Практическая работа № 7

Построение диаграммы классов UML при проектировании программного обеспечения

Цель работы:

1.1. Закрепить теоретические знания по принципам создания диаграммы классов.

1.2. Получить практические навыки по построению диаграммы классов.

Задание на практическую работу

1. Составить перечень правил языка UML

|  |  |
| --- | --- |
| № | Правило |
| 1 | Класс - элемент диаграммы, обозначающий множество объектов, обладающих  одинаковой внутренней структурой, поведением и отношениями с объектами  других классов. Изображается класс на диаграмме в виде прямоугольника,  разделённого на три секции:   1. Имя класса 2. Список полей класса 3. Список методов класса |
| 2 | В качестве имени класса выбирается существительное в  единственном числе. Если имя класса состоит из нескольких слов, то  записывается слитно, в верблюжьем стиле (от англ. "CamelCase"). |
| 3 | Имя должно быть уникальным в пределах диаграммы. |
| 4 | Статический класс – это набор утилит, не имеет объектов и к имени дописывается модификатор «utility» |
| 5 | Абстрактный класс не содержит объекты и записывается курсивом |
| 6 | Каждое поле класса должно описываться следующим образом:  <уровень видимости><идентификатор>[кратность]: <тип поля>=<начальное значение> {свойство}  <visibility><identificator>[multiplicity]:<type>=<initial value>{property} |
| 7 | Открытое поле – означает, что к поле можно обратиться из любой части программы и это обозначается «+». |
| 8 | Закрытое поле – обозначается «-» и означает, что получить доступ к полю можно получить внутри класса |
| 9 | Защищенное поле – обозначается «#» и означает, что получить доступ к полю можно внутри класса и внутри производных классов |
| 10 | Поле содержит идентификатор, который является уникальным |
| 11 | Поле должно содержать ограничение значения, которое называется типом поля |
| 12 | Если у поля есть кратность, то его следует считать массивом |
| 13 | Метод содержит уровень видимости, идентификатор, список аргументов и типа возвращаемого значения |
| 14 | Отношение ассоциации – сплошная линия с наконечником с надписью наверху в виде глагола, а внизу в виду кратности. Стрелка ассоциации направлена от класса пользователя к классу владельцу используемой функциональности. |
| 15 | Если кратность ассоциации не указана, то будет подразумеваться кратность в виде [0..\*], в случае чего со статическими классами кратность не указывается, там показана [1] |
| 16 | Отношение зависимости визуализируется в виде пунктирной линии с заостренным наконечником и показывает, что изменение одного класса требует изменения другого класса. |
| 17 | Отношение зависимости используется для изменения одного класса, который требует изменения другого класса. |
| 18 | В отношении наследования стрела направлена от производного класса к базовому. |
| 19 | Отношение наследования используется для того, чтобы показать, кто является родителем для другого класса (потомка) |
| 20 | Отношение агрегации показывает между двумя классами, что один из них включает в себя другой класс в качестве составной части |
| 21 | В отношении агрегации идет параллельное разделение классов на две составляющие для объединения в класс, который будет включать несколько частей |
| 22 | Объект класса состоит из объектов, которые его дополняют |
| 23 | Класс-часть может существовать обособленно от класса-целого |
| 24 | Надо использовать кратность, чтобы показать, как объект одного класса входит в другой класс |
| 25 | В отношении агрегации надо использовать кратность, чтобы показать, сколько объектов одного класса входят в состав объекта другого класса |
| 26 | Отношение композиции является частным случаем отношения агрегации |
| 27 | Части, которые соединяют обособленно не могут существовать без целого класса |

1. Укажите результаты:
   1. Практической работы «Диаграмма вариантов использования»:

Роль «Администратор»:

Особенности:

* Управление заявками:
  + Просмотр всех заявок
  + Изменение статуса и приоритета
  + Удаление заявок
  + Экспорт/импорт заявок
  + Управление метаданными
* Управление пользователями:
  + Просмотр списка пользователей
  + Добавление новых пользователей
  + Редактирование пользователей
  + Удаление пользователей
  + Управление ролями
* Архивирование и резервное копирование:
  + Настройка резервного копирования
  + Создание резервных копий
  + Восстановление из резервных копирований
  + Архивация данных
* Мониторинг производительности:
  + Просмотр характеристик системы
  + Генерация отчетов
* Настройка системы:
  + Изменение общих настроек
  + Управление уведомлениями
* Безопасность и доступ:
  + Управление правами доступа
  + Отслеживание последних входов

Роль «Оператор»:

Особенности:

* Управление очередью заявок:
  + Просмотр входящих заявок
  + Присвоение заявок
  + Определение приоритета
  + Распределение заявок
  + Эскалация заявок
* Работа с деталями заявок:
  + Просмотр деталей заявки
  + Ответ на заявку
  + Добавление комментариев и материалов
  + Смена статуса заявки
  + Просмотр истории обработки
* Экспорт и отчетность:
  + Экспорт данных заявок
  + Генерация отчетов

Роль «Пользователь»:

Особенности:

* Создание заявок:
  + Форма создания заявки
  + Отправка заявки
* Просмотр и управление заявками:
  + Просмотр всех заявок
  + Фильтрация заявок
  + Быстрый поиск
  + Редактирование заявок
  + Просмотр деталей заявки
* Управление профилем:
  + Редактирование личных данных
  + Настройки предпочтений
  + Управление уведомлениями
* Доступ к справочным материалам:
  + База знаний
  + Часто задаваемые вопросы
* Взаимодействие со службой поддержки:
  + Просмотр ответов операторов
  + Добавление комментариев и вложений

2.2. Практической работы «Практическое применение диаграммы в нотации

IDEF1X».

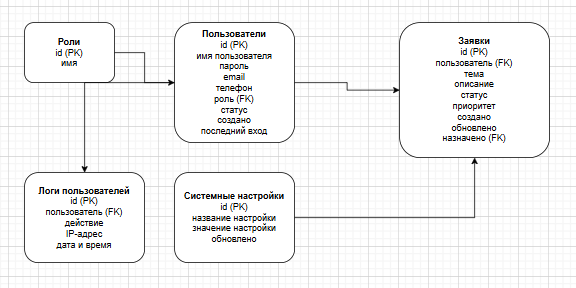


Рисунок 1 - Таблицы

* 1. Создайте макет окон для пользователей ИС (актёра)

Реализованные страницы интерфейса:

1. Окно пользователя:

