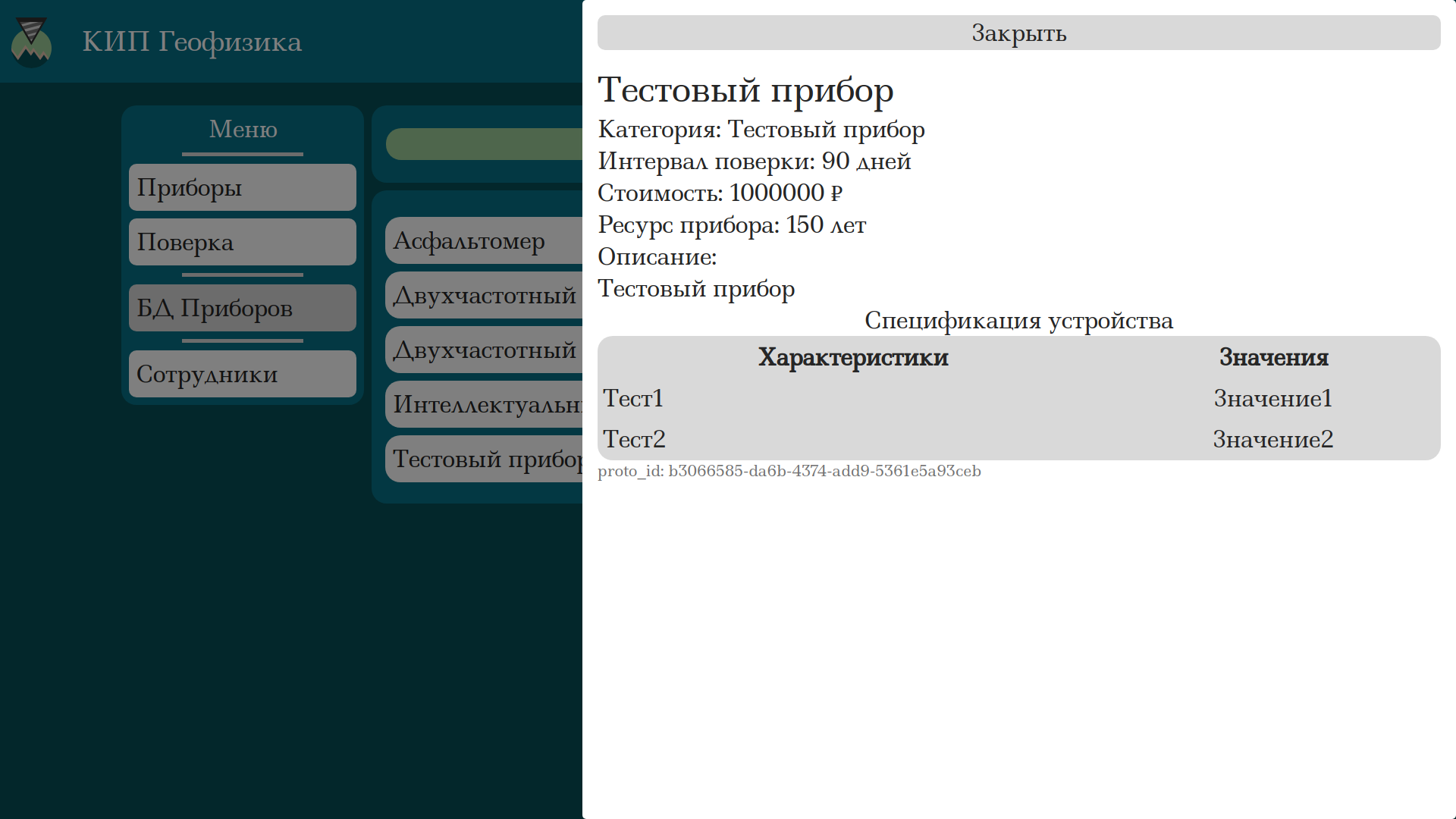


Рисунок 4.5 – Проверка наличия тестового добавленного прибора в базу

1. Ставим устройство на учёт

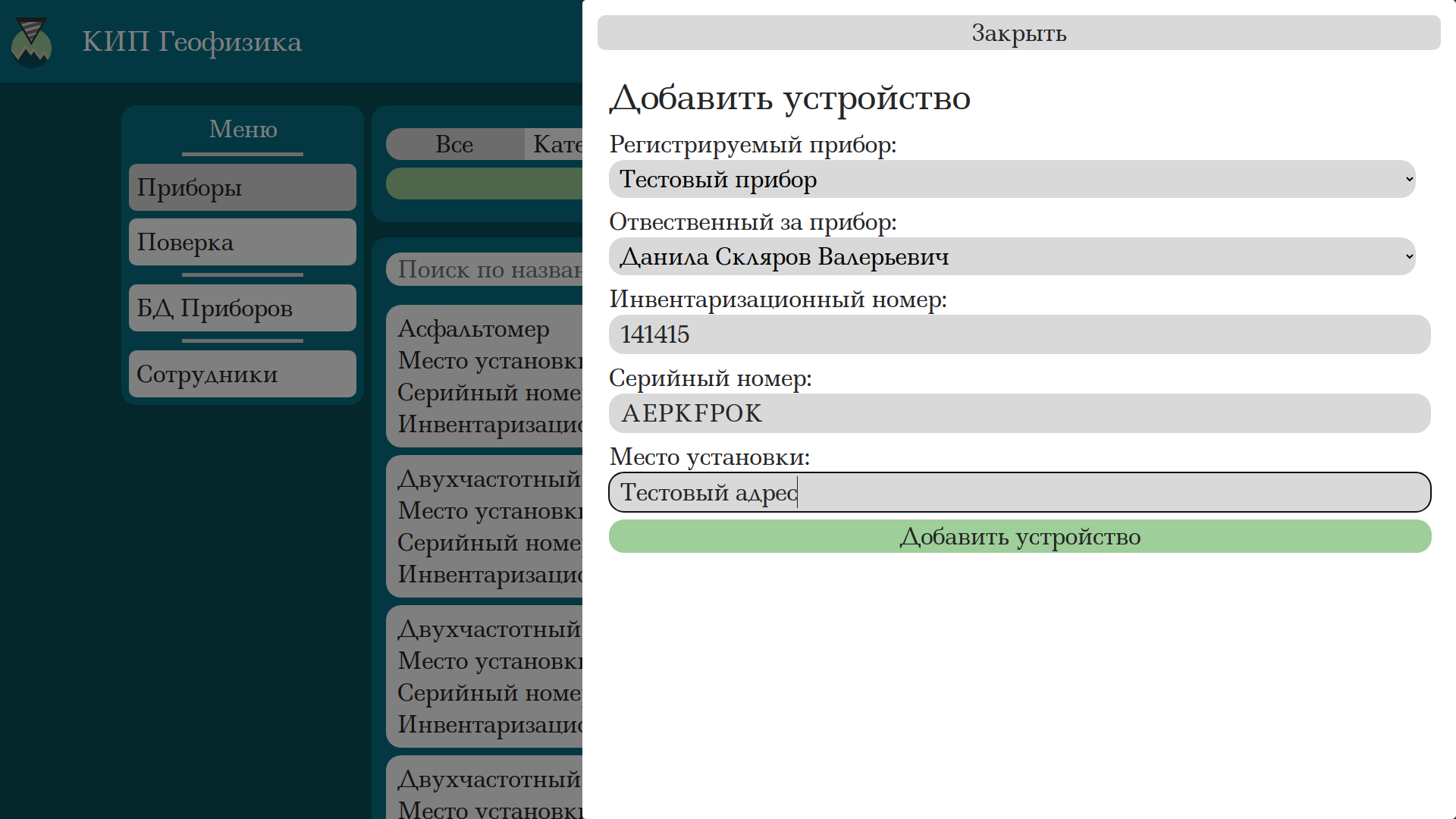


Рисунок 4.6 – Форма постановки прибора на учёт

1. Получаем ответ

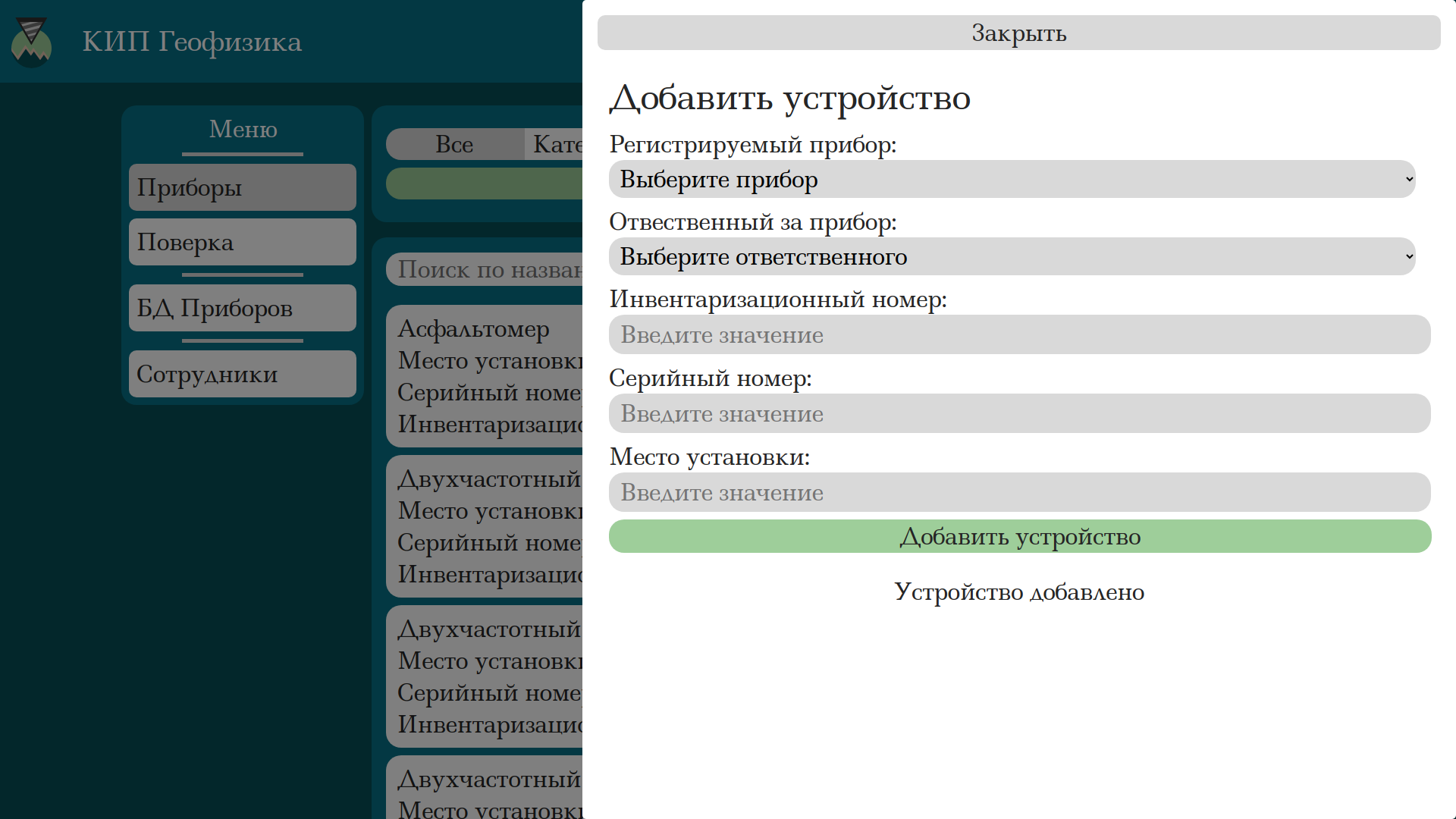


Рисунок 4.7 – Ответ на добавление устройства

1. Указываем статус на поверку

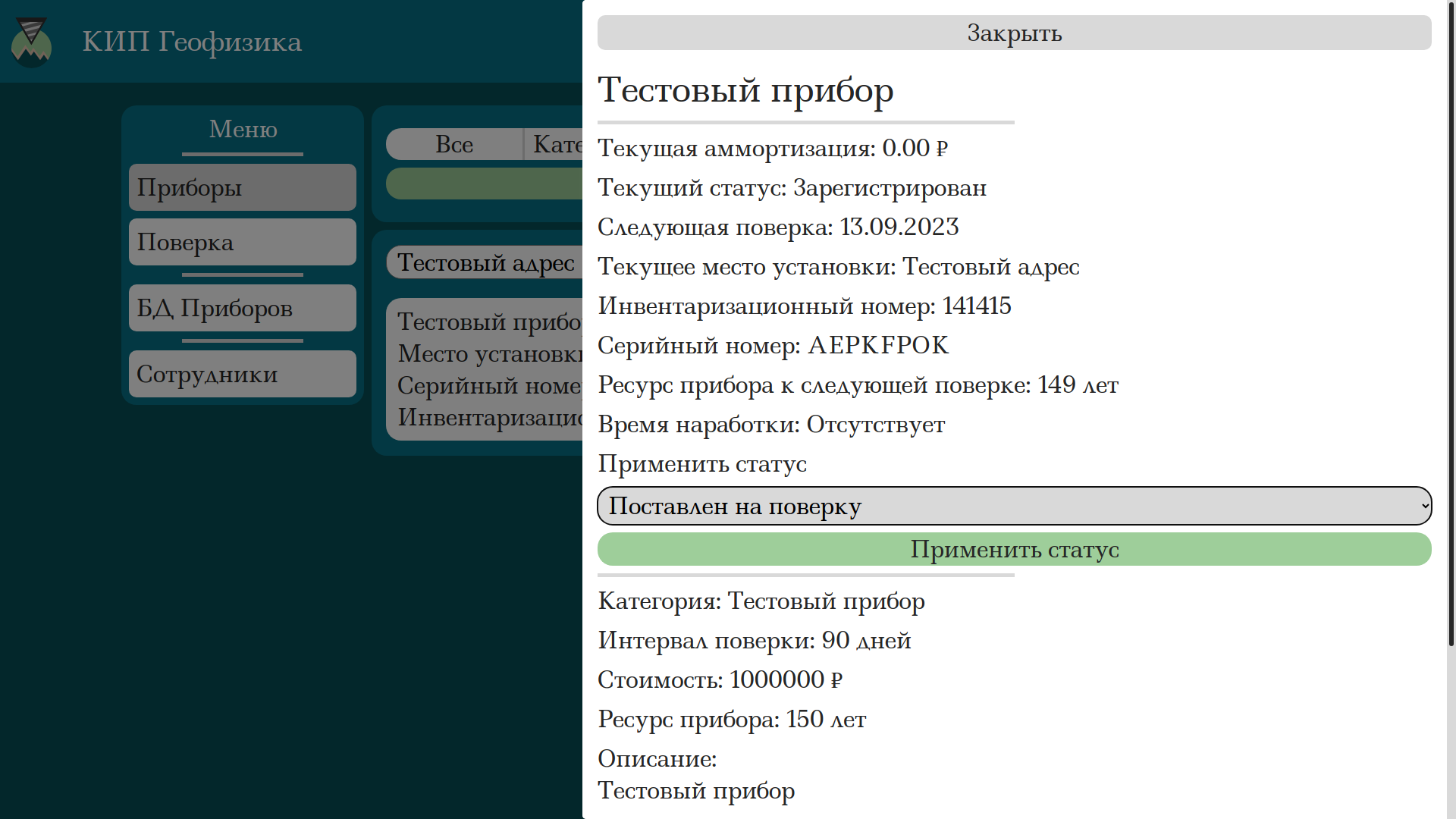


Рисунок 4.8 – Изменение статуса прибора

1. Получаем ответ

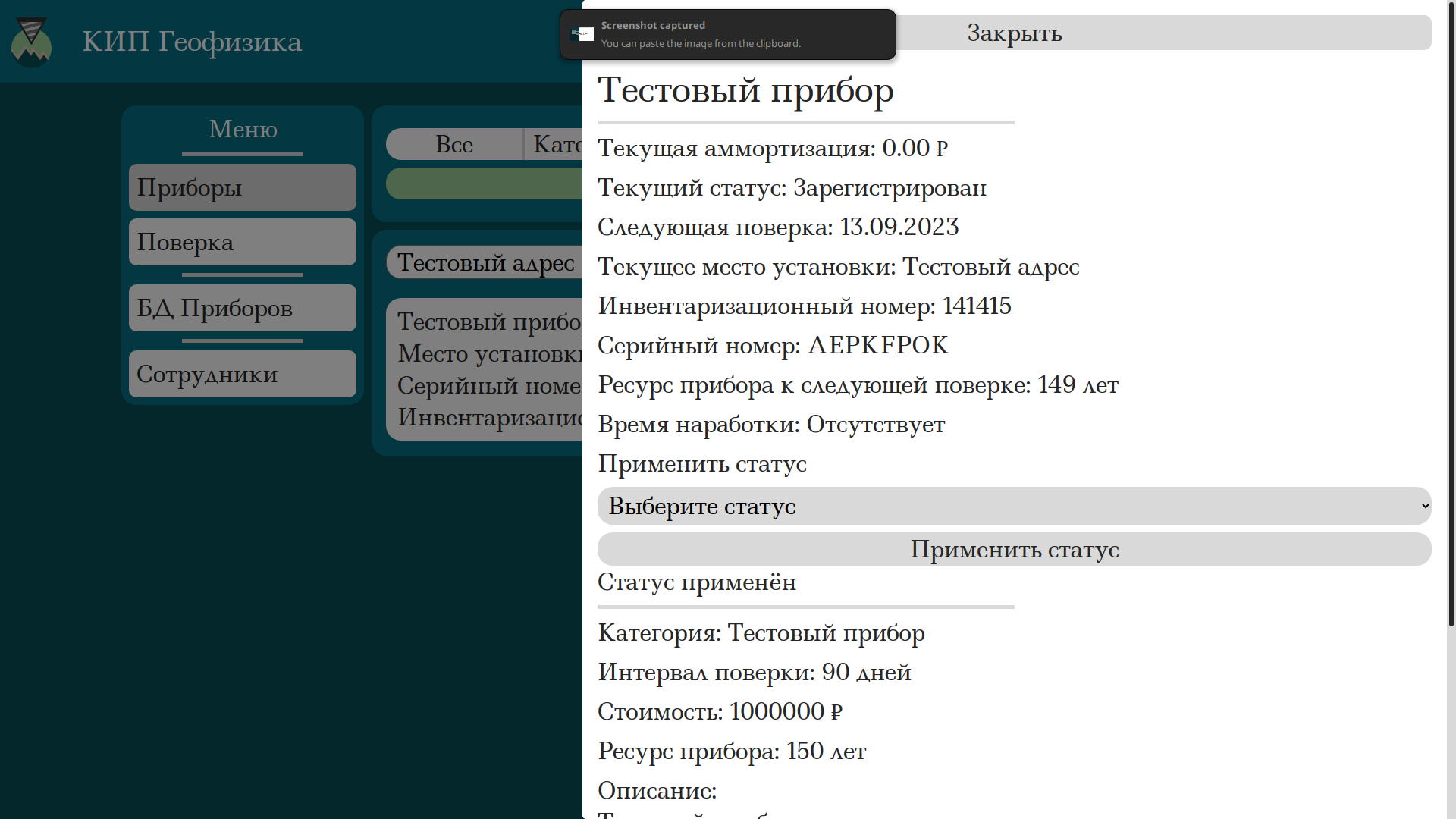


Рисунок 4.9 – Ответ на изменение статуса

1. Проверяем наличие прибора в поверенных

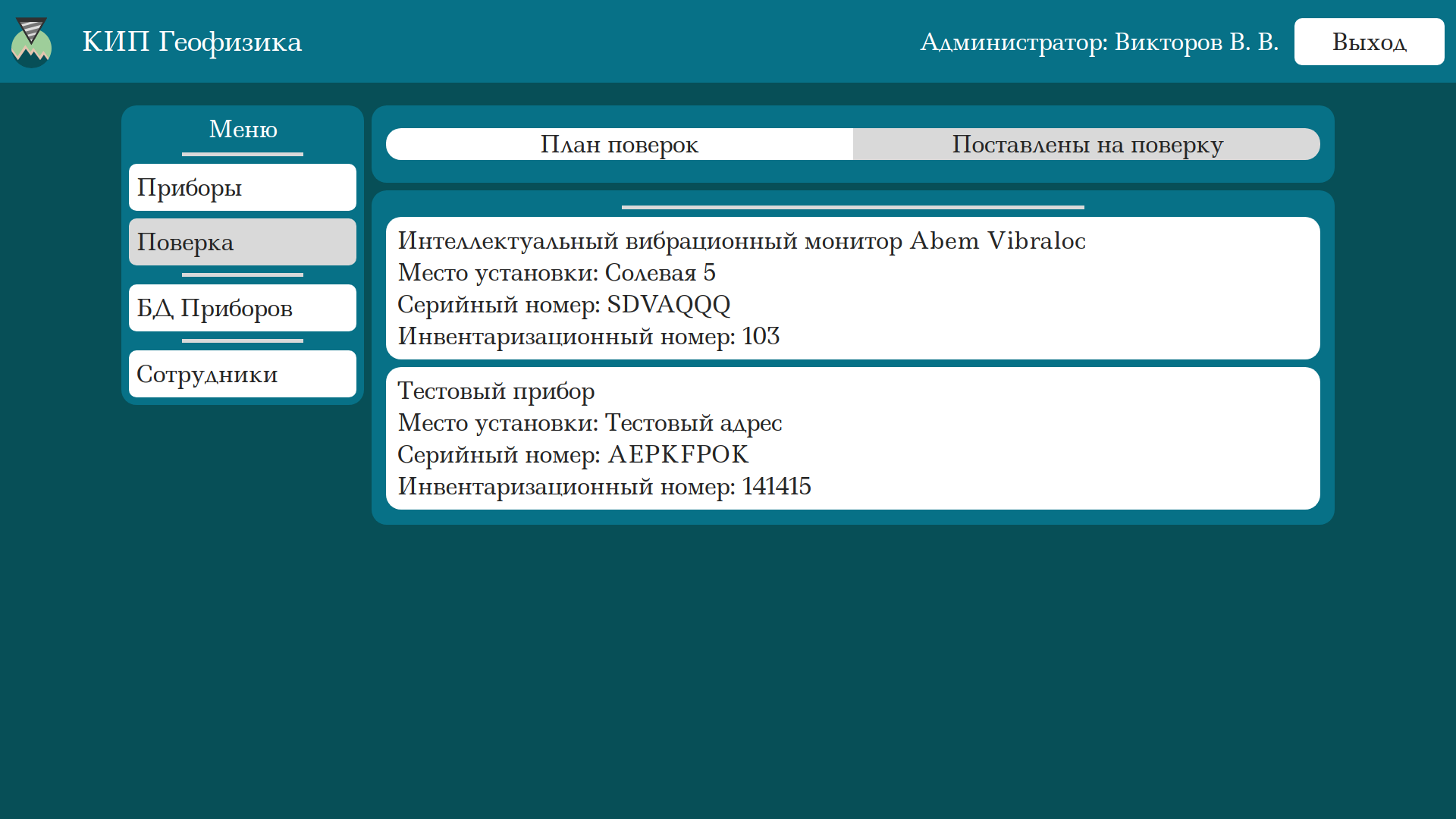


Рисунок 4.10 – Проверка прибора в поставленных на учёт

Как можно убедиться из вышеперечисленного, данная система оттестирована и полностью работает без ошибок.

# ЗАКЛЮЧЕНИЕ

В основе полученного задания была выполнена работа по разработке информационной системы для поддержки процесса эксплуатации оборудования КИПиА промышленного предприятия в сфере геофизики. Данная система гибко настраивается в соответствии с желаниями заказчика, легко запускается и также просто поддерживается. Исходный код программы представлен на сервисе GitHub по ссылке: <https://github.com/Niatomi/kip-system>