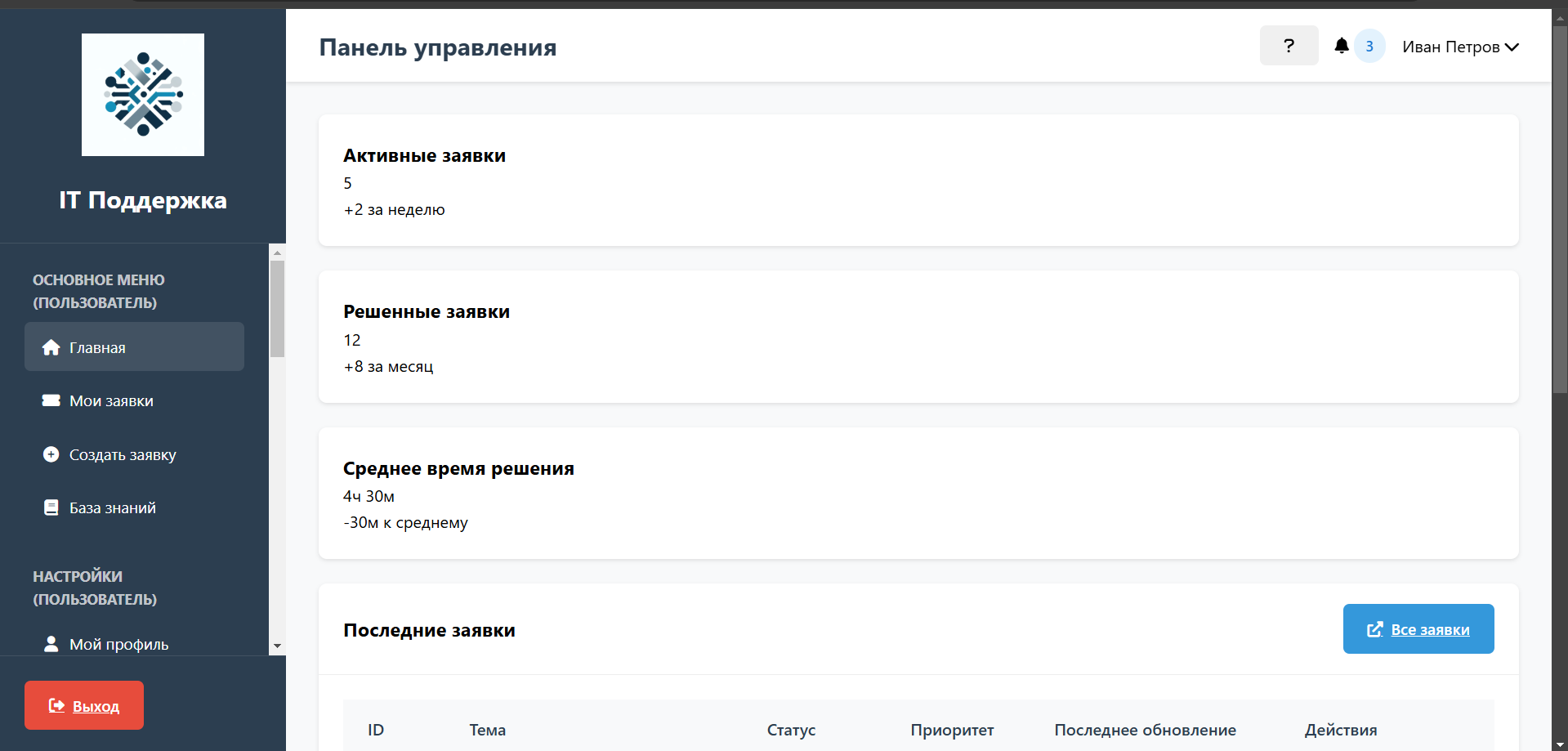


Рисунок № 2 – Окно пользователя

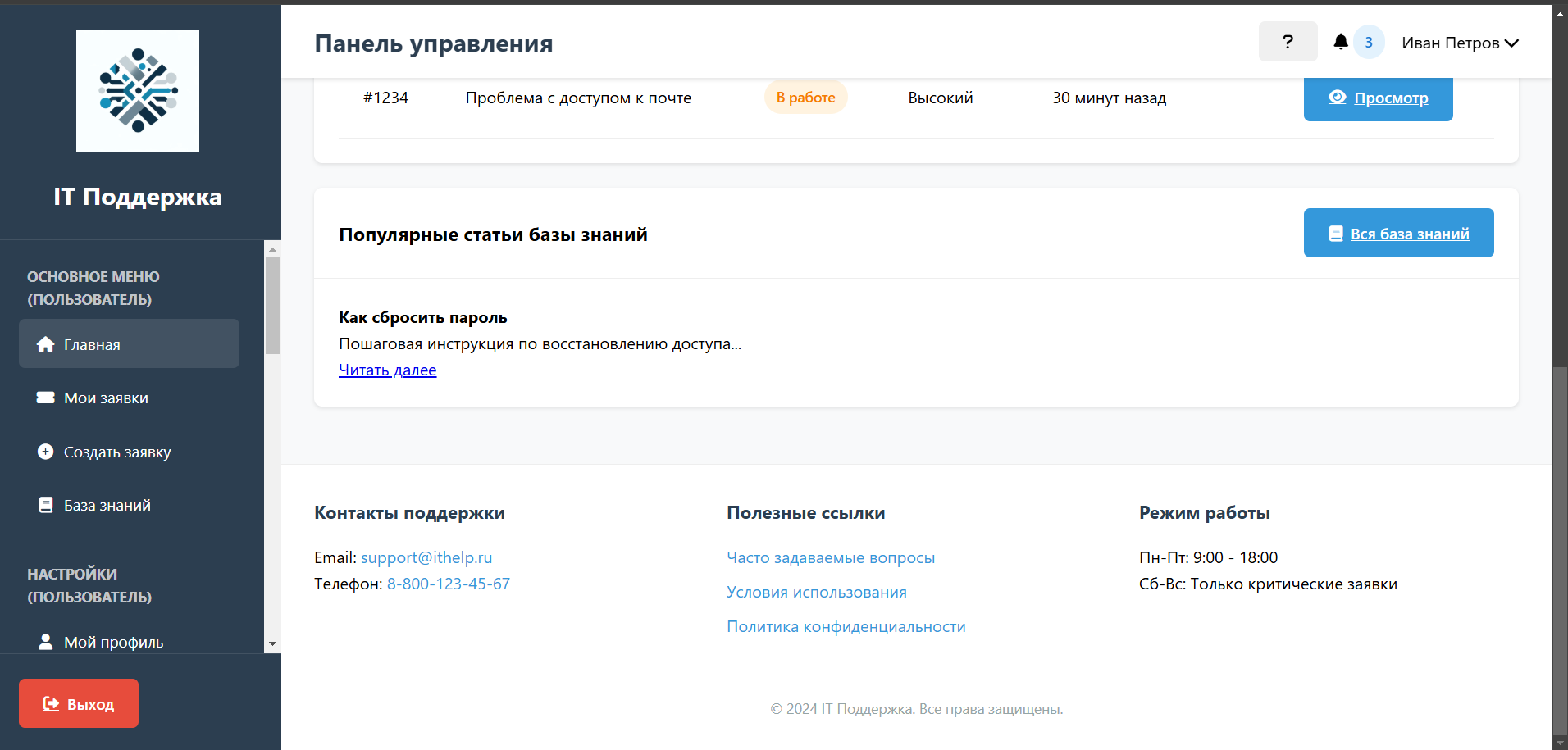


Рисунок № 3 – Окно пользователя

При нажатии на вопросительный знак появляется подсказка

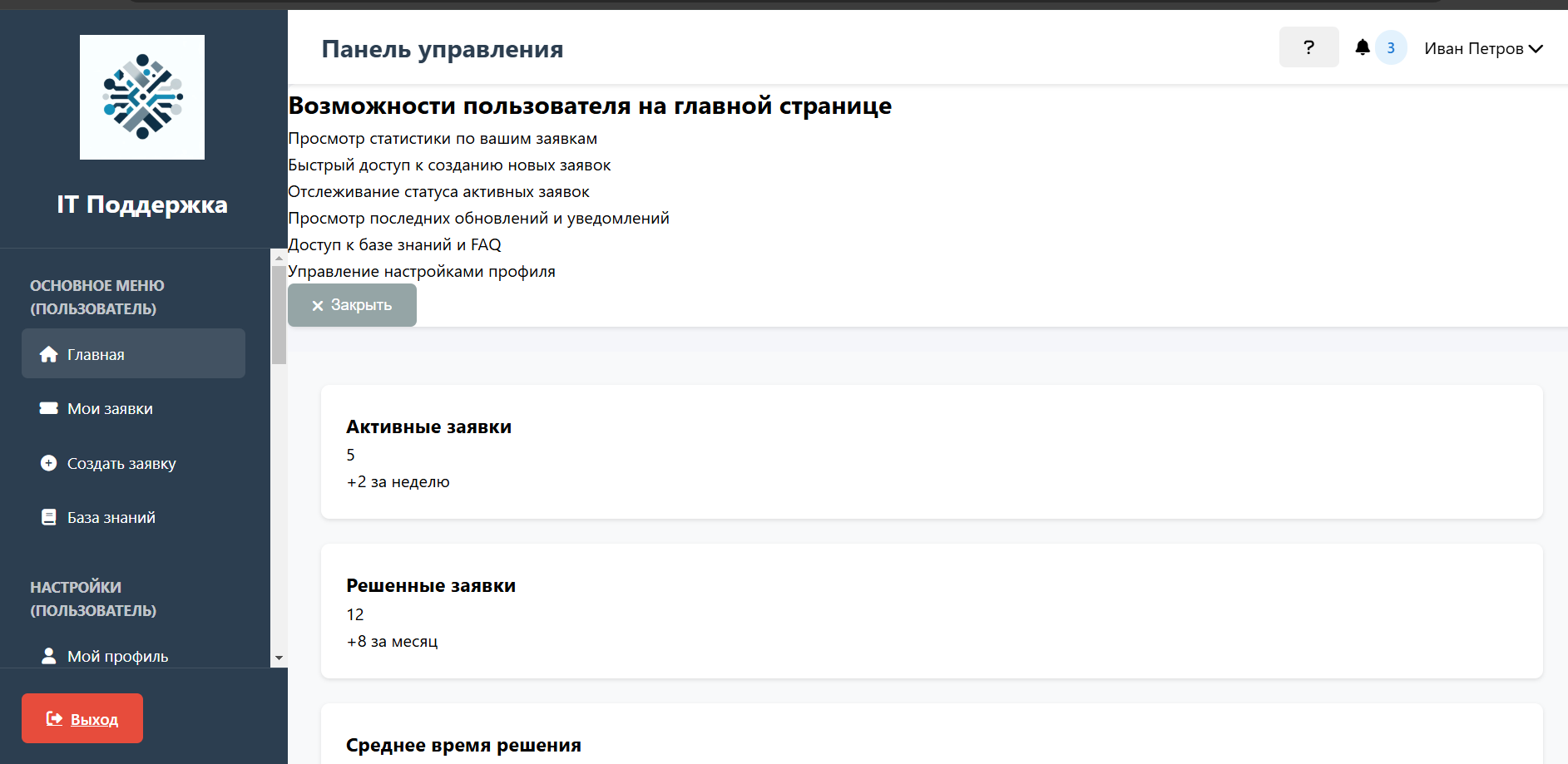


Рисунок № 4 – Возможность пользователя на главной странице

1. Просмотр заявок пользователем

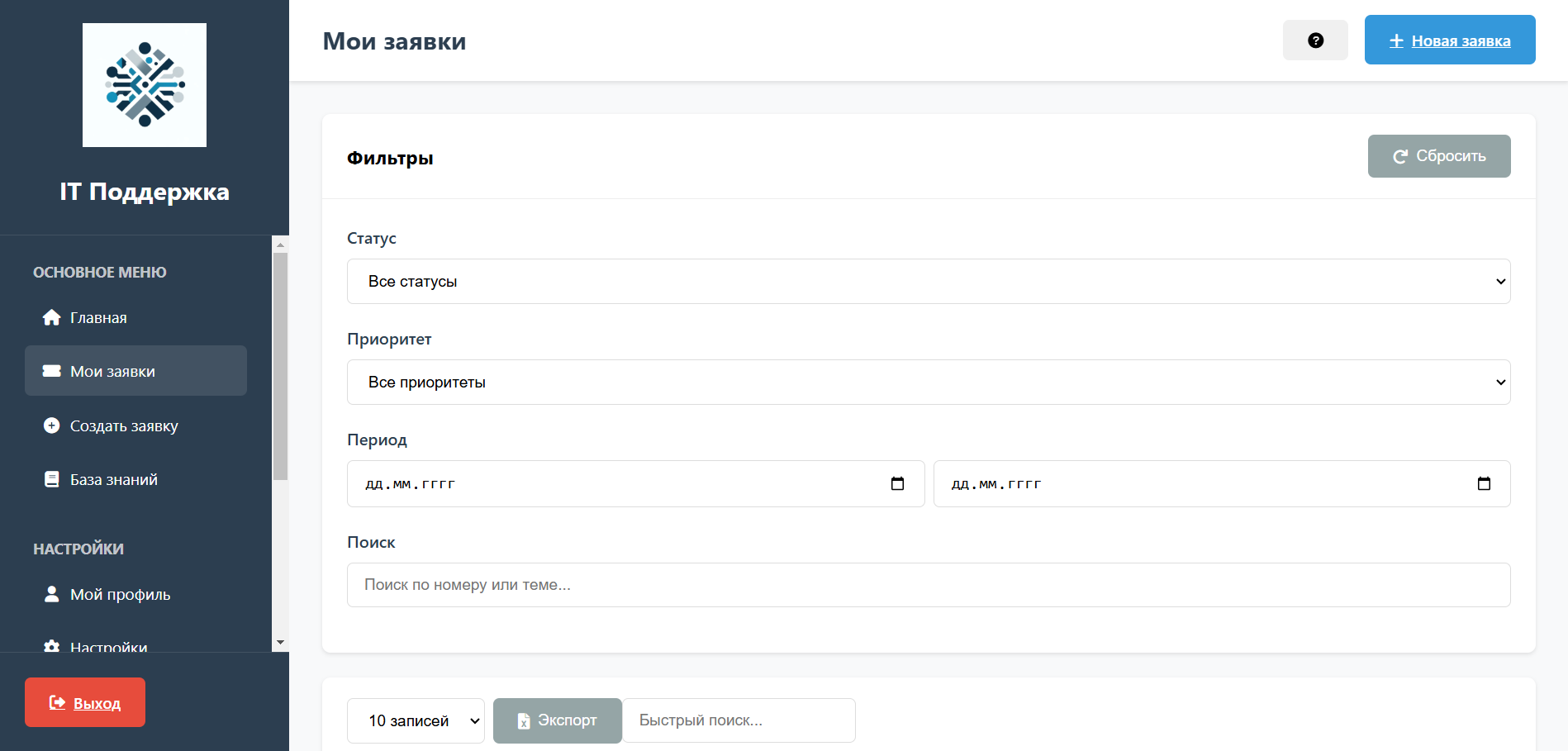


Рисунок № 5 – Просмотр заявок пользователем

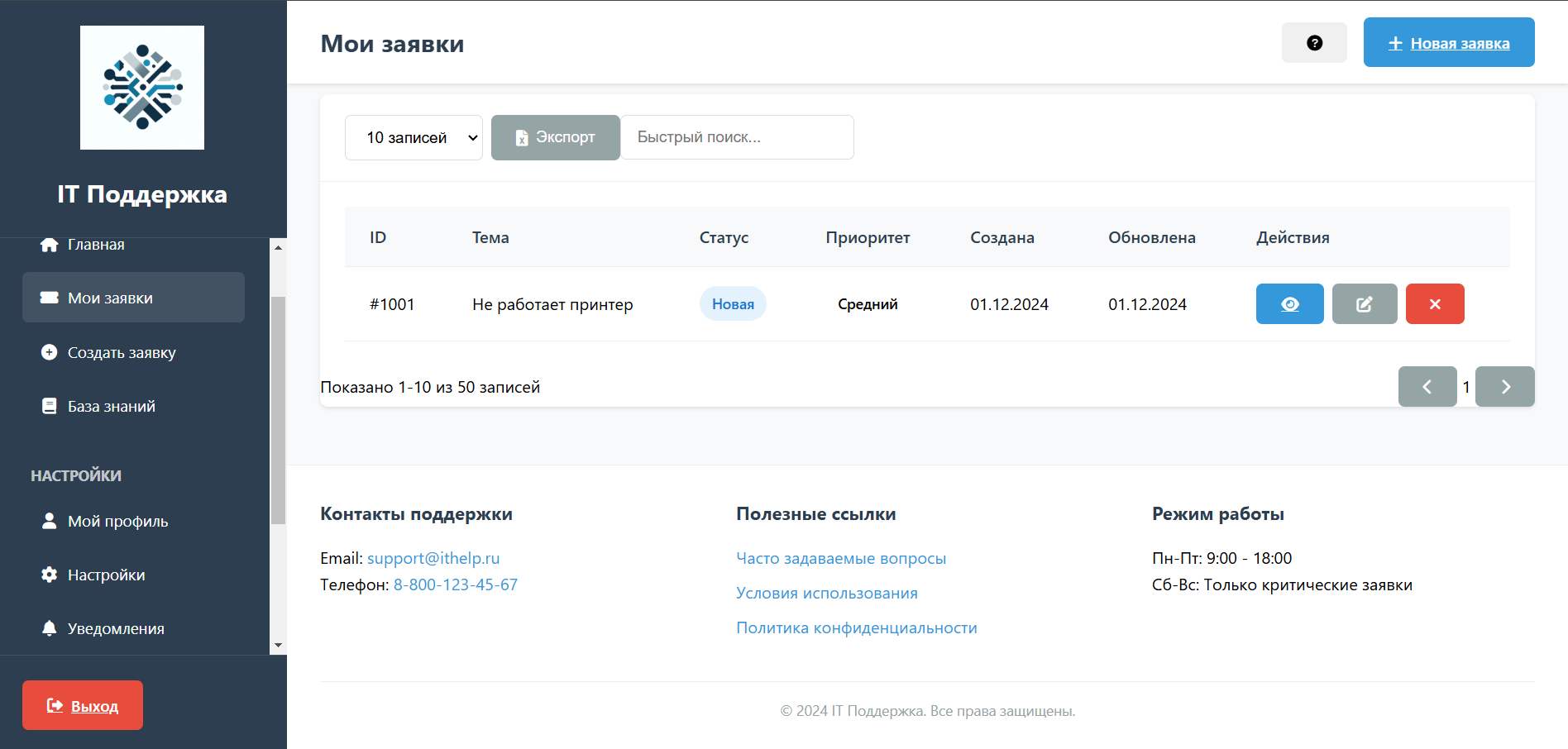


Рисунок № 6 – Просмотр заявок пользователем

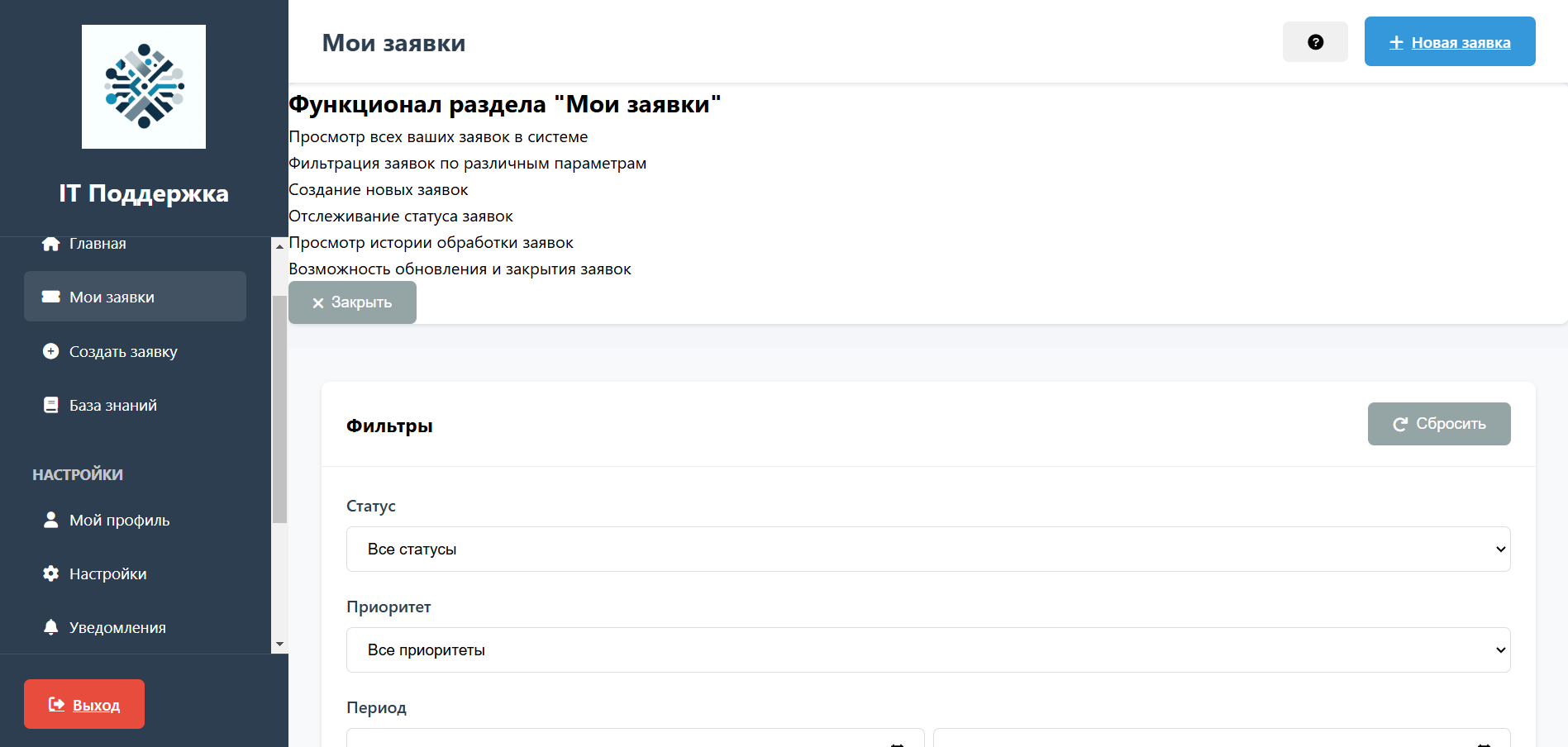


Рисунок № 7 – Подсказка по функционалу

1. Создание заявки пользователем

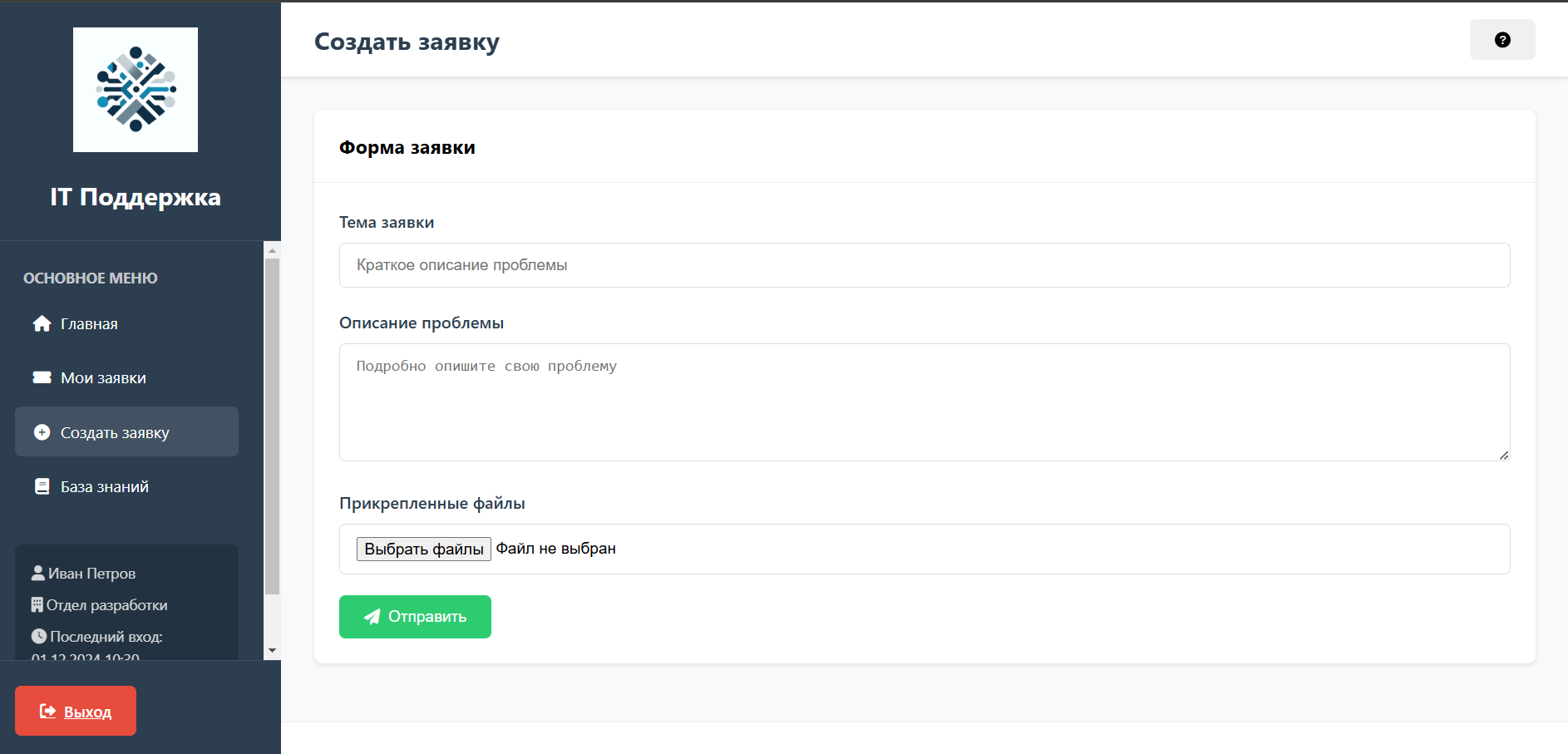


Рисунок № 8 – Создание заявки пользователем

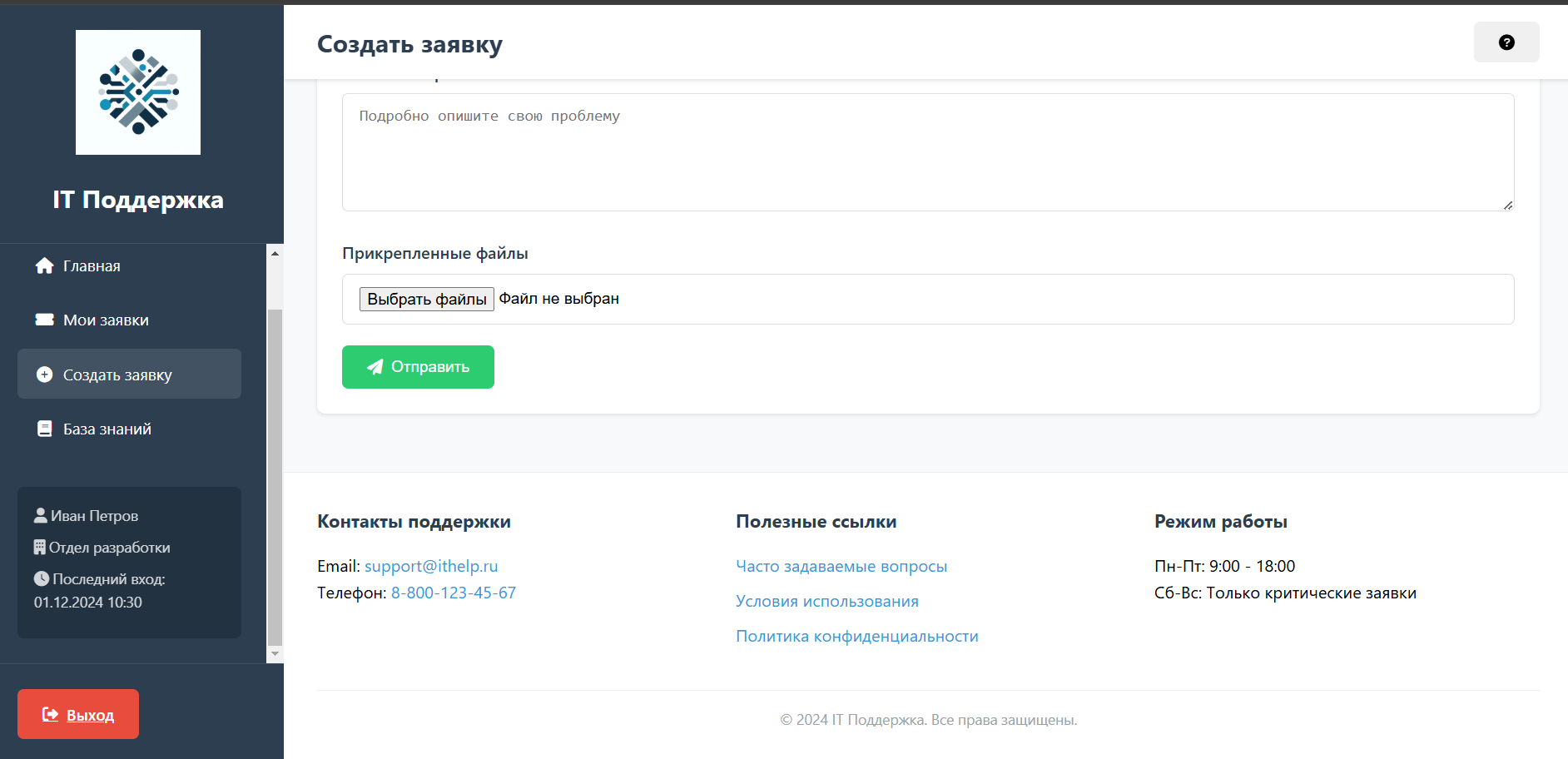


Рисунок № 9 – Создание заявки

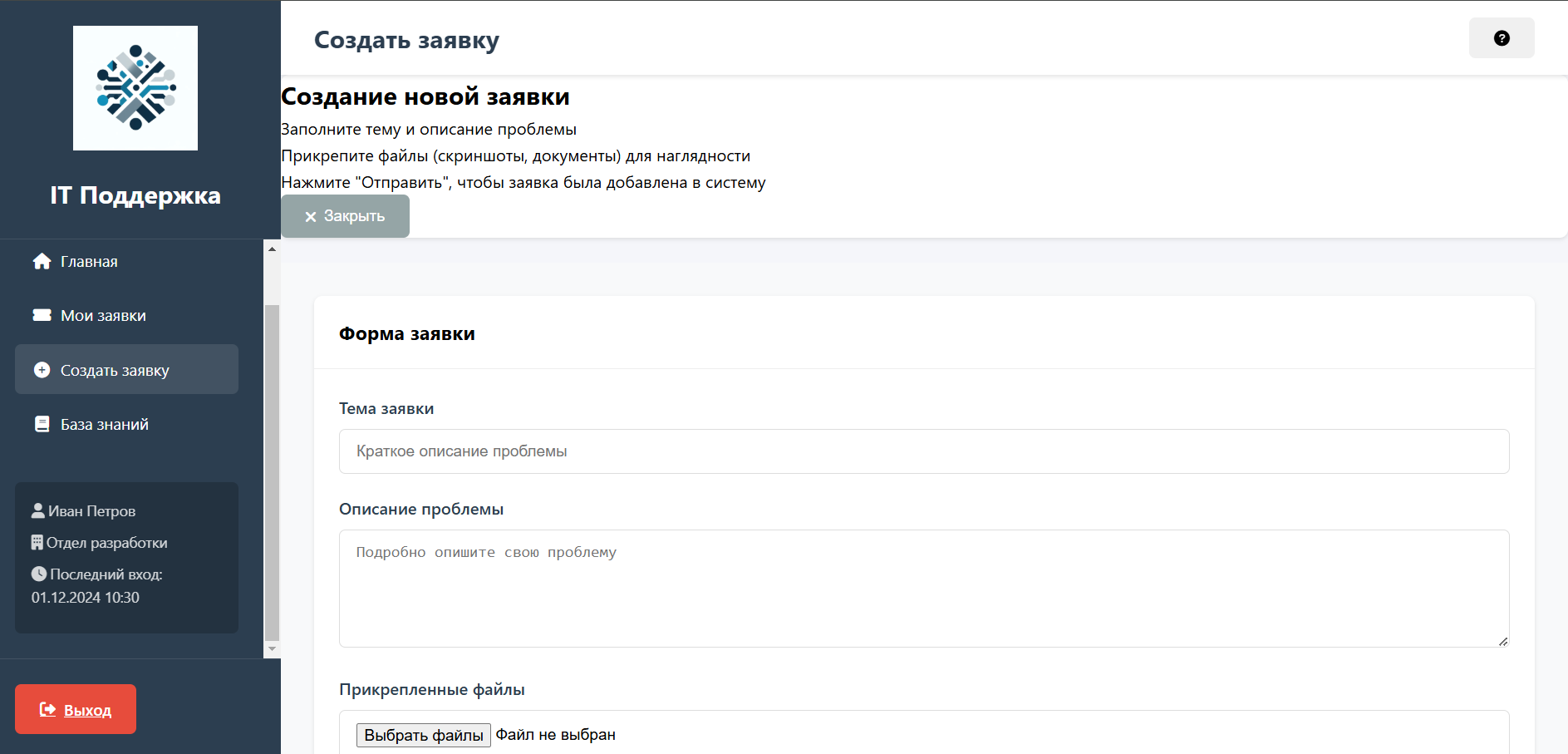


Рисунок № 10 - Подсказка

1. Страница профиля пользователя

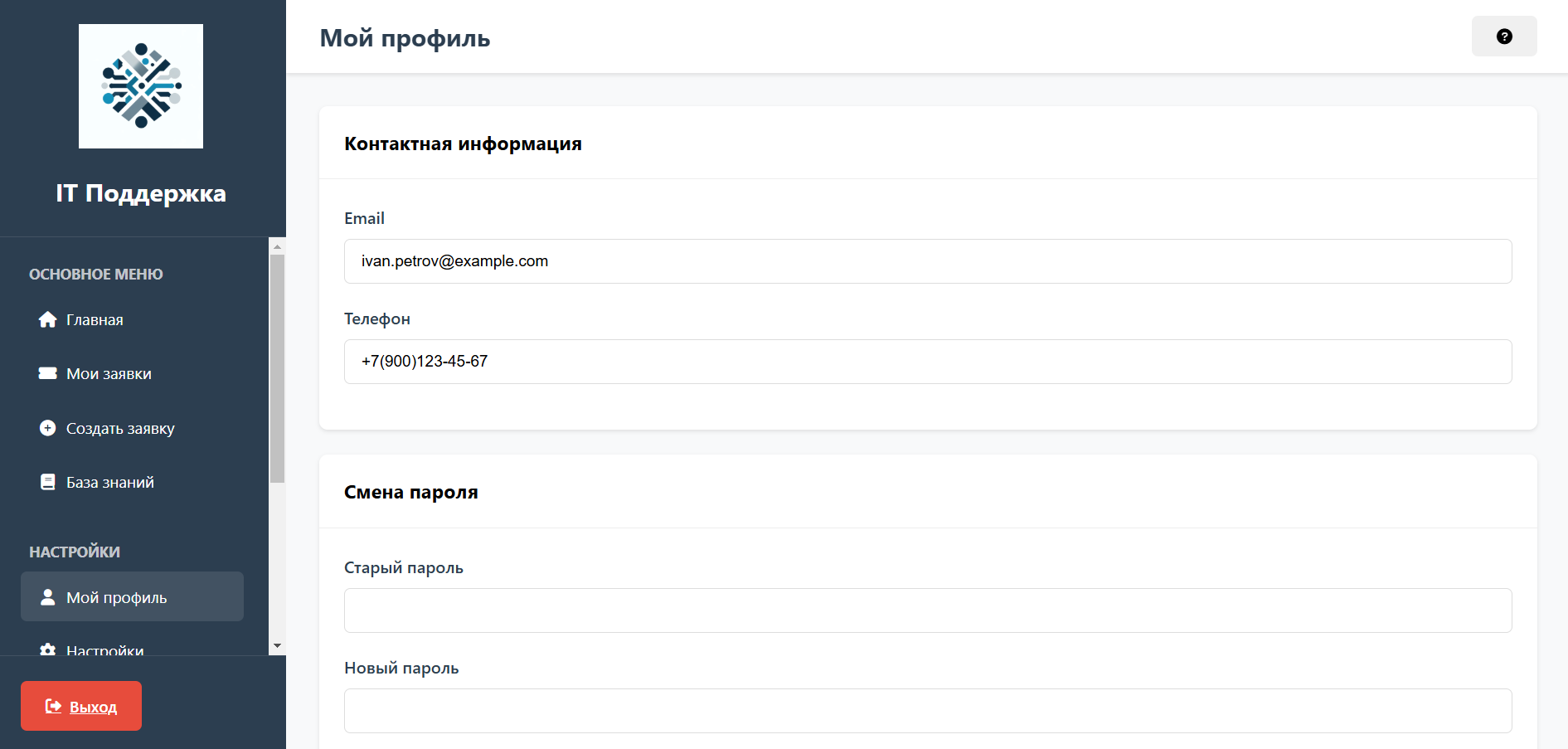


Рисунок № 11 – Страница профиля пользователя