# СПИСОК ЛИТЕРАТУРЫ

1. Start with Pytest. [Электронный ресурс]; URL: <https://docs.pytest.org/en/7.3.x/>
2. FastAPI is a modern, fast (high-performance), web framework for building APIs. [Электронный ресурс]; URL: <https://fastapi.tiangolo.com/>
3. The Progressive JavaScript Framework [Электронный ресурс]; URL: <https://vuejs.org/>
4. What is Vuex? [Электронный ресурс]; URL: <https://vuex.vuejs.org/>
5. Что такое ClickHouse. [Электронный ресурс]; URL: <https://clickhouse.com/docs/ru>
6. PostgreSQL: The World's Most Advanced Open Source Relational Database. [Электронный ресурс]; URL: <https://www.postgresql.org/>
7. MongoDb. The Developer Data Platform [Электронный ресурс]; URL: <https://www.mongodb.com/>
8. Docker Compose overview. [Электронный ресурс]; URL: <https://docs.docker.com/compose/>
9. Why was Tortoise ORM built? [Электронный ресурс]; URL: <https://tortoise.github.io/#why-was-tortoise-orm-built>
10. Motor: Asynchronous Python driver for MongoDB [Электронный ресурс]; URL: <https://motor.readthedocs.io/en/stable/>
11. Что такое Axios? [Электронный ресурс]; URL: <https://axios-http.com/ru/docs/intro>