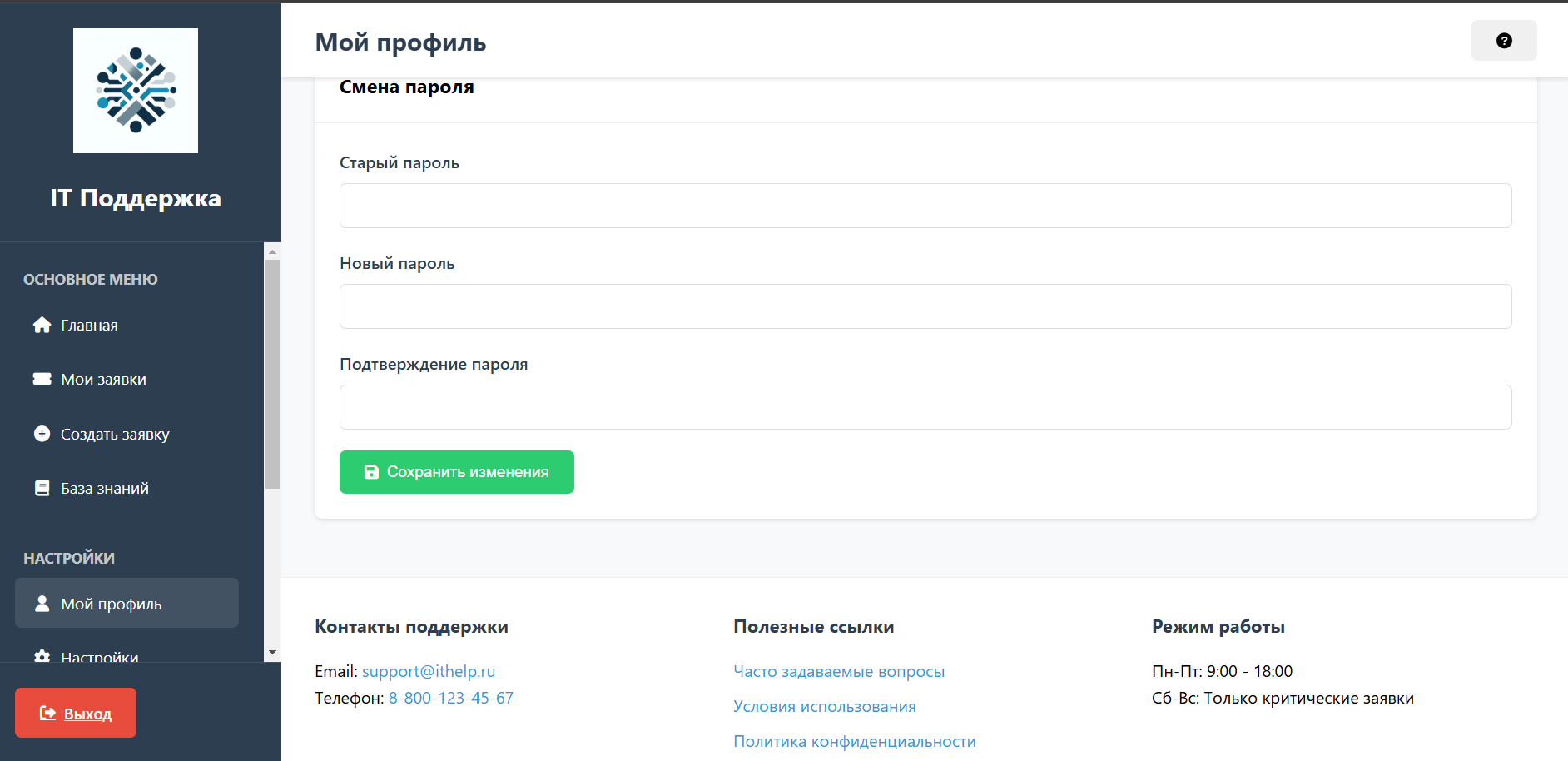


Рисунок № 12 – Страница профиля пользователя

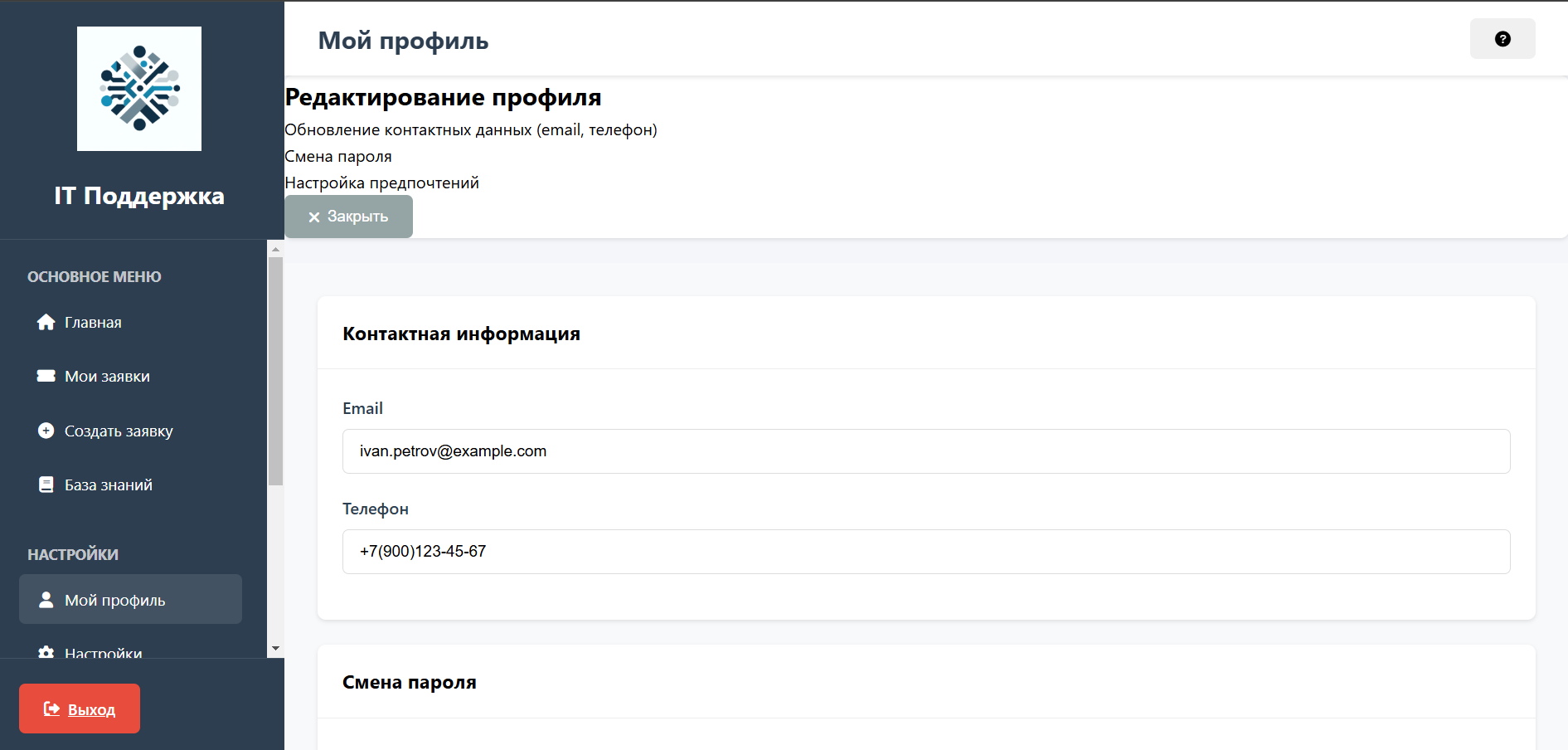


Рисунок № 13 – Подсказка при нажатии на вопросительный знак

1. Окно администратора управления заявка

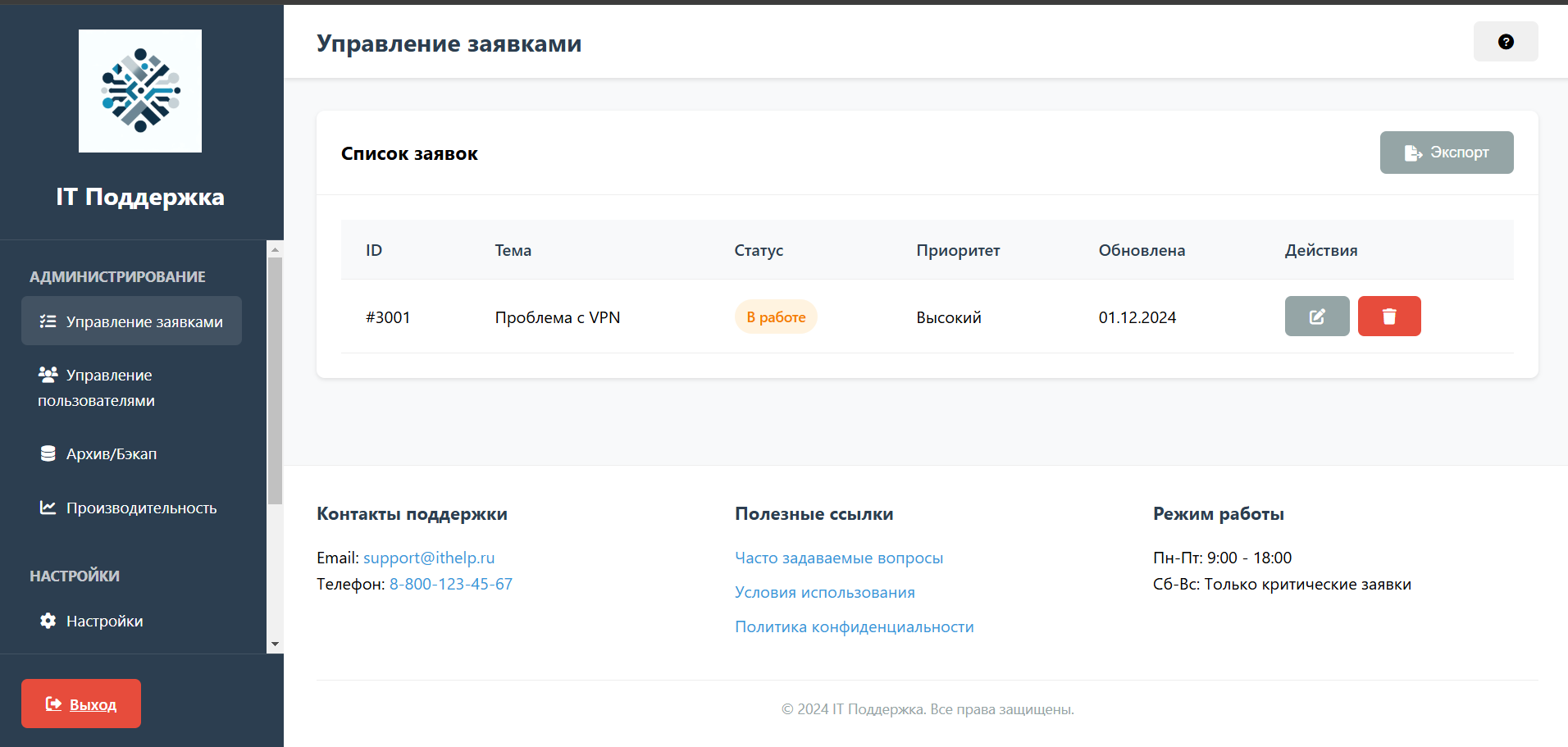


Рисунок № 14 – Окно администратора

При нажатии на кнопку карандаша можно редактировать заявку

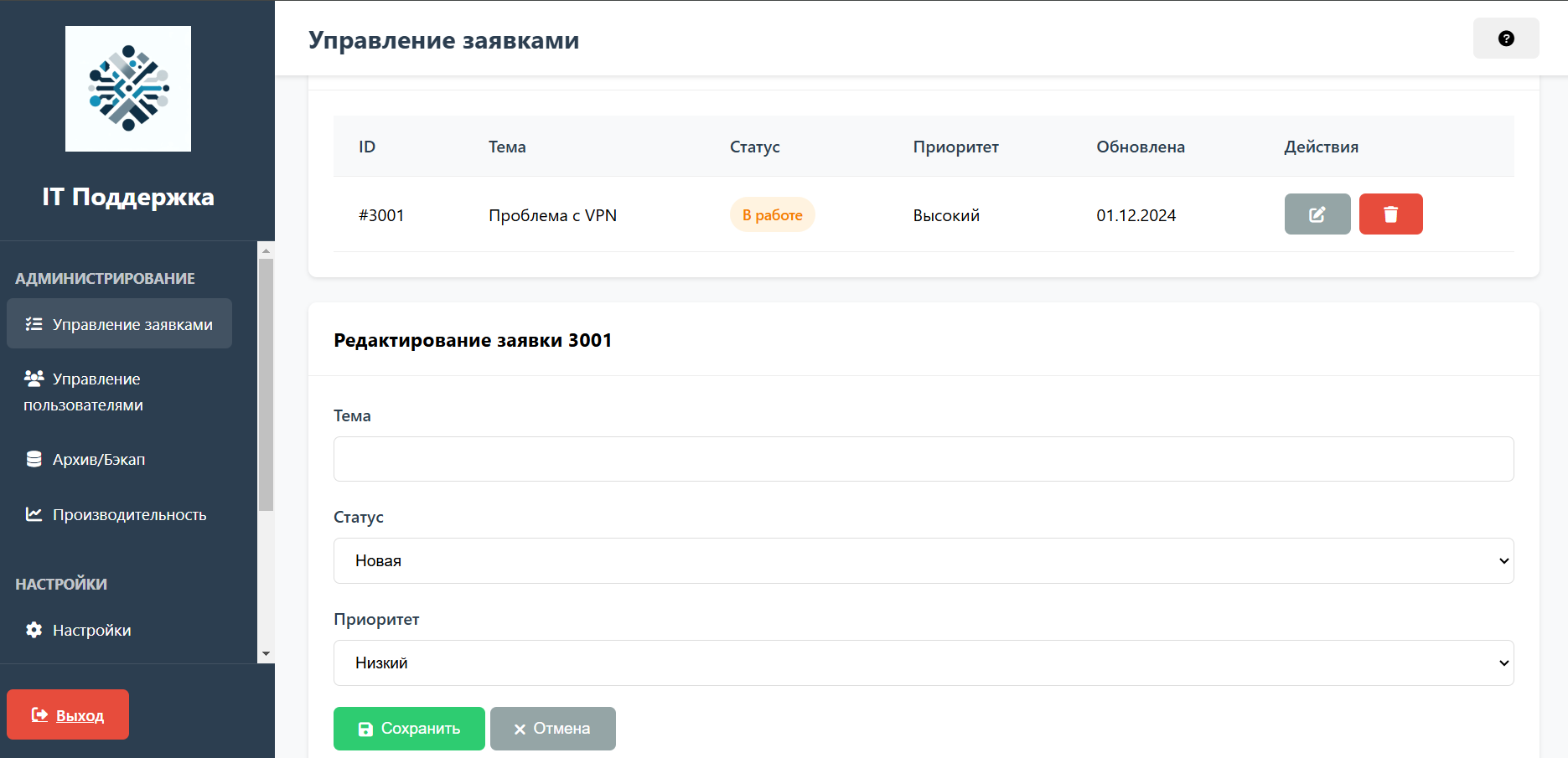


Рисунок № 15 – Редактирования заявки

Подтверждение удаления при нажатии на мусорку