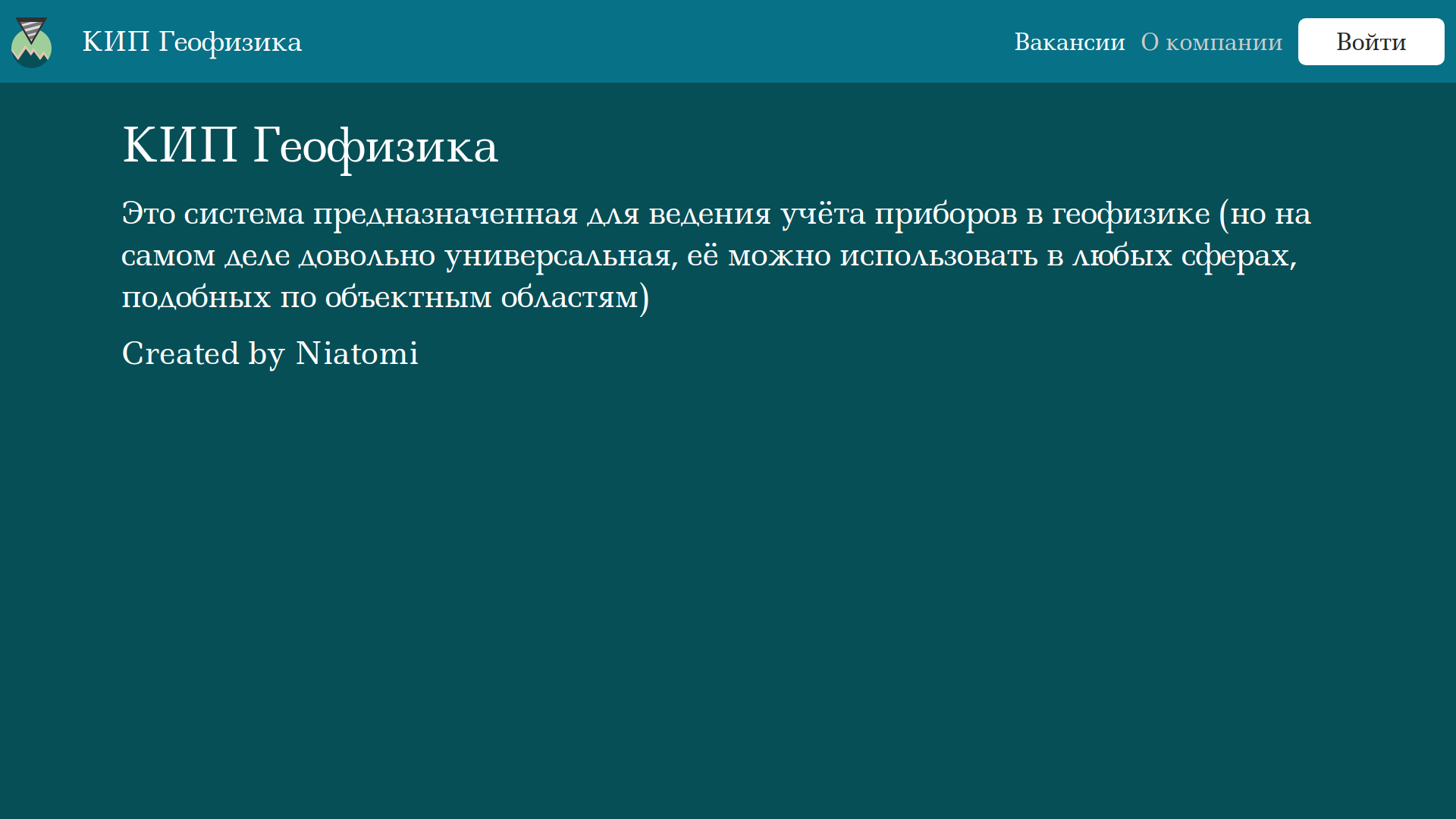
# Приложение А Страница с логотипом



# Приложение Б Страница с вакансиями



# Приложение В Информация о компании



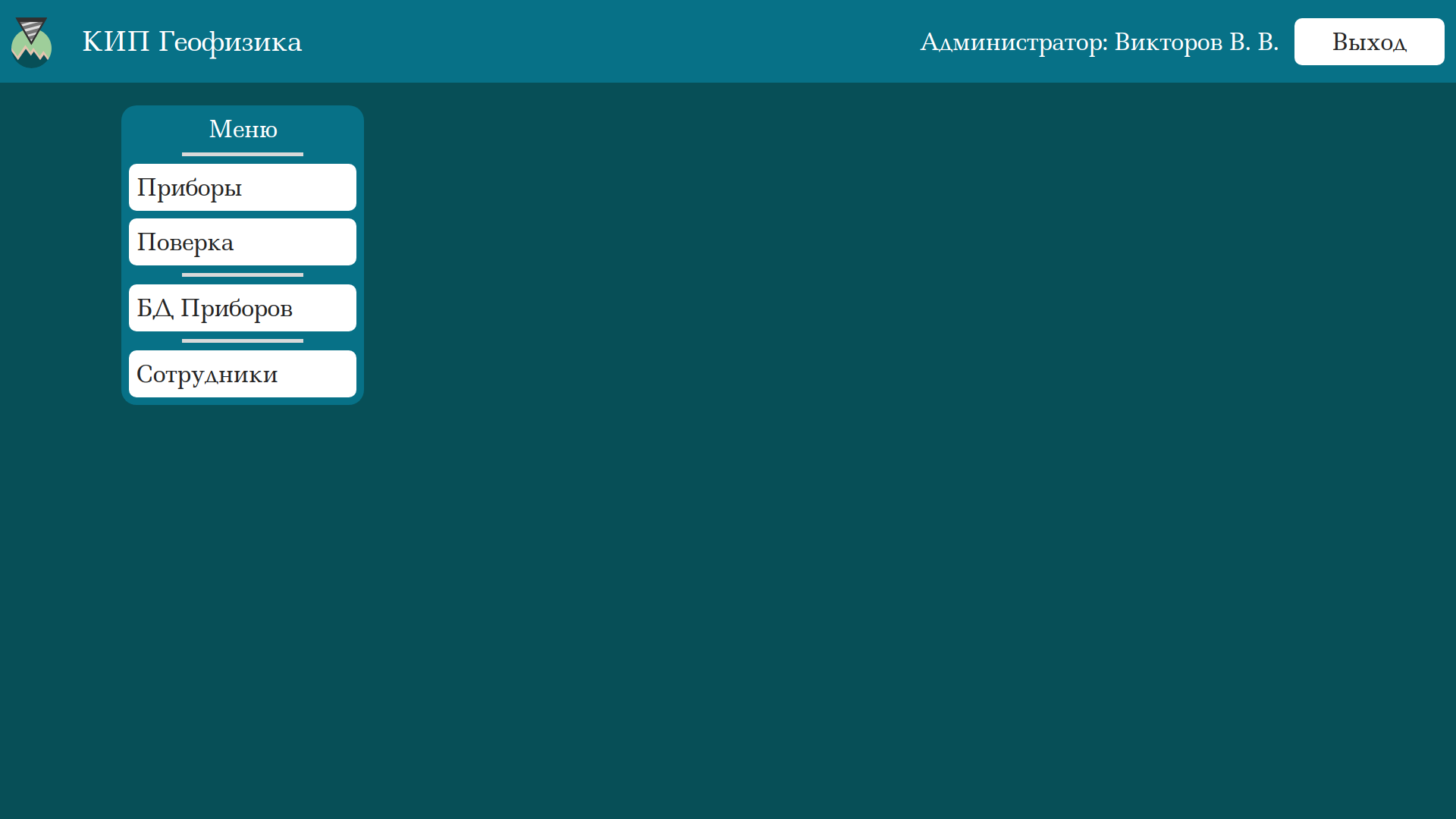
# Приложение Г Форма авторизации



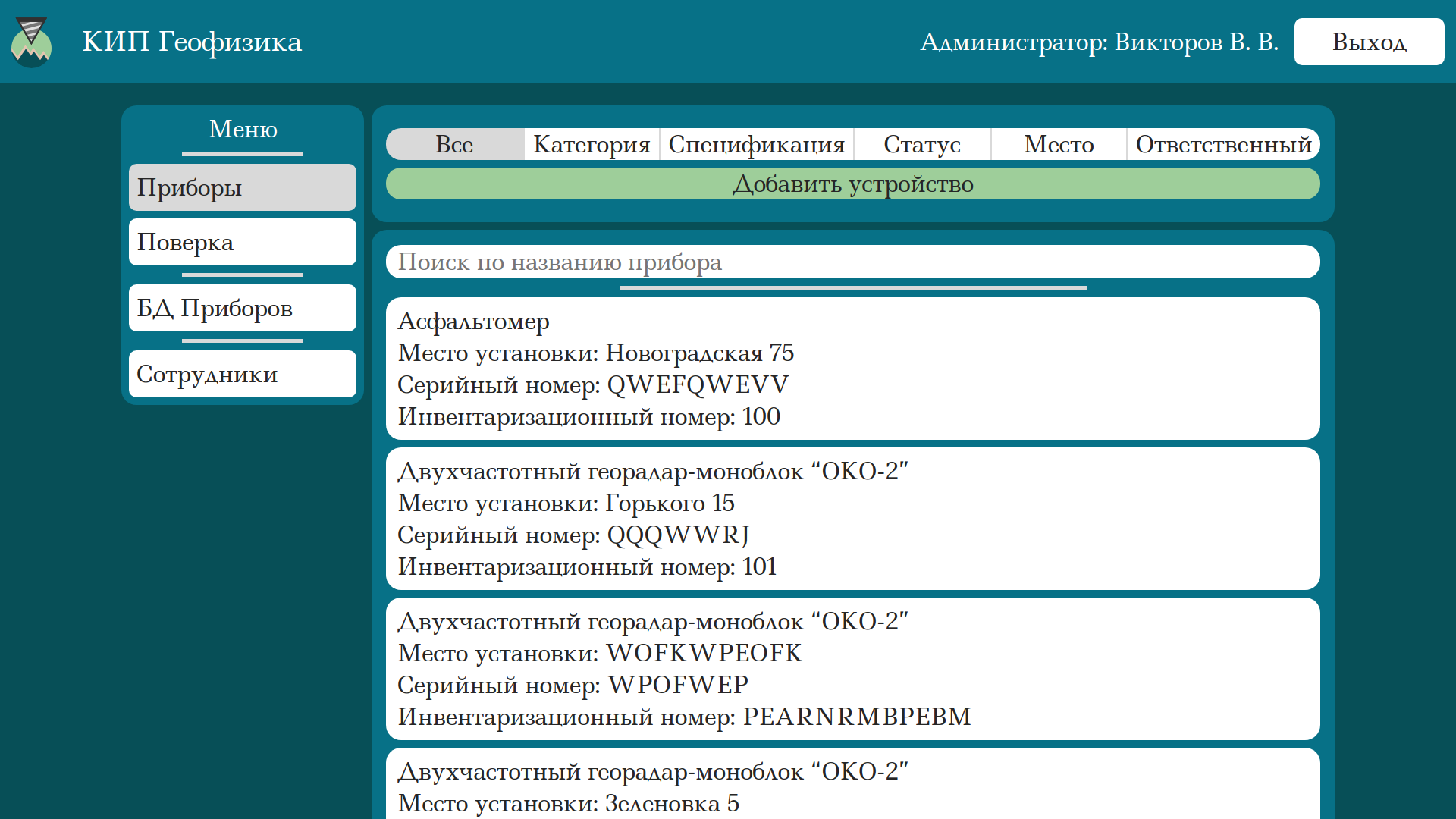
# Приложение Д Форма авторизации с ошибкой



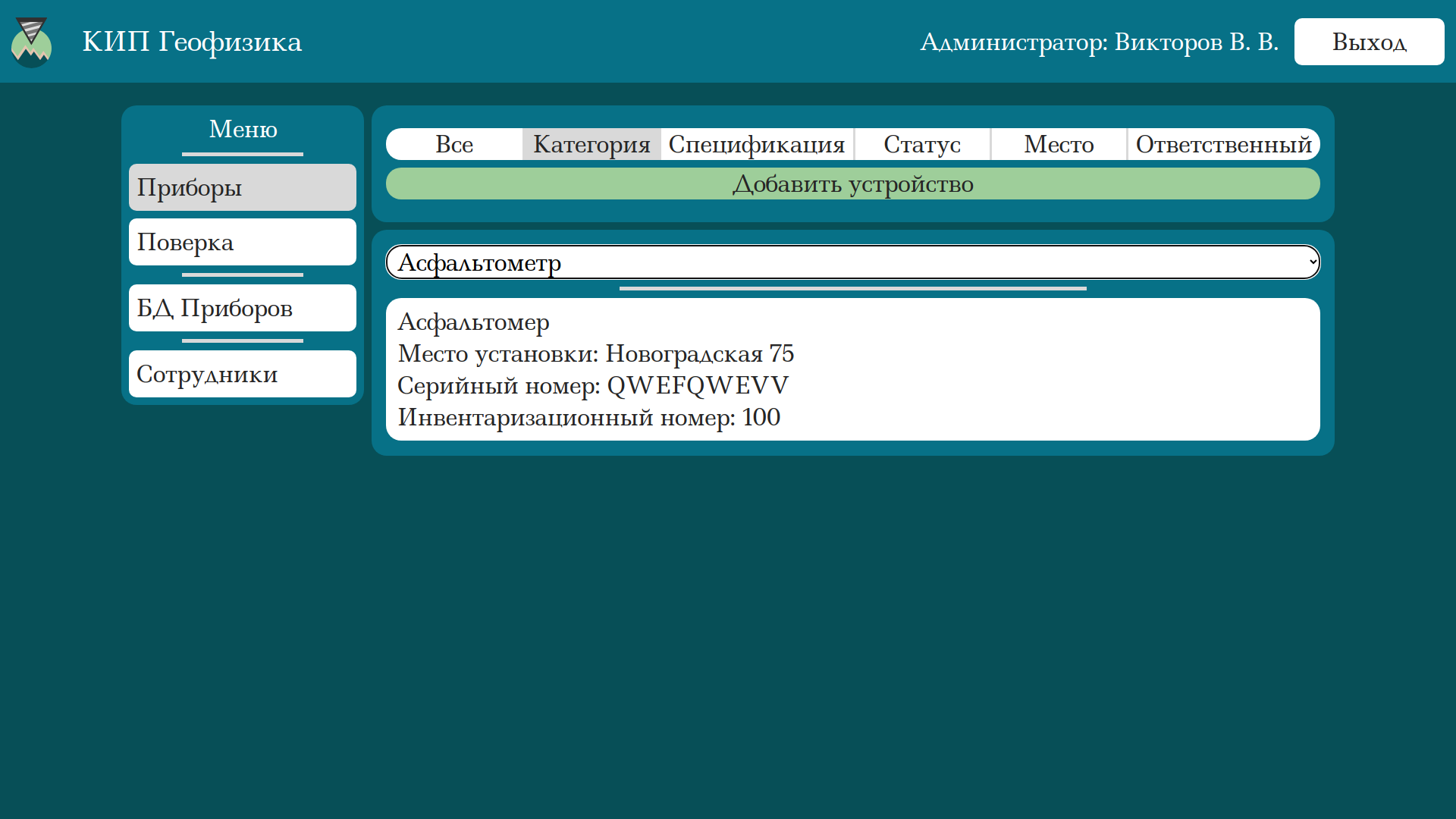
# Приложение Е Страница с меню



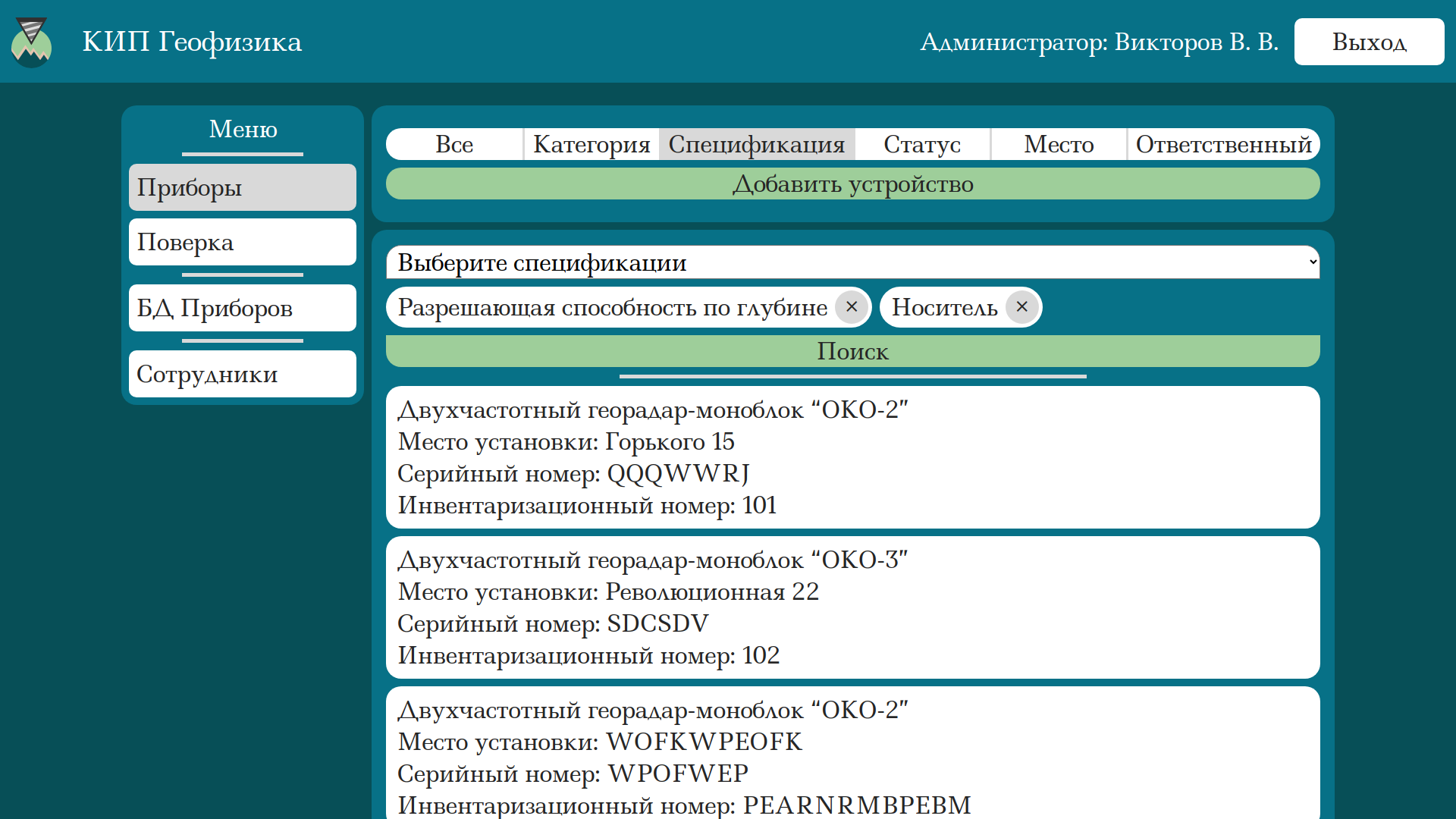
# Приложение Ё Отчёт со всеми устройствами