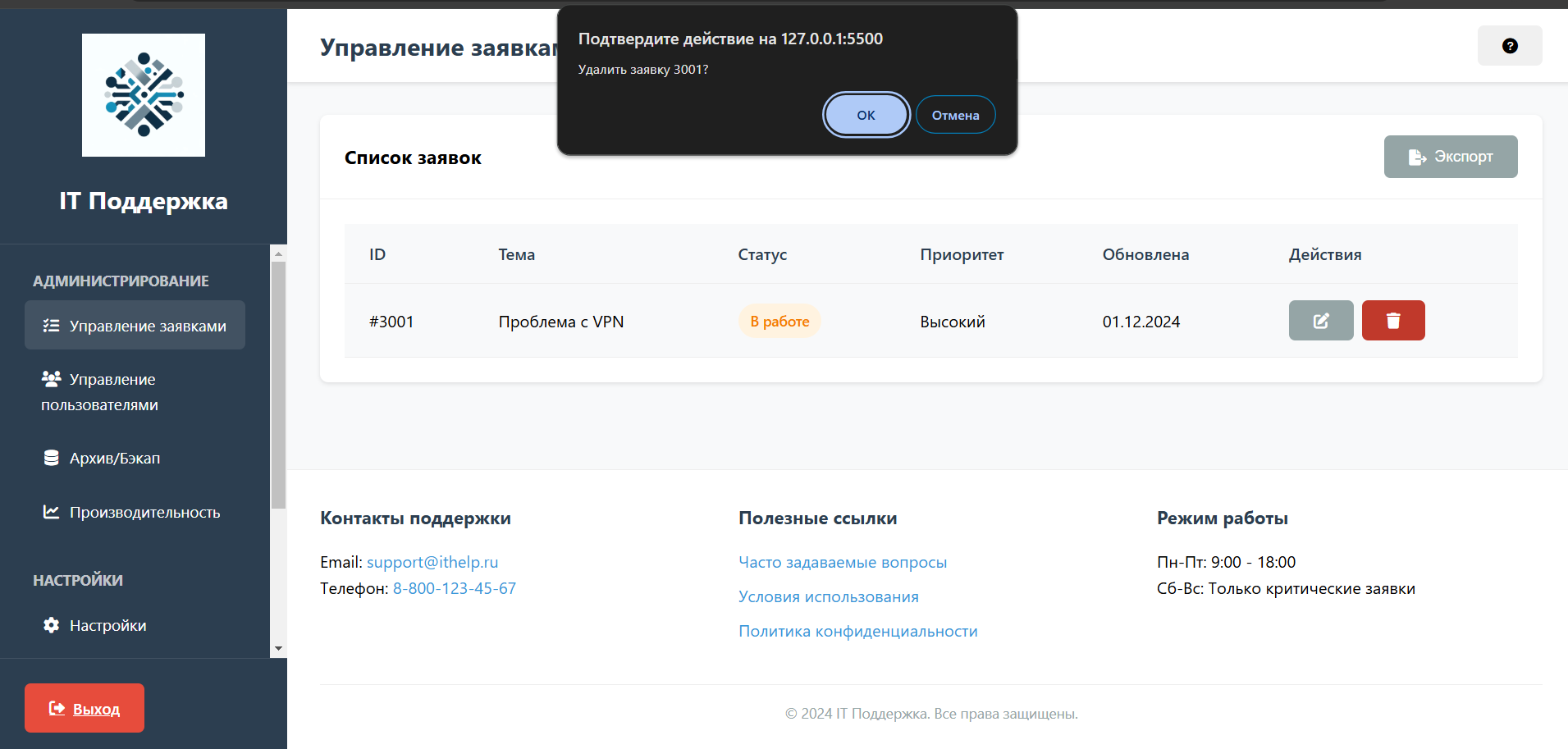


Рисунок № 16 – Подтверждение действия

При нажатии на кнопку экспорт появляется окно с тем, что производится экспорт



Рисунок № 17 – Экспорт

1. Окно управления пользователями

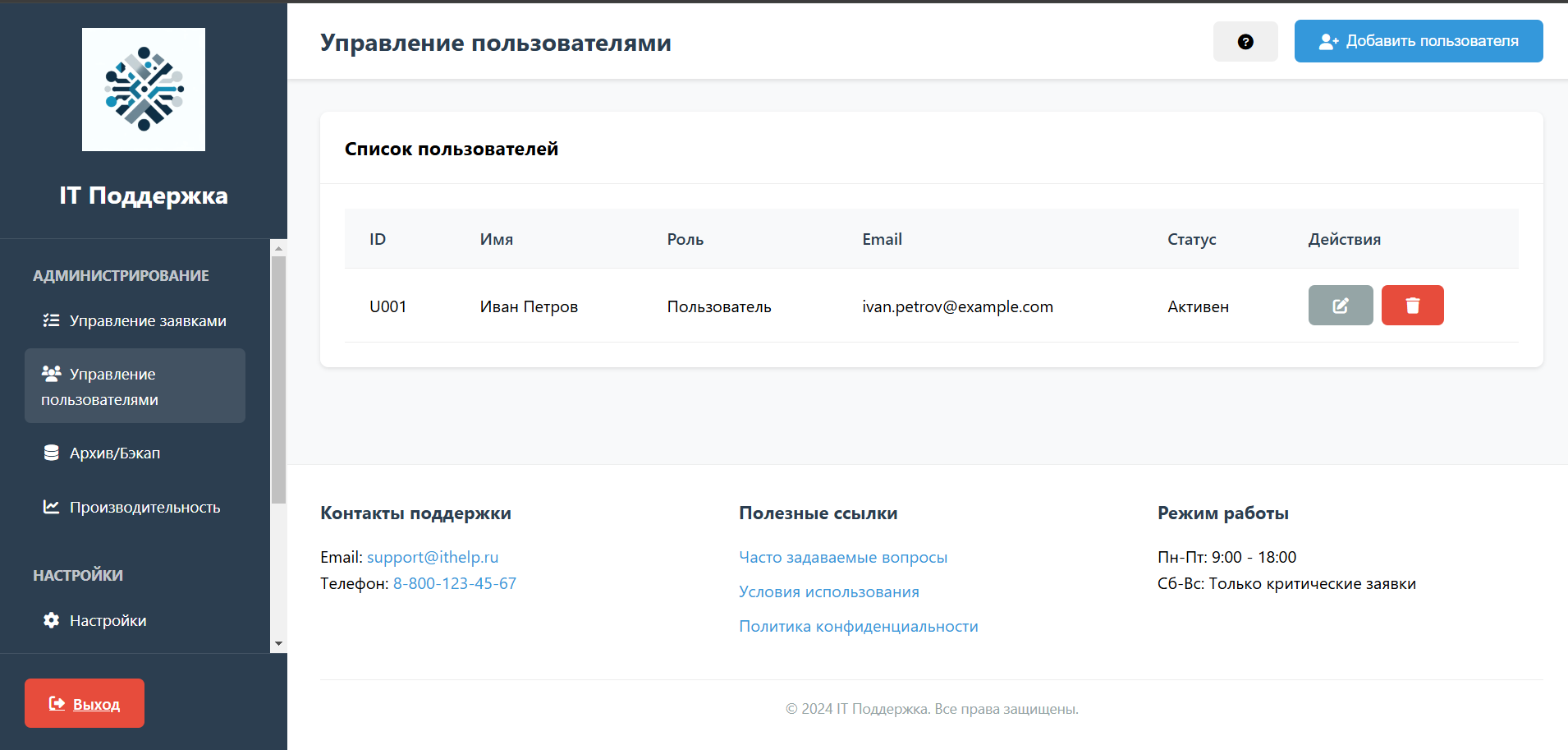


Рисунок № 18 – Окно управления пользователями

При нажатии на кнопку добавить пользователя появляется такая настройка

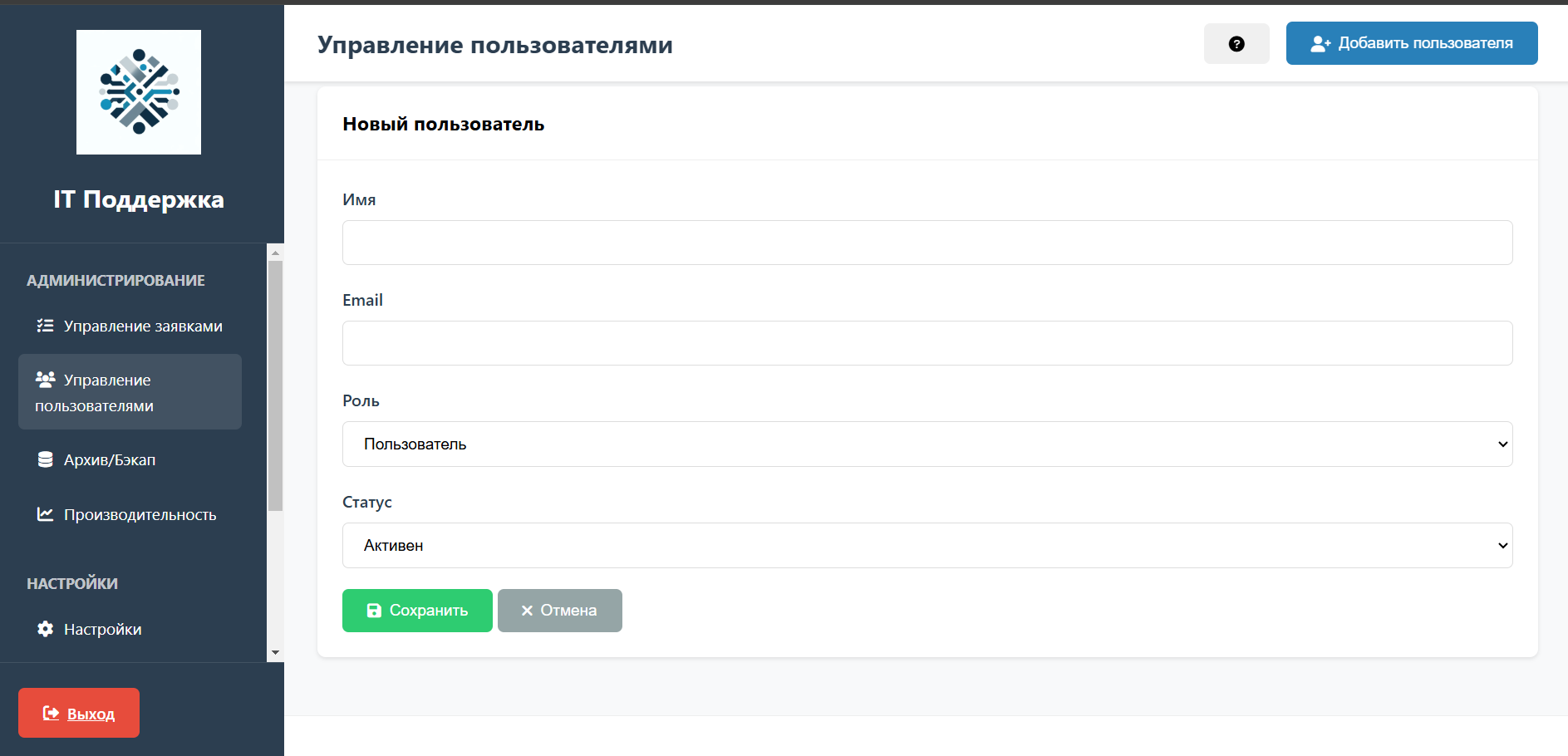
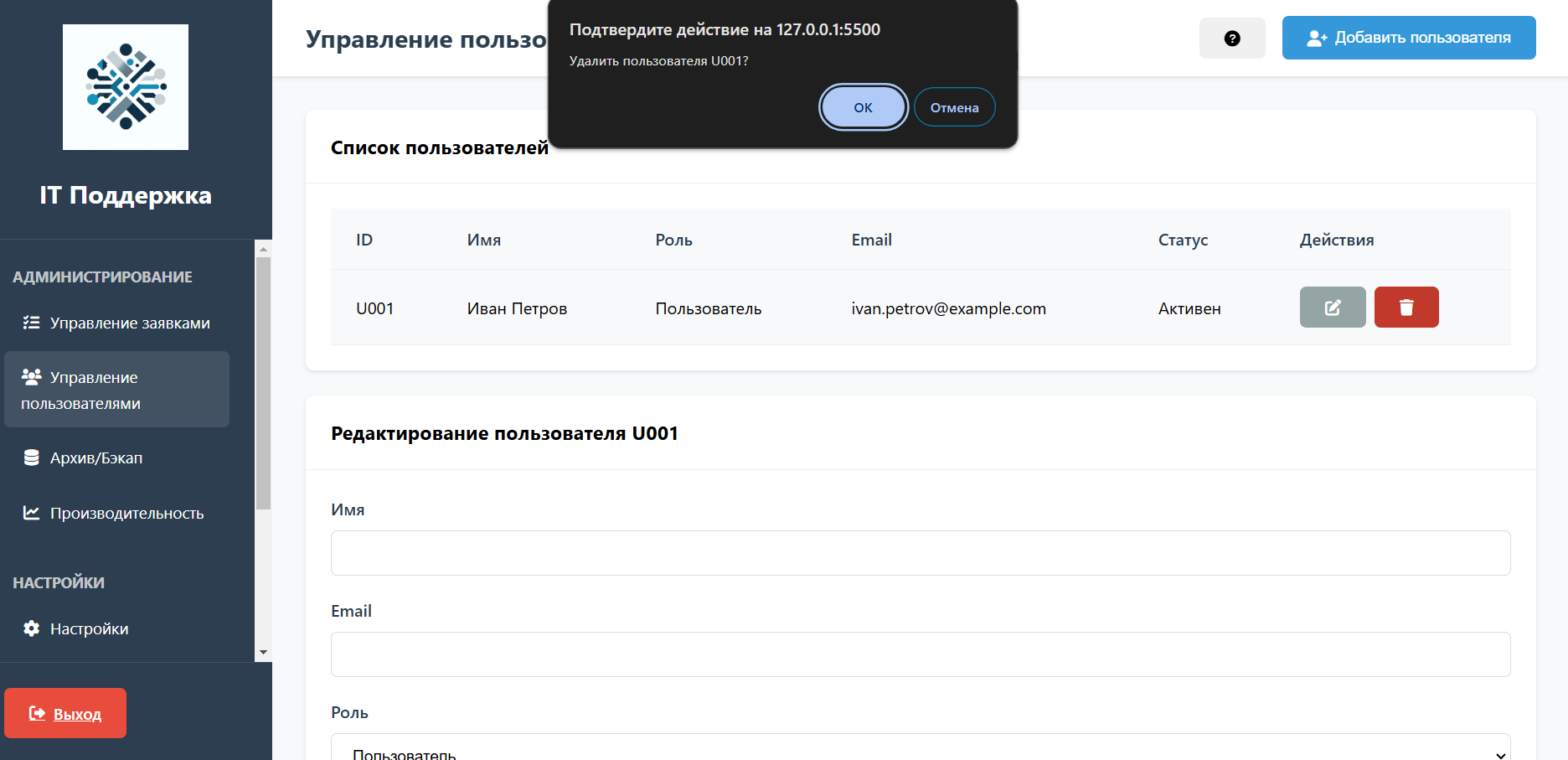