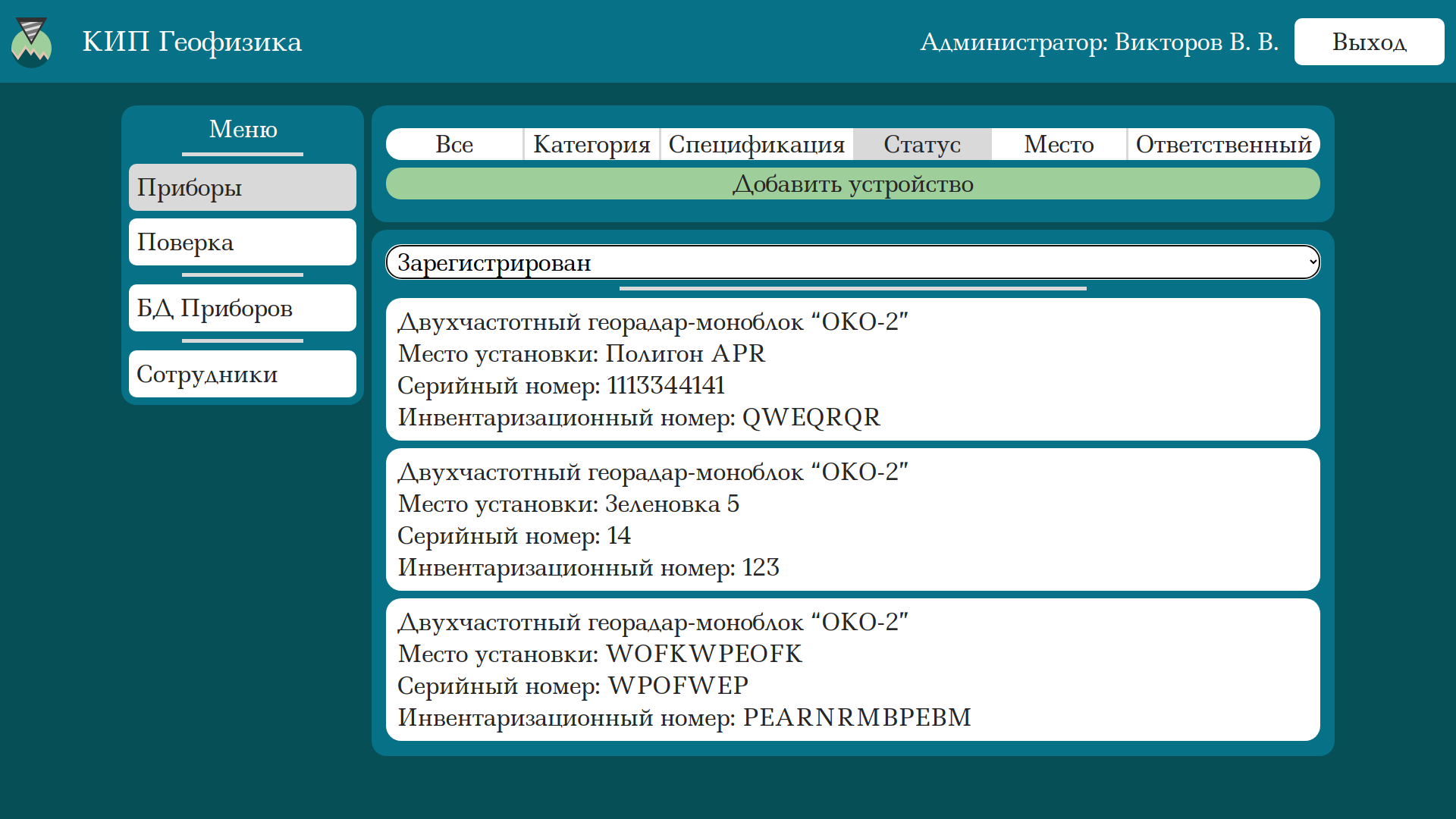
# Приложение Ж Отчёт с устройствами по категориям



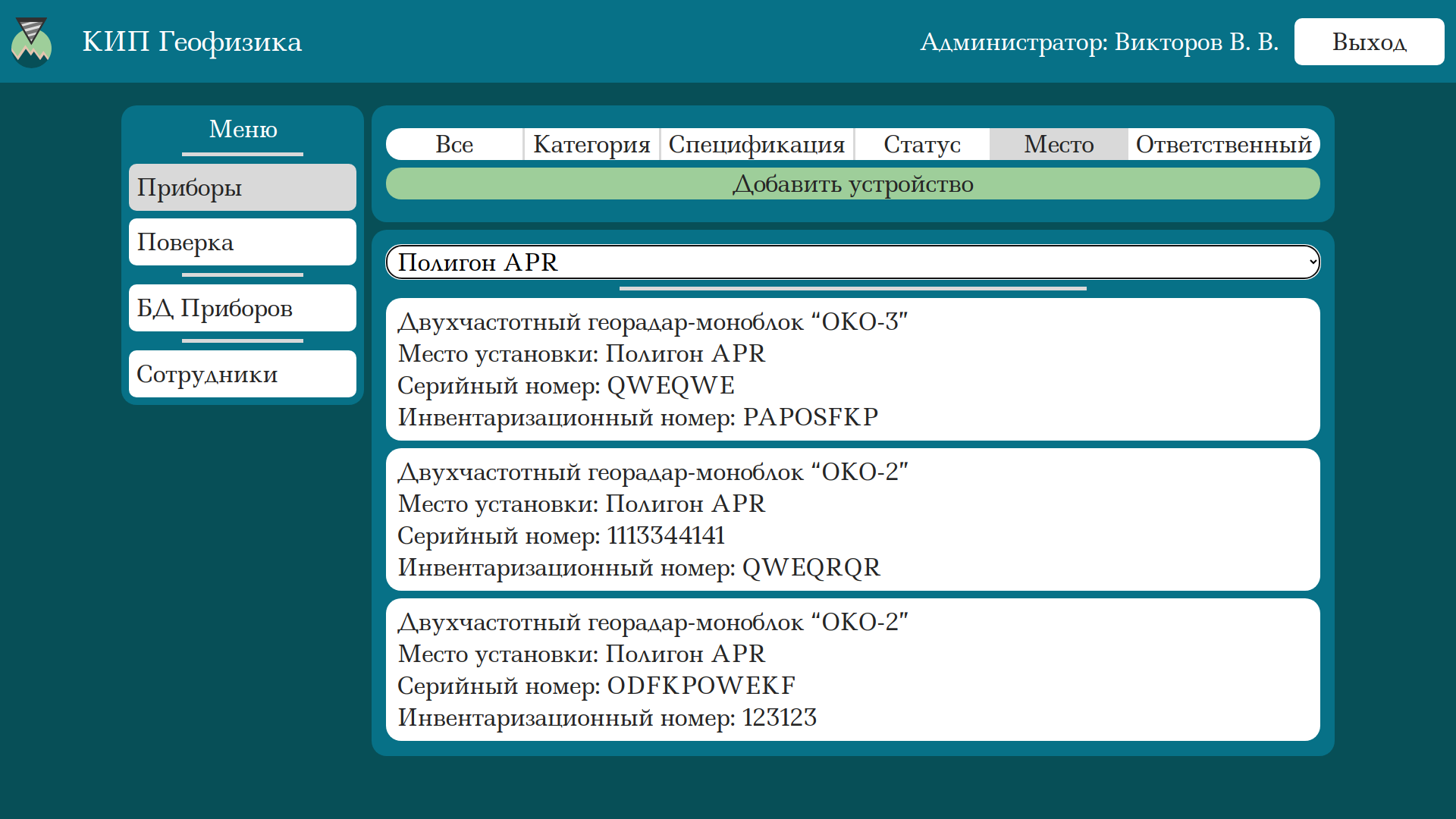
# Приложение З Отчёт с устройствами по спецификациям



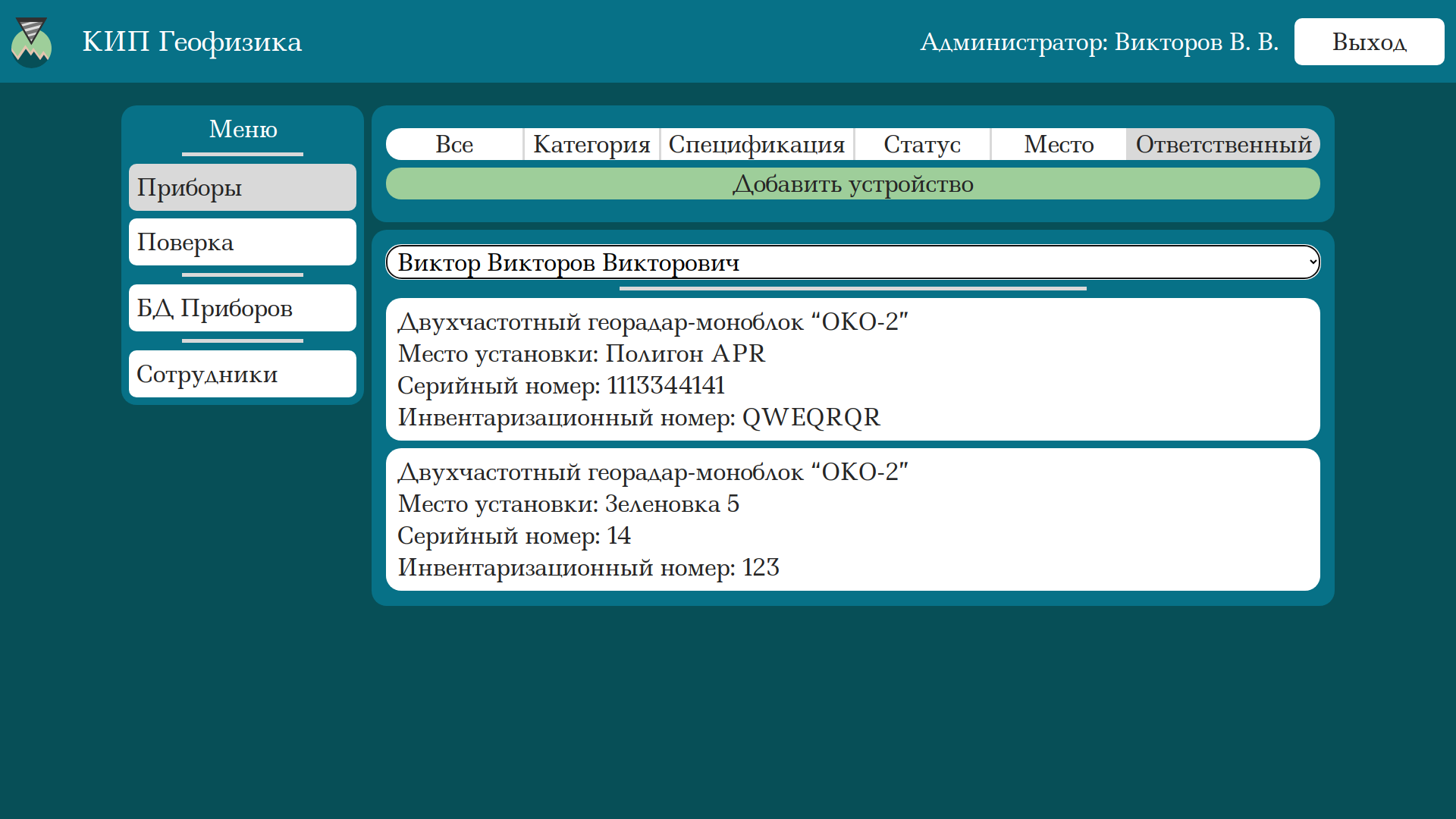
# Приложение И Отчёт с устройствами по текущим статусам