Рисунок № 19 – Создание нового пользователя

При удалении появляется окно подтверждения

  
Рисунок № 20 – Окно подтверждения

1. Архивирование и резервное копирования

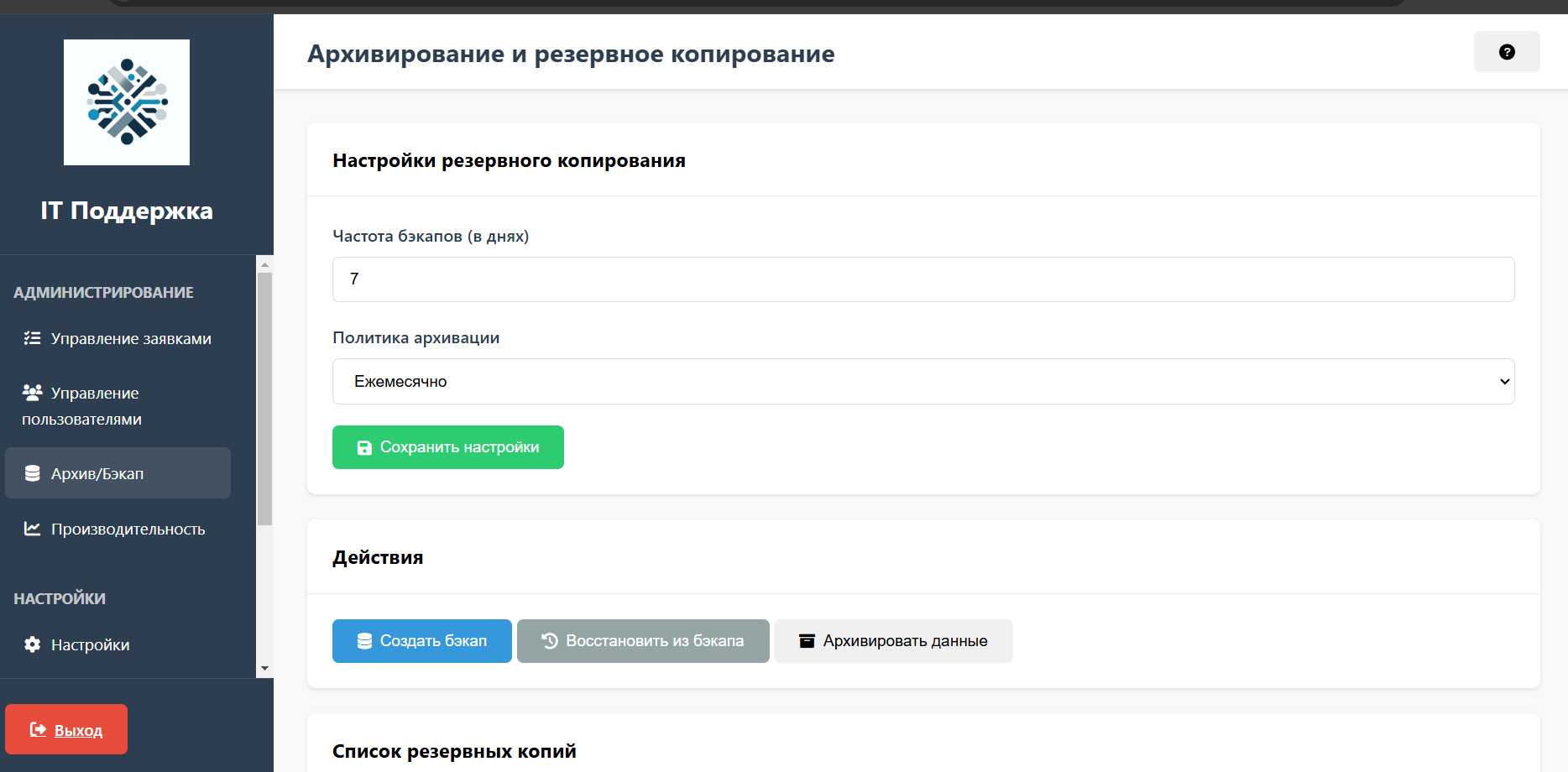


Рисунок № 21 – архивирование и резервное копирование

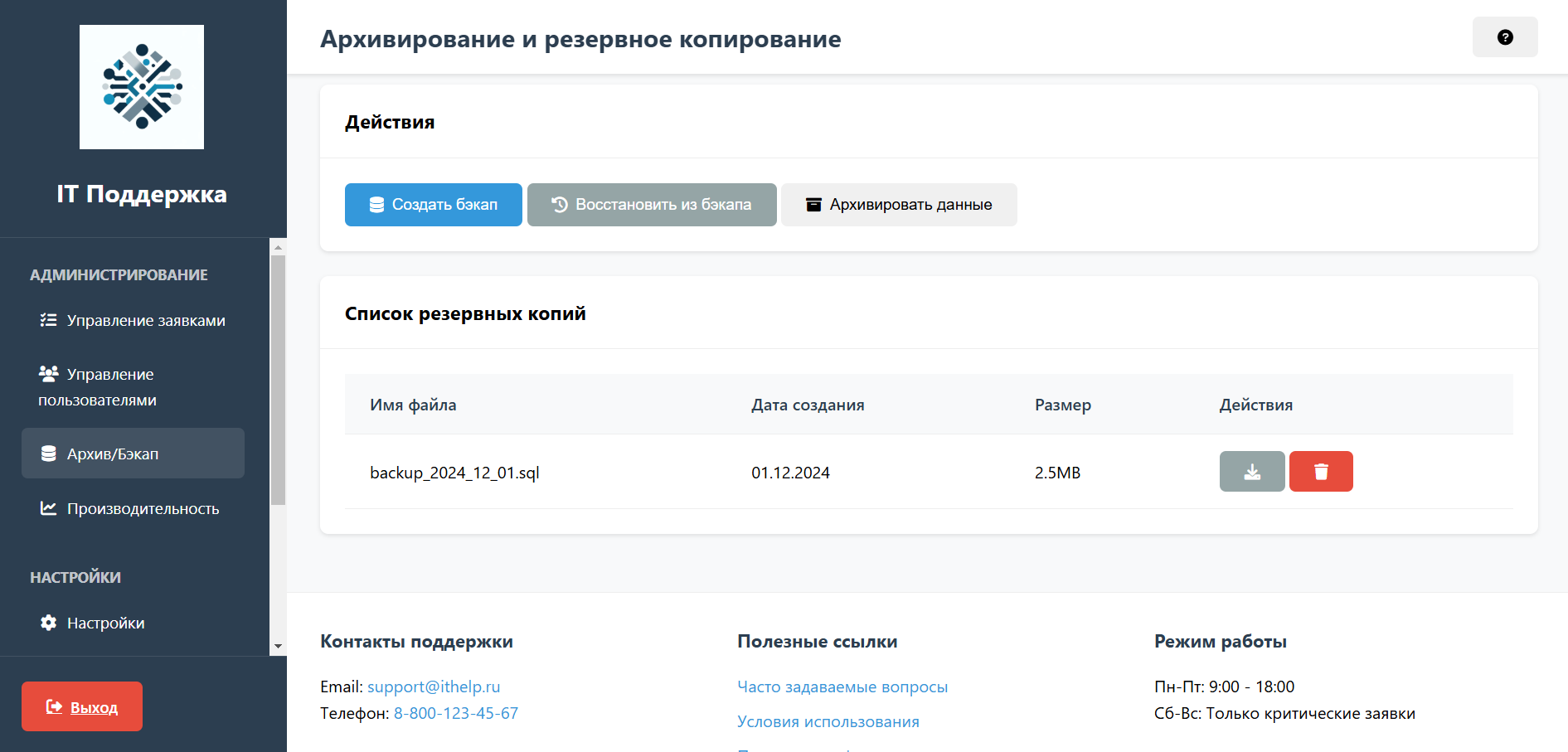


Рисунок № 22 – архивирование и резервной копирование

Все интерактивные кнопки

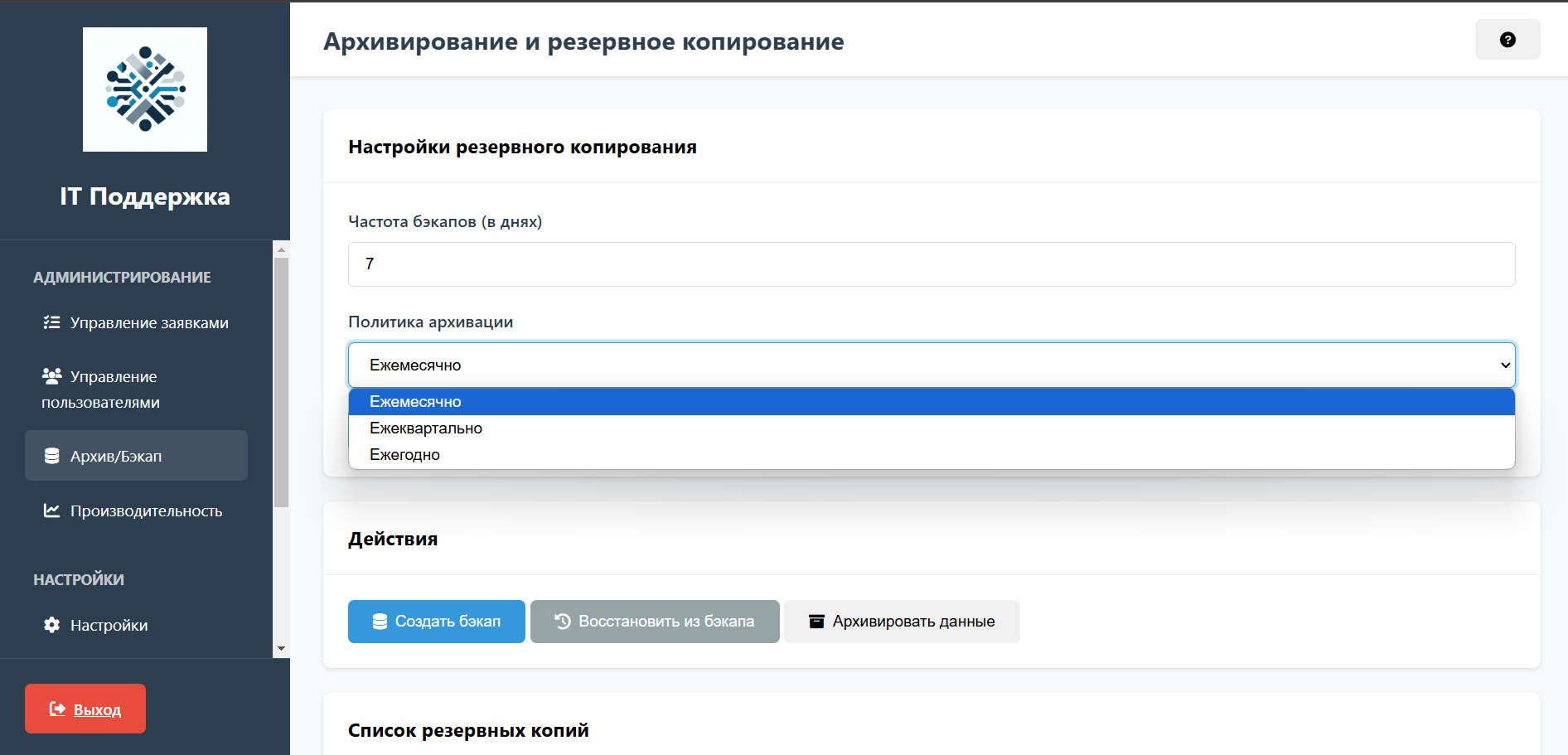


Рисунок № 23 – Список

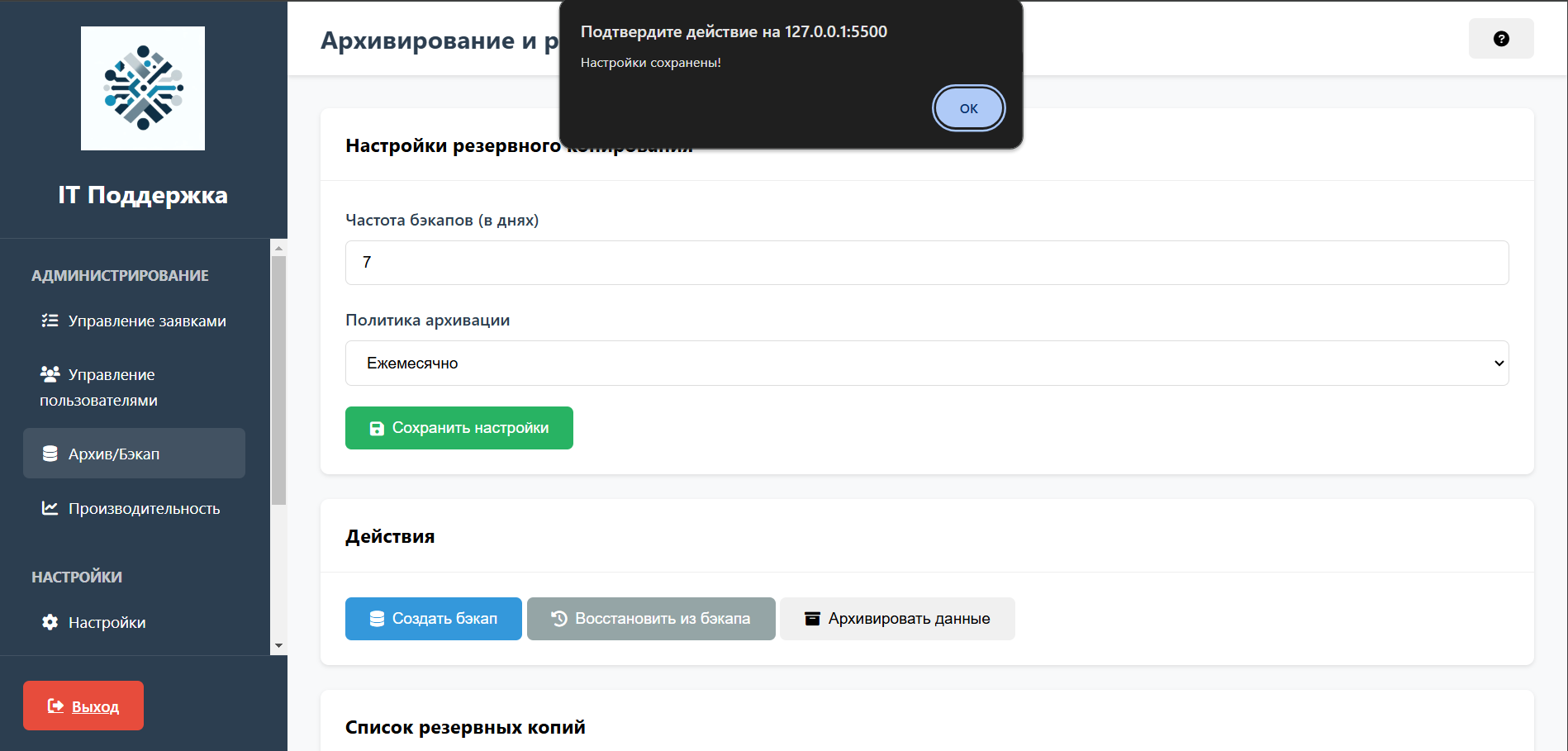


Рисунок № 24 – Настройки сохранены

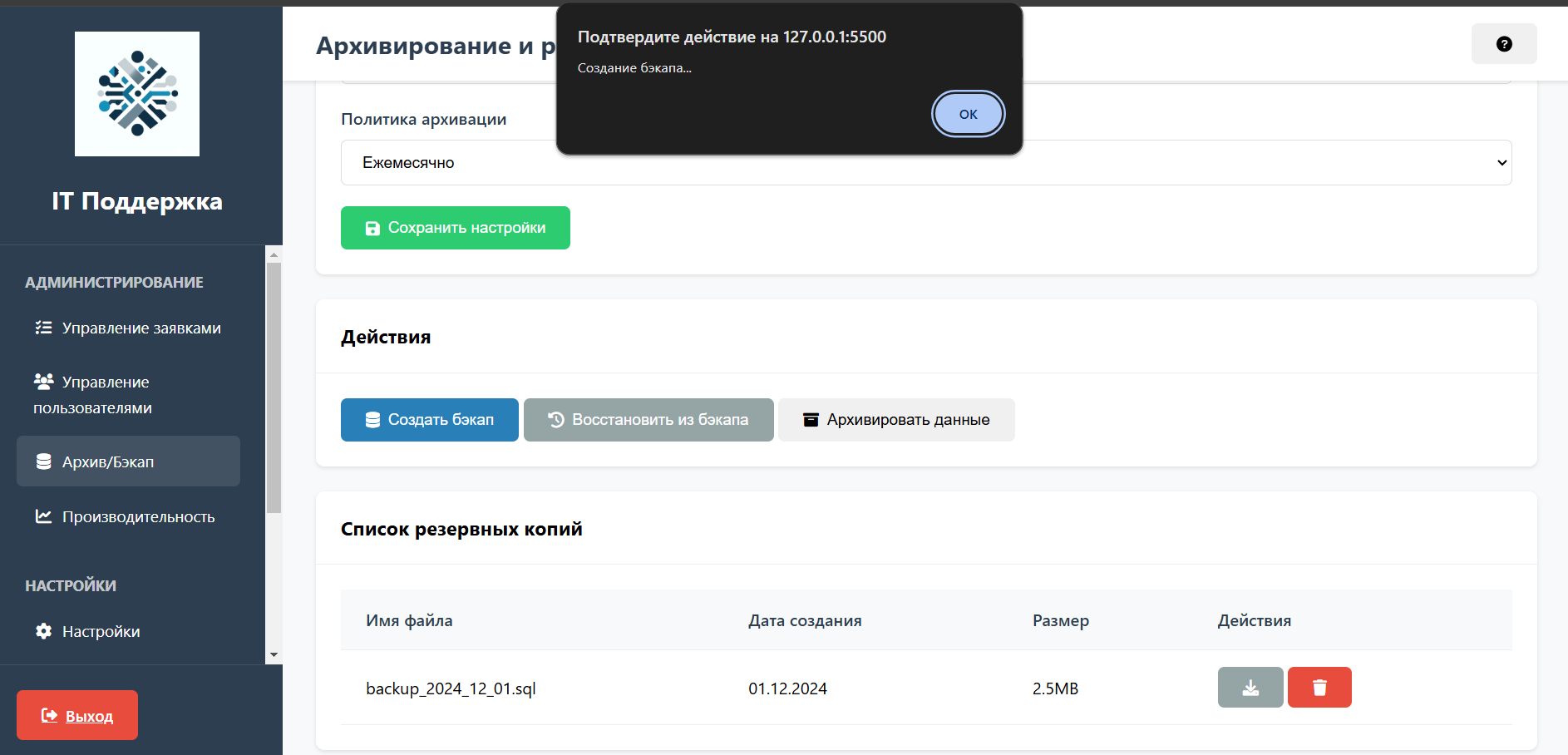


Рисунок № 25 – Создание бэкапа

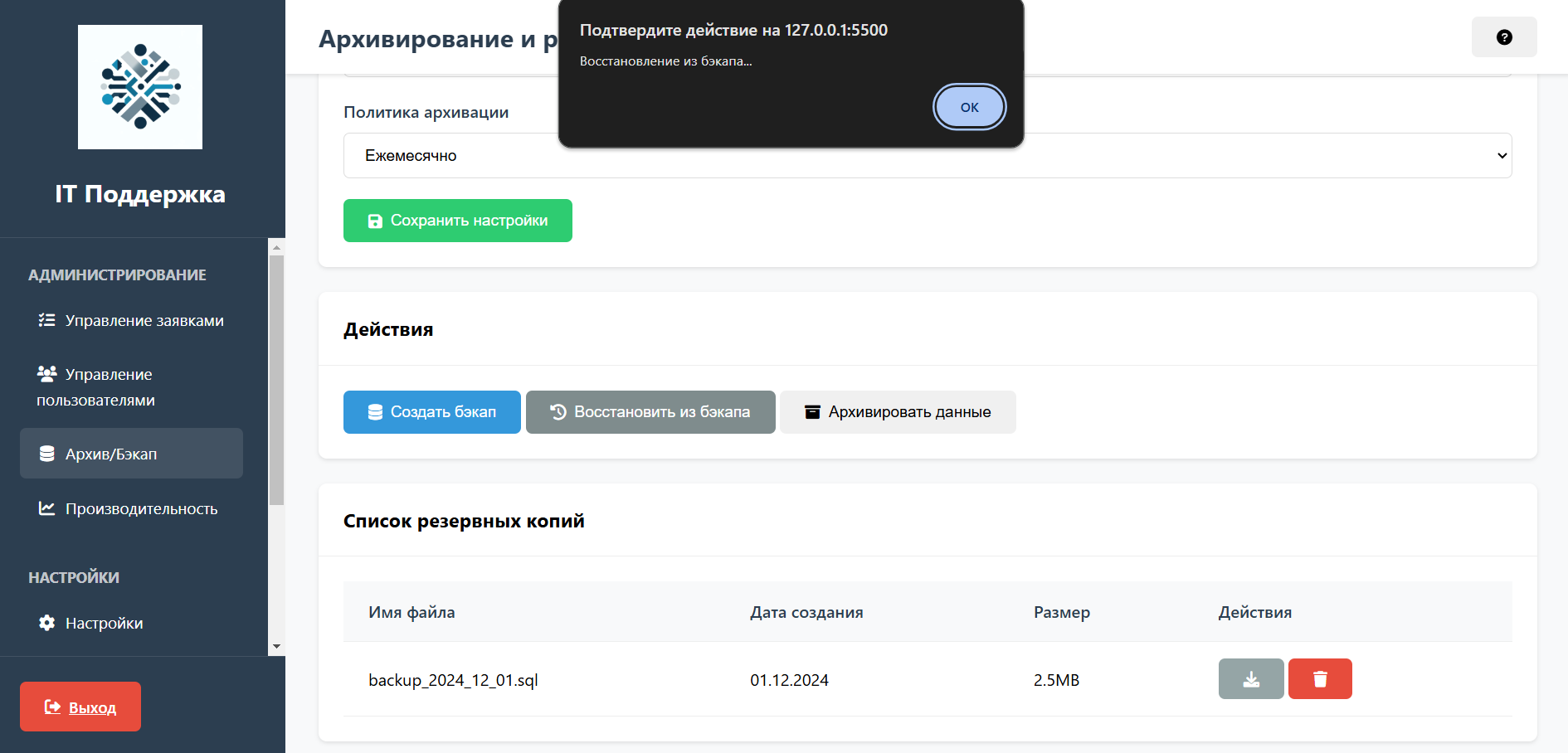


Рисунок № 26 – Восстановление из бэкапа

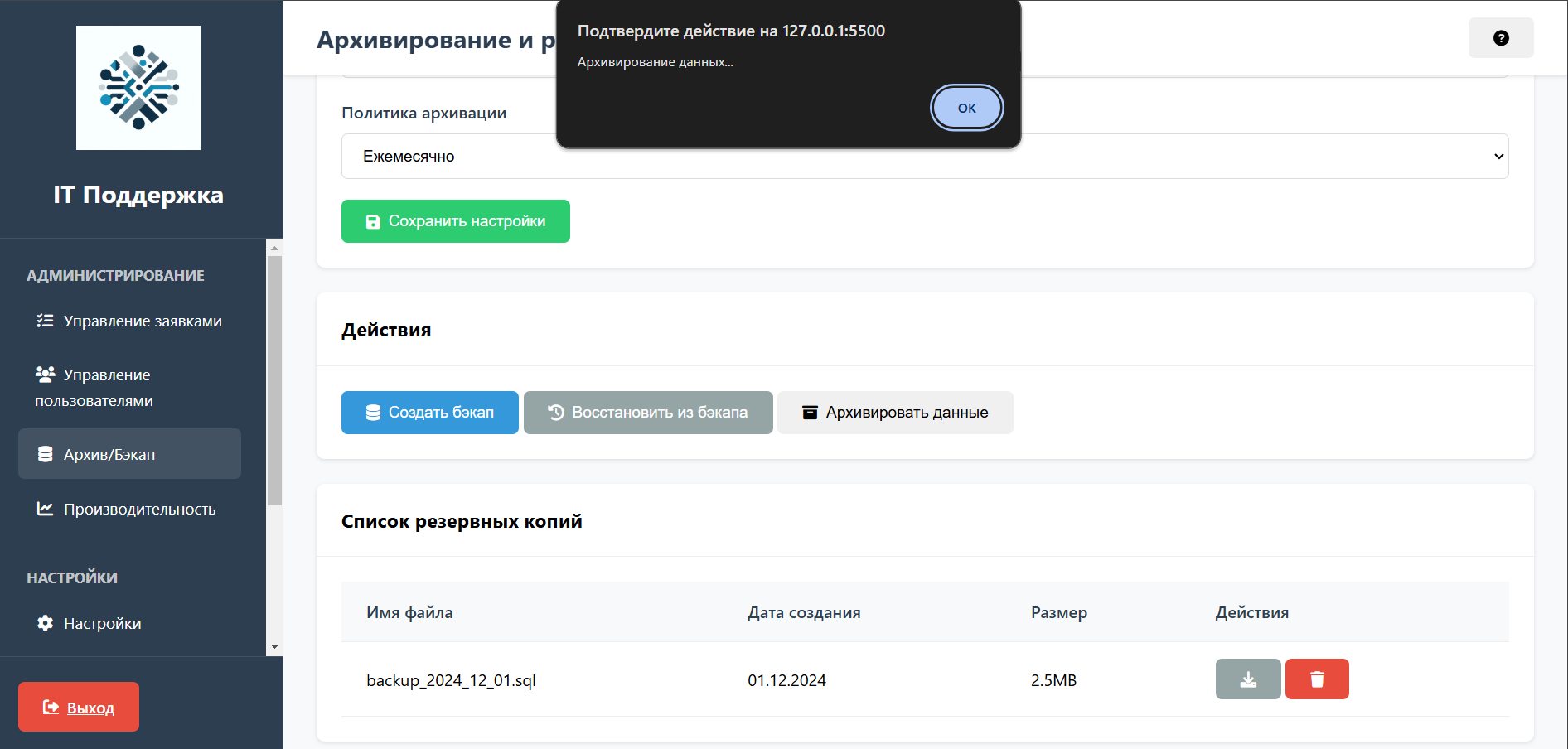


Рисунок № 27 – Архивирование данных

8. Производительность и характеристика бд

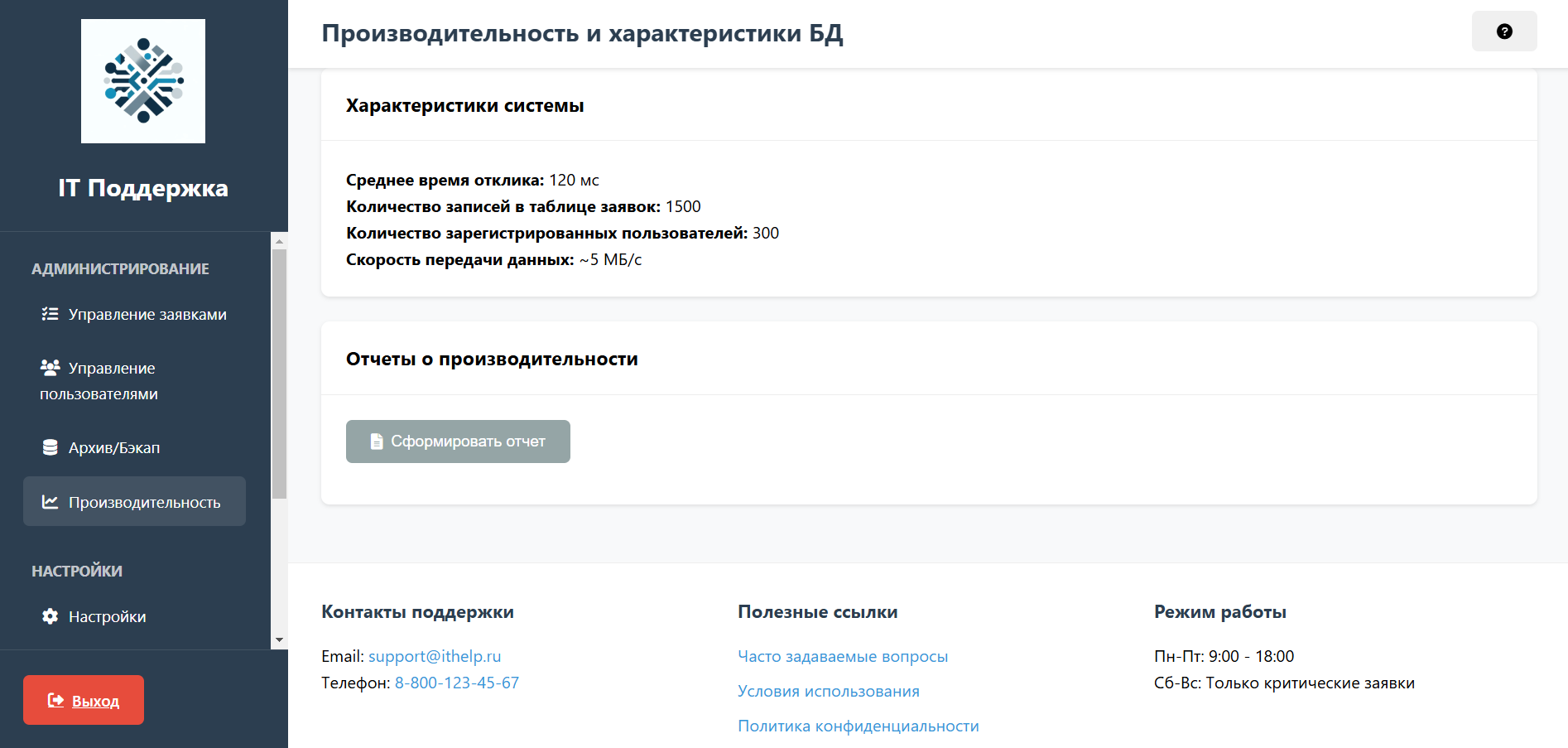


Рисунок № 28 – Производительность и характеристика бд

Формирование отчета

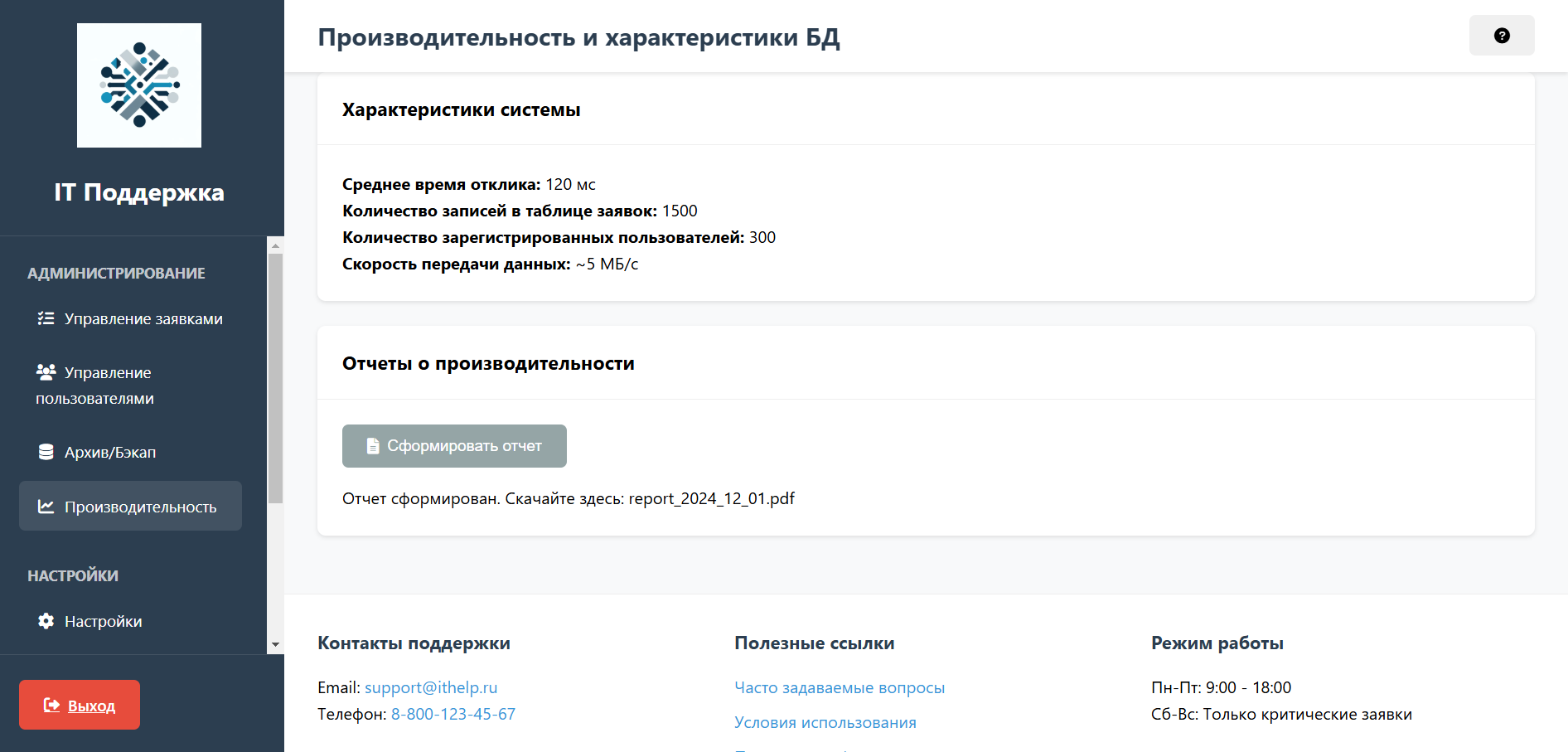


Рисунок № 29 – Формирование отчета

2.4.9. Просмотр очереди из заявок у оператора



Рисунок № 30 – Просмотр очереди

При нажатии на кнопку взять в работу появляется окно

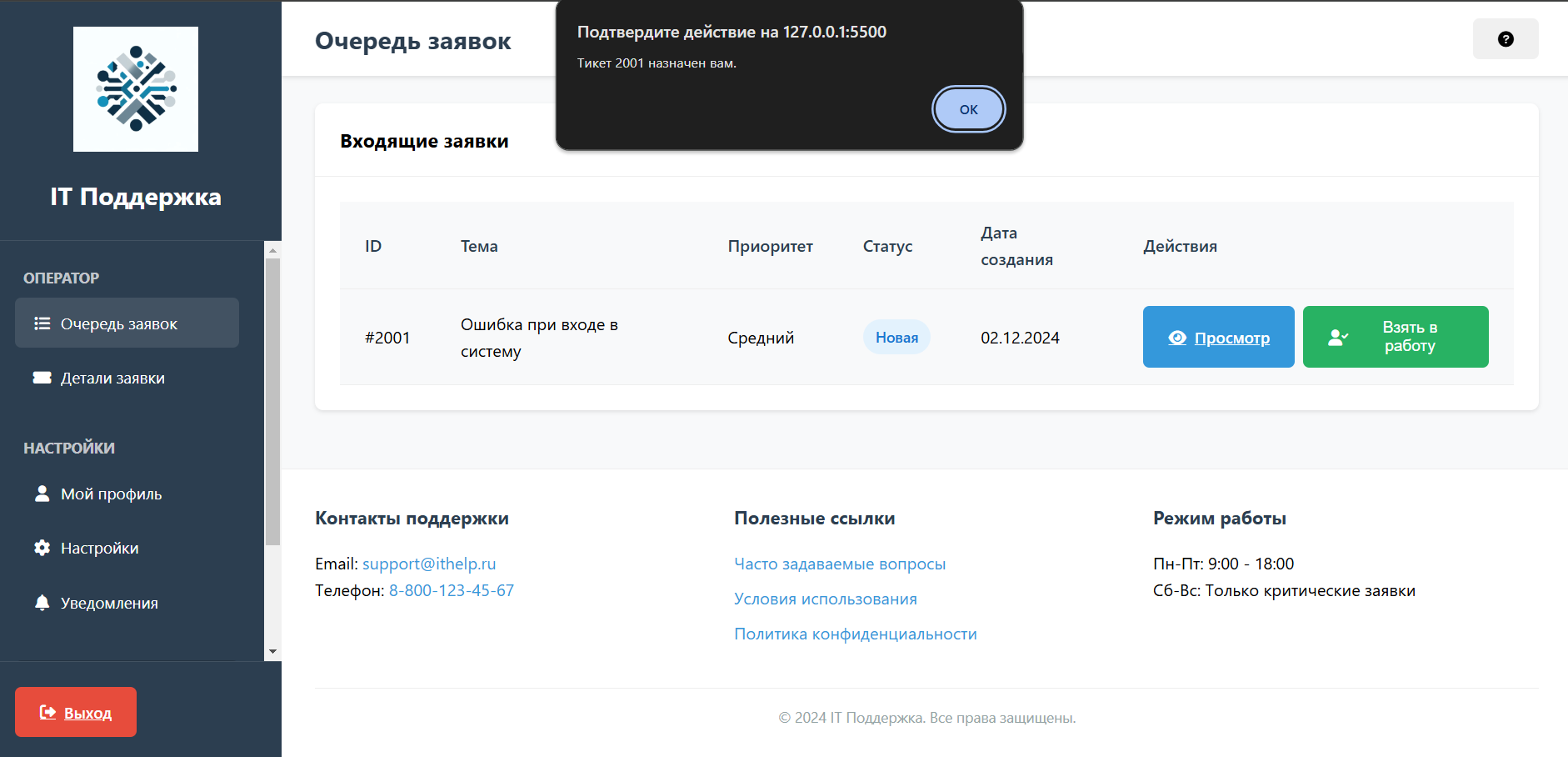


Рисунок № 31 – Подтверждение действия

2.4.10. При нажатии на кнопку просмотр появляется информация о заявке

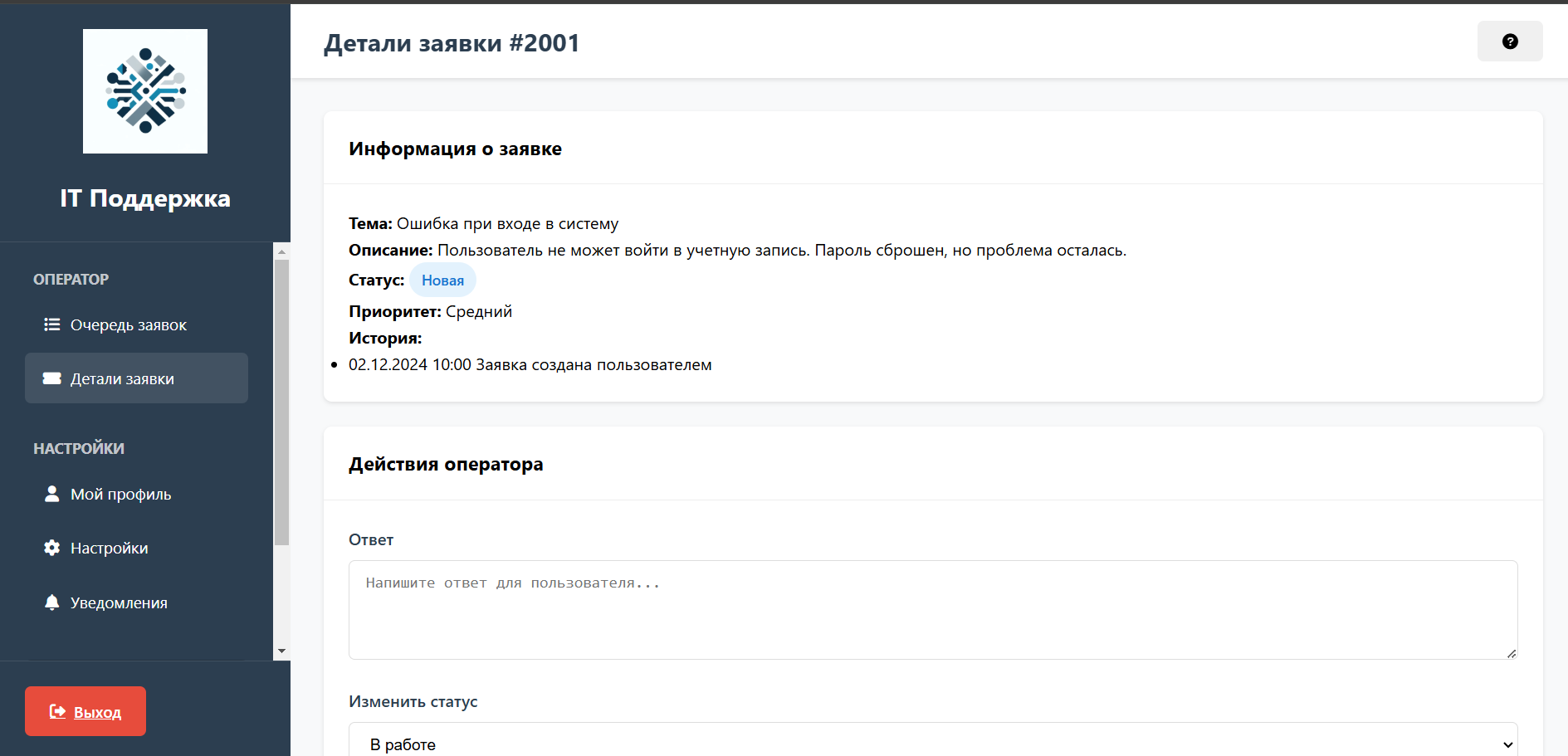


Рисунок № 32 – Детали заявки

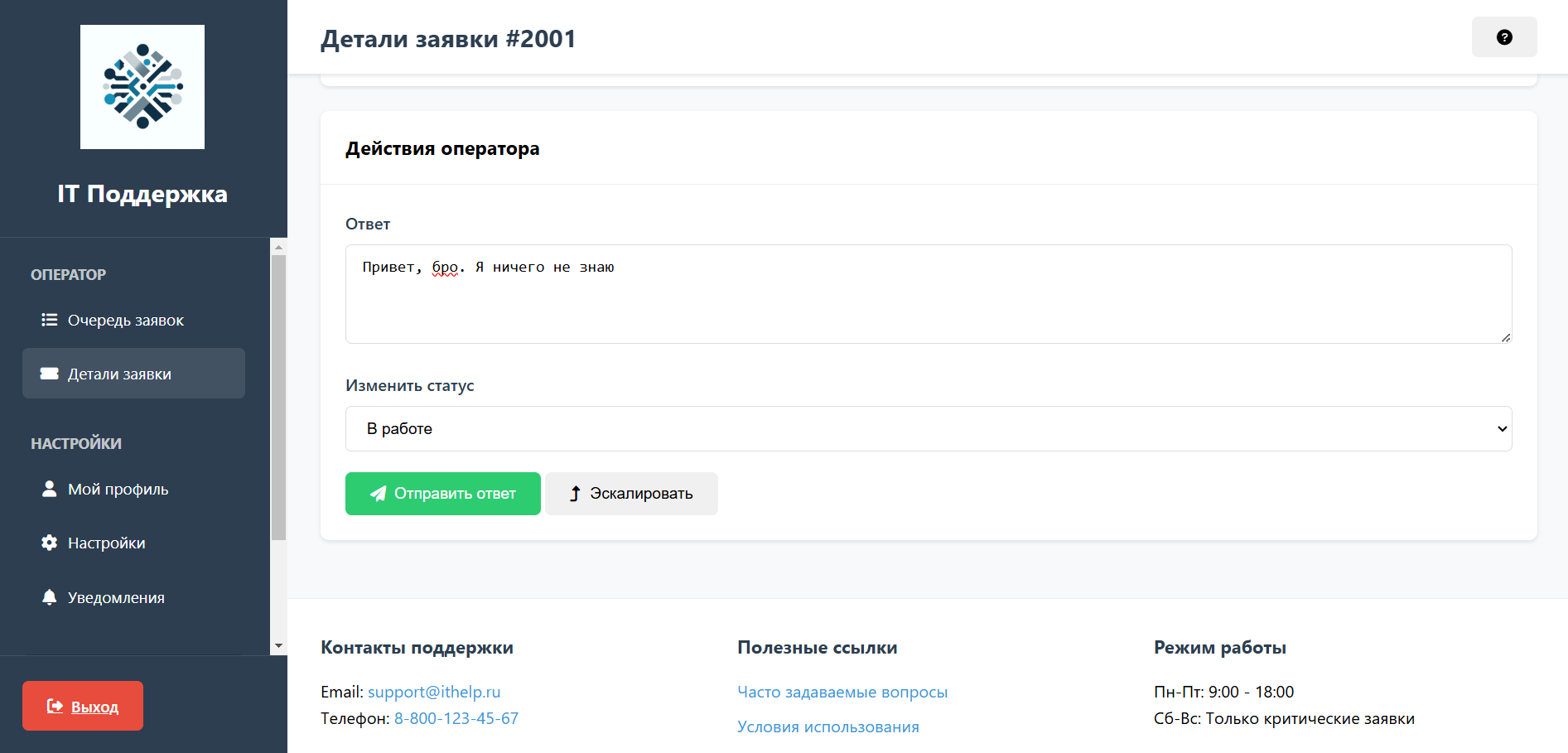


Рисунок № 33 – Детали заявки

3. Построить диаграмму классов.

1. Обычные классы

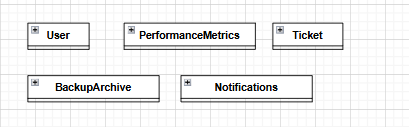


Рисунок 34 – Первый уровень диаграммы классов

Добавлены классы:

User: Отвечает за управление учетными данными пользователя, его ролью, статусом и взаимодействием с другими модулями системы.

PerformanceMetrics: Сбор и анализ данных о производительности системы, таких как время отклика, количество заявок и скорость передачи данных

Ticket: Обрабатывает создание, редактирование и закрытие заявок, а также обеспечивает доступ к истории изменений

BackupArchive: Обеспечивает создание резервных копий, восстановление данных и управление архивами системы

Notifications: Управляет отправкой уведомлений пользователям о событиях и обновлениях в системе

2. Классы тильды

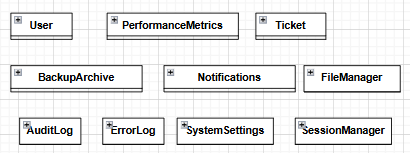


Рисунок 35 – Второй уровень диаграммы классов