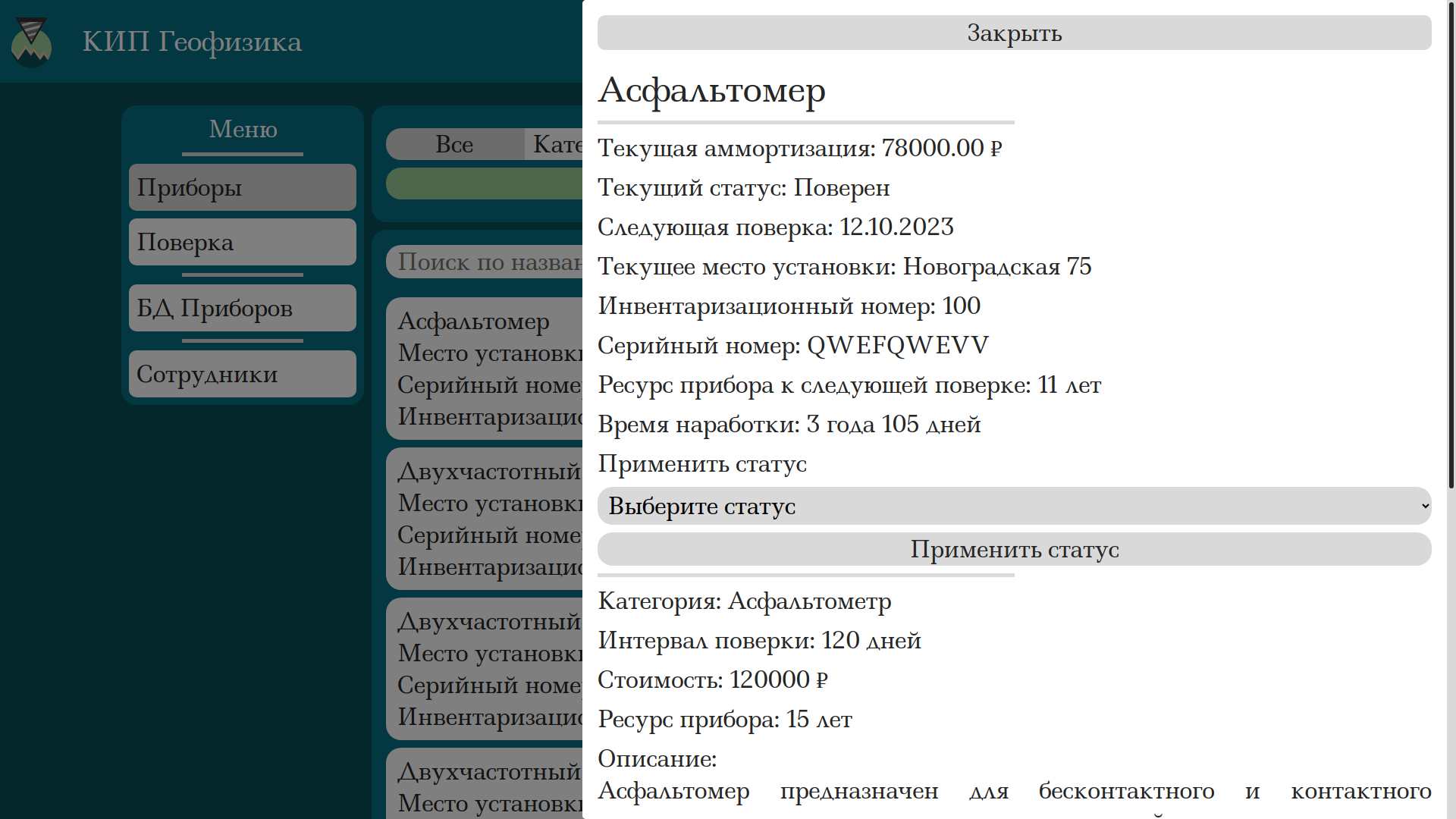
# Приложение К Отчёт с устройствами по местам установки



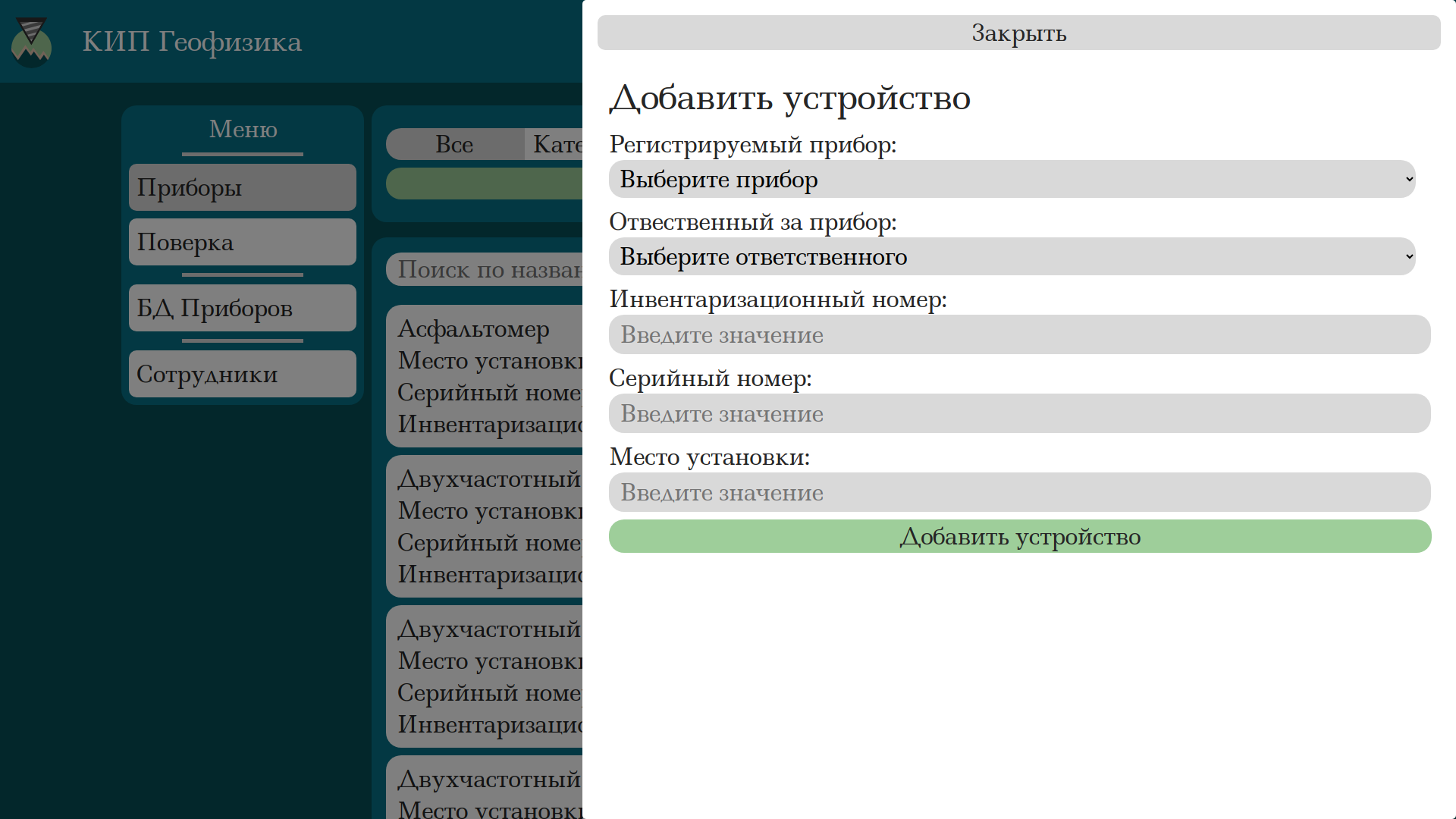
# Приложение Л Отчёт с устройствами по закреплённым за ними сотрудниками



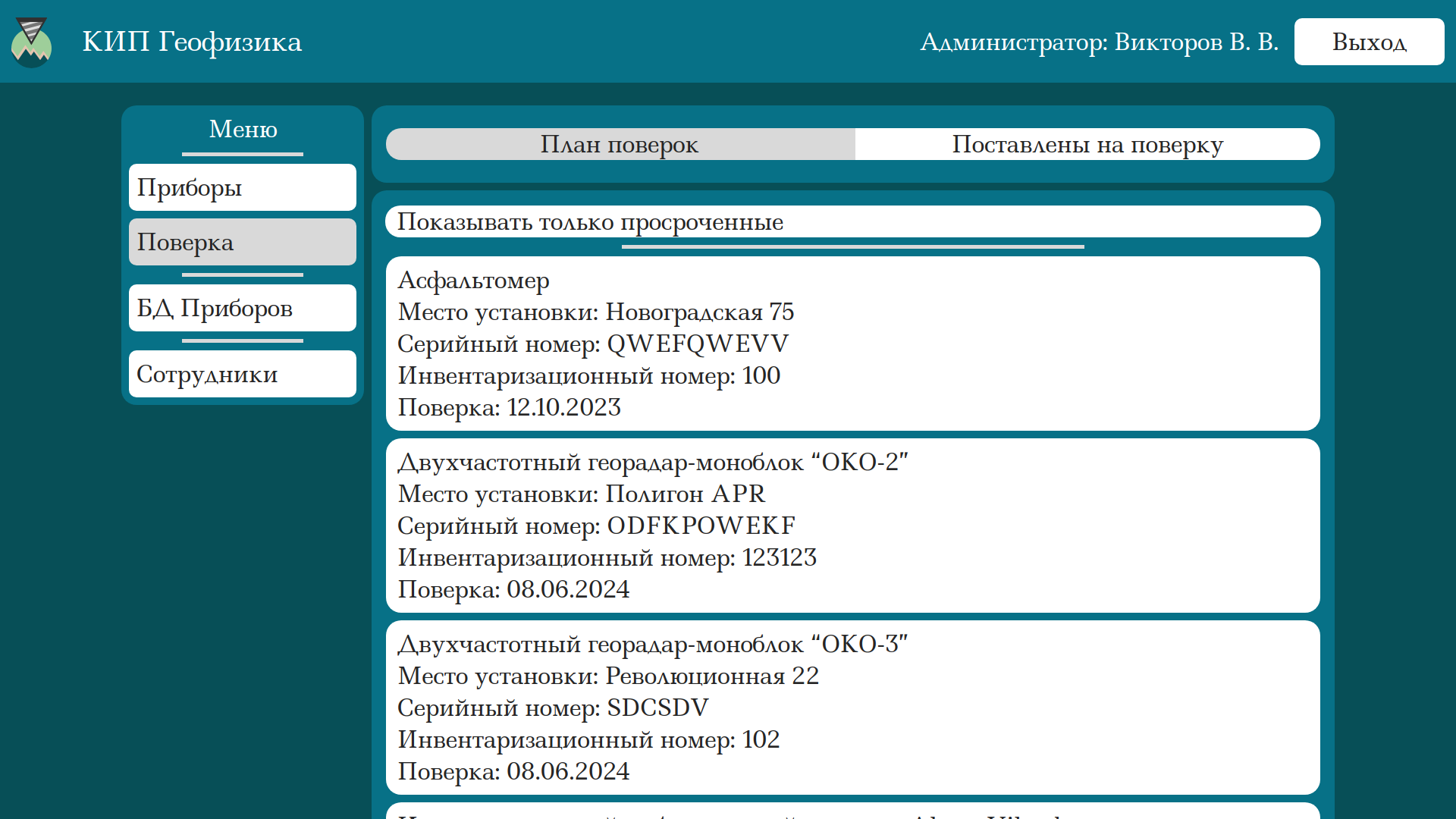
# Приложение М Отчёт с полной информацией об устройстве



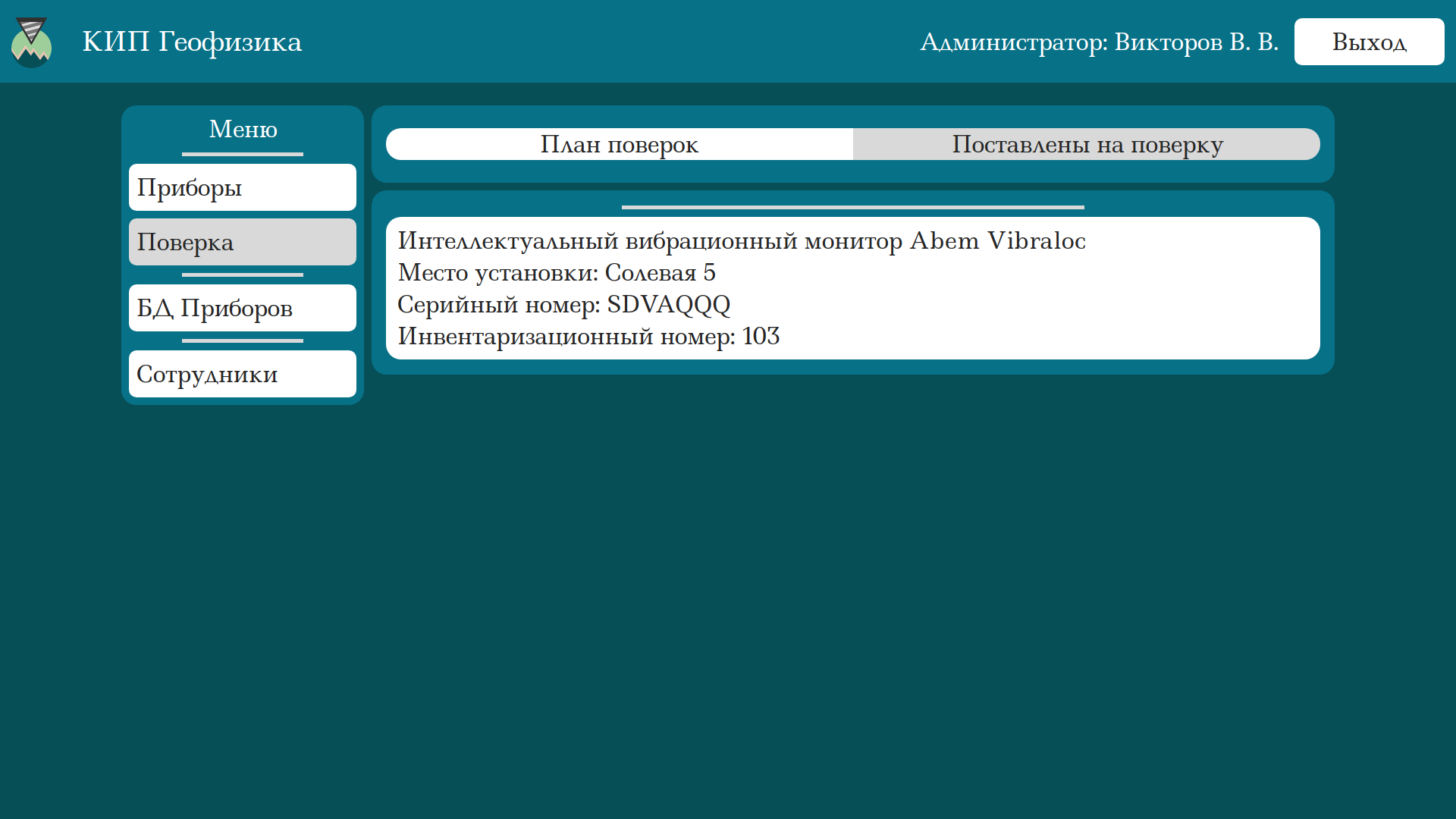
# Приложение Н Форма добавления устройства



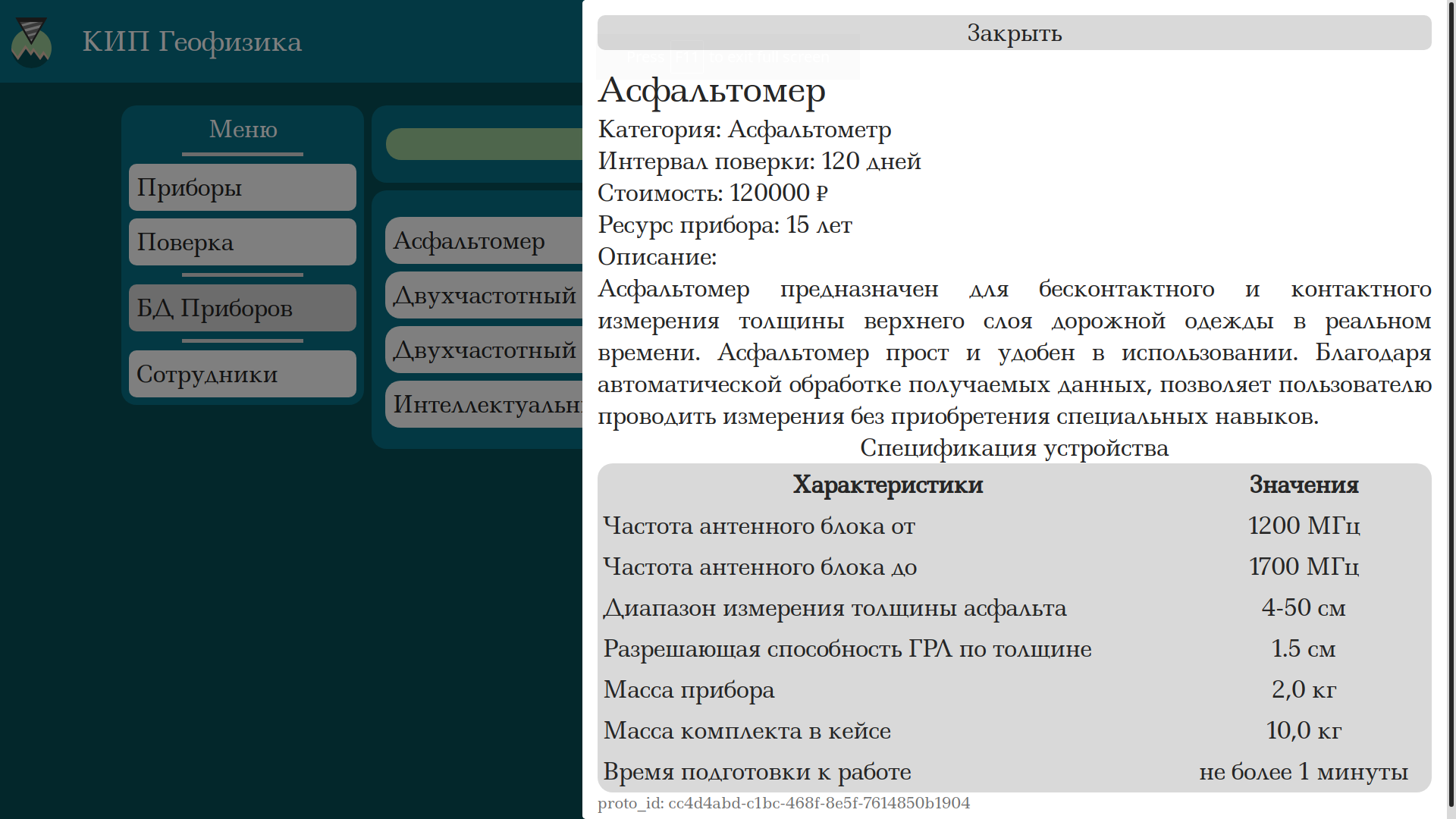
# Приложение О Отчёт по плану поверок



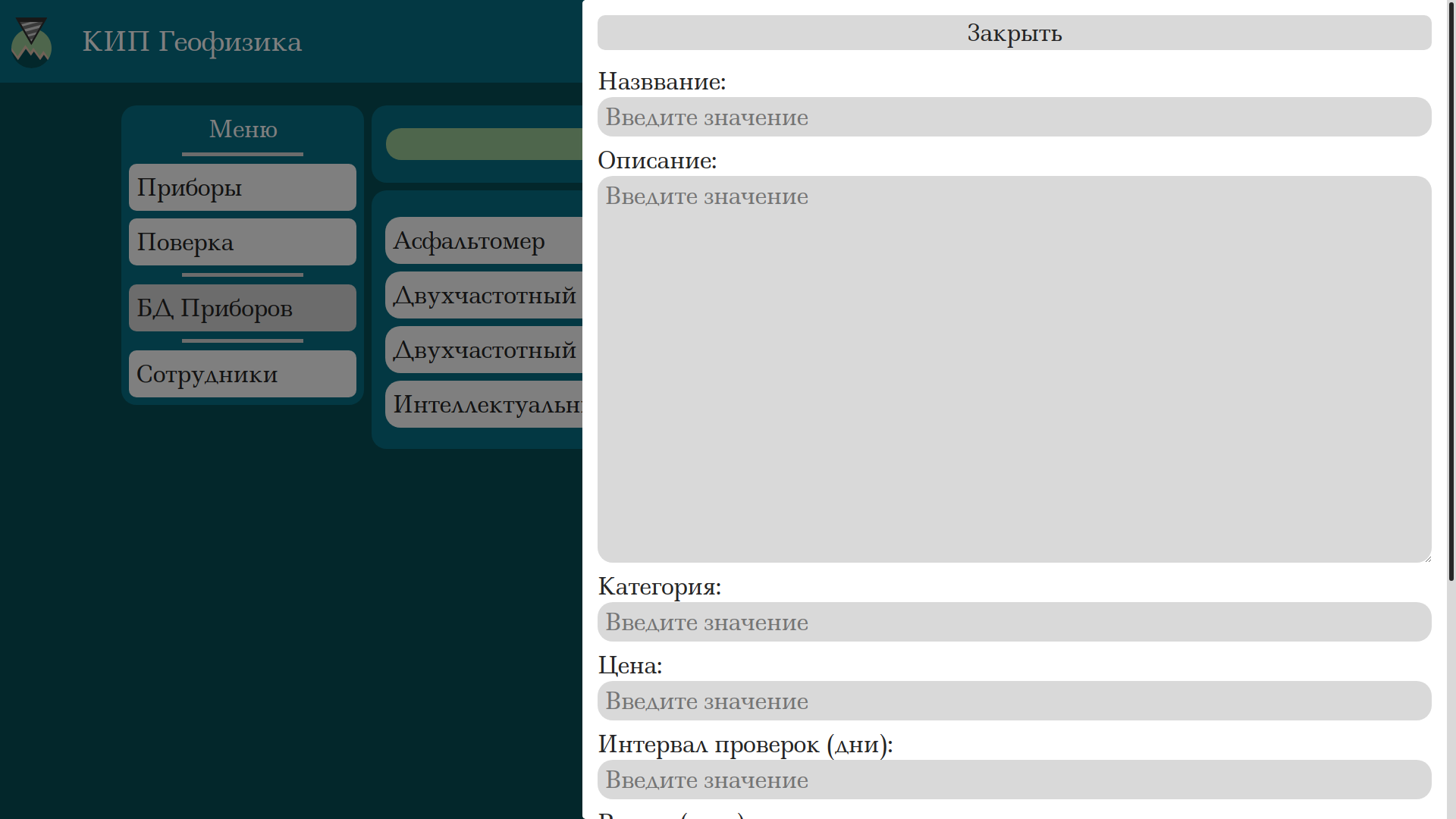
# Приложение П Отчёт по приборам, со статусом назначенной поверки



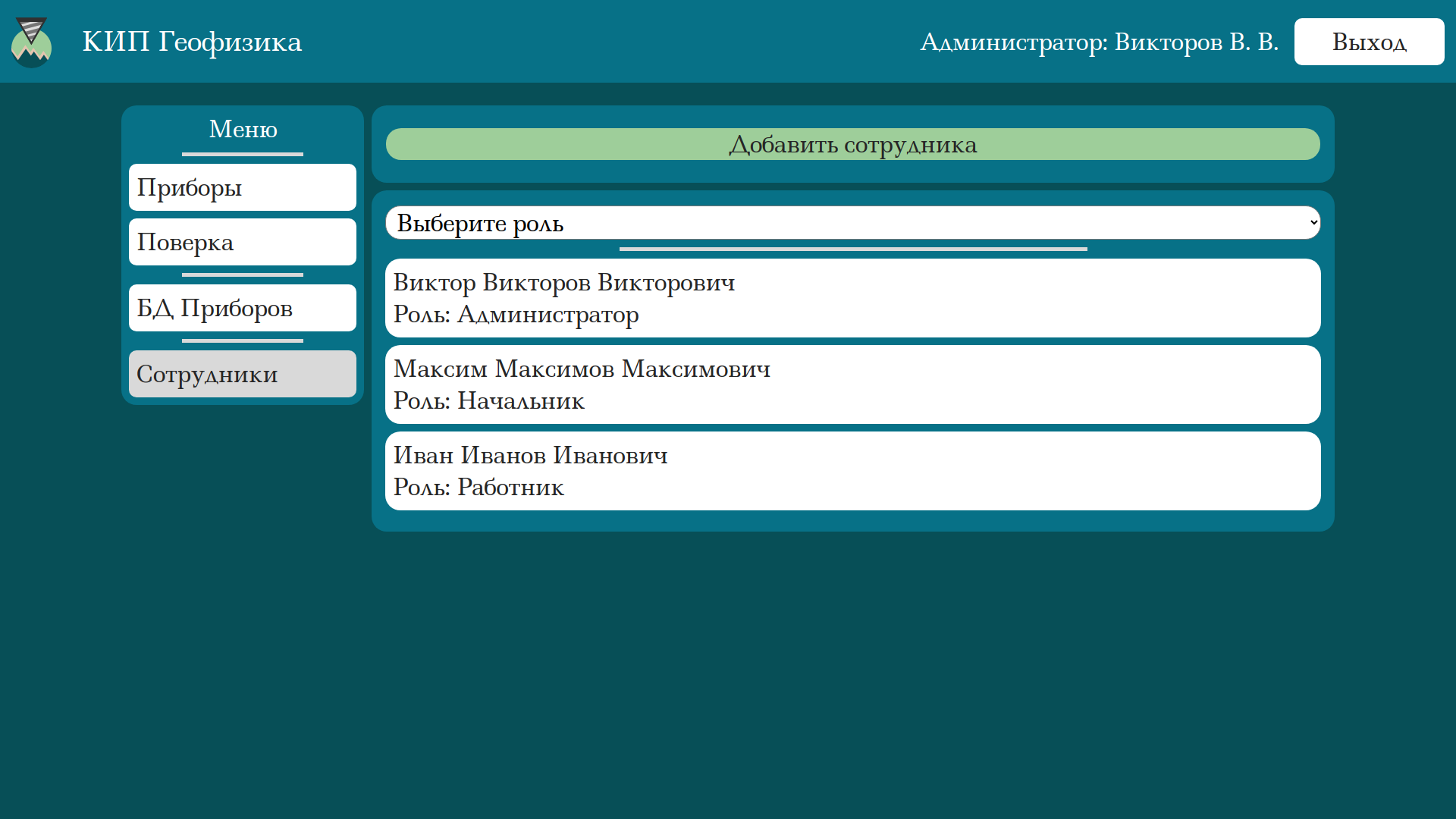
# Приложение Р Отчёт о известных устройствах



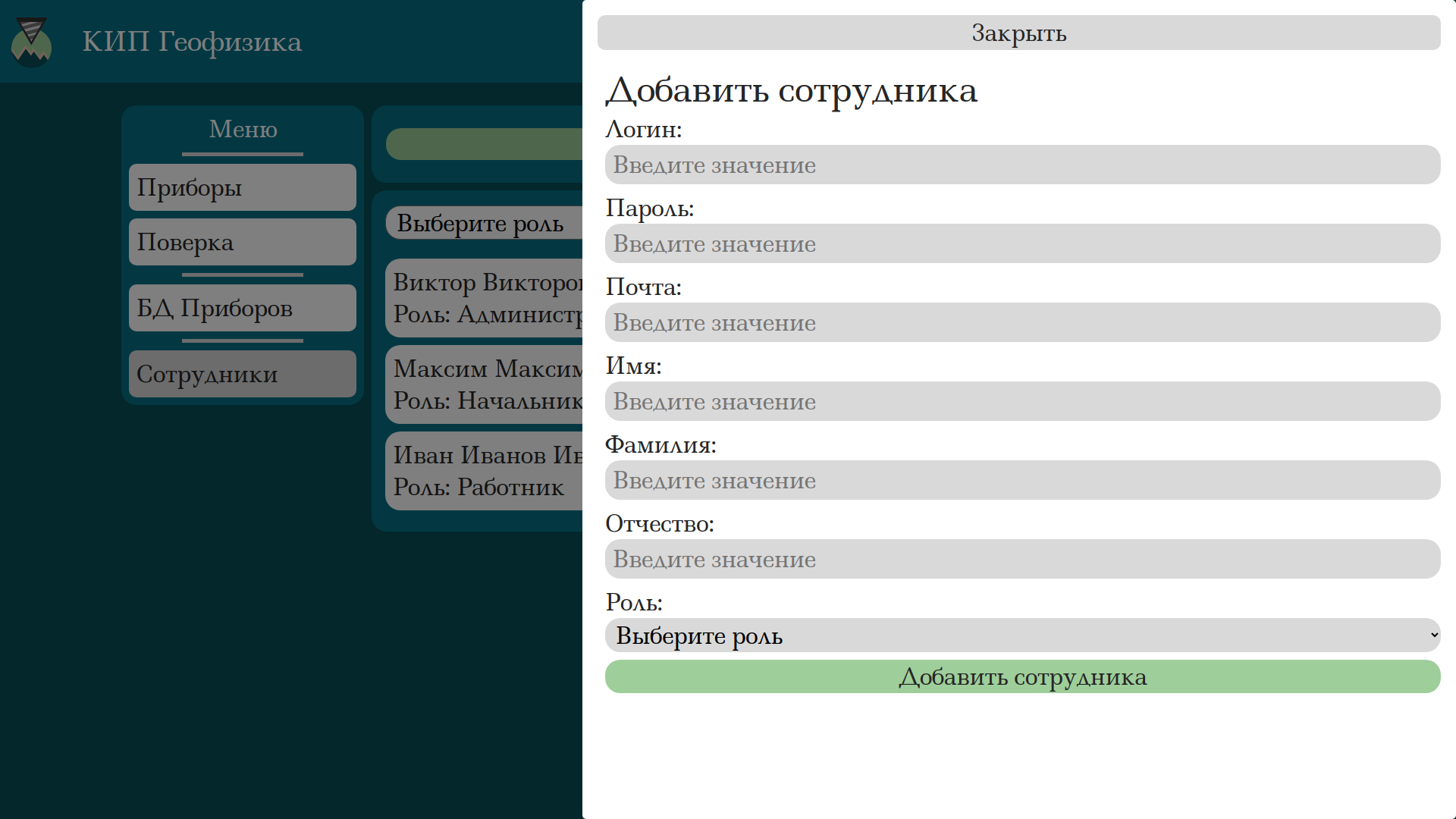
# Приложение С Форма добавления нового прибора



# Приложение Т Отчёт о сотрудниках



# Приложение У Форма добавления нового сотрудника



# Приложение Ф Антиплагиат