Добавлены классы:

AuditLog: Класс для отслеживания изменений и действий пользователей в системе.

ErrorLog: Класс для записи и хранения ошибок, возникающих в процессе работы системы.

SystemSettings: Класс для управления глобальными настройками системы.

SessionManager: Класс для управления пользовательскими сессиями в системе.

FileManager: Класс для управления файлами, загруженными пользователями или системой.

1. Поля (зоны видимости)

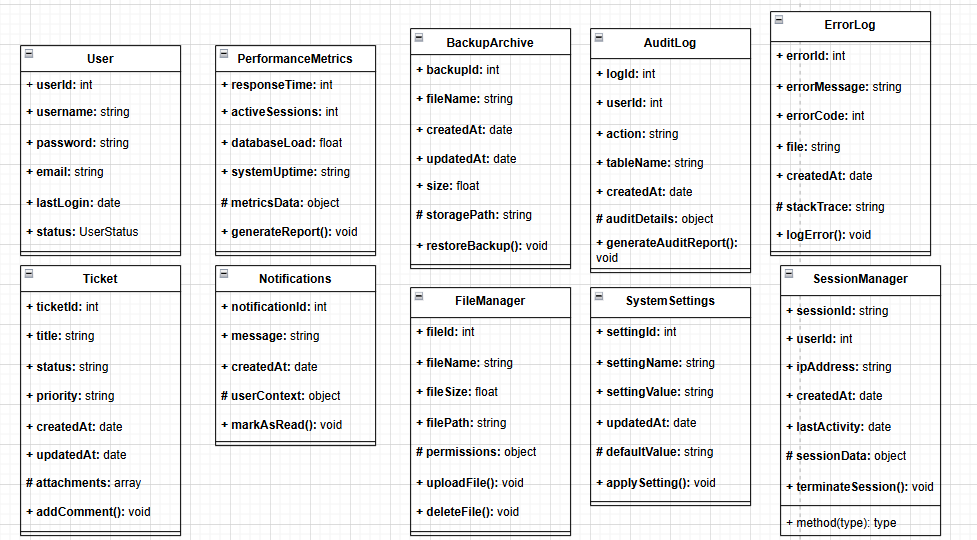


Рисунок 36 – Зоны видимости

Классы:

User (Пользователь):

+ userId: int — идентификатор пользователя

+ username: string — имя пользователя

+ password: string — пароль пользователя

+ email: string — электронная почта

+ lastLogin: date — дата последнего входа

+ status: UserStatus — статус пользователя

PerformanceMetrics (Метрики производительности)

+ responseTime: int — время отклика системы (мс)

+ activeSessions: int — количество активных сессий

+ databaseLoad: float — нагрузка на базу данных (%)

+ systemUptime: string — время непрерывной работы системы

# metricsData: object — детализированные данные метрик (protected)

+ generateReport(): void — метод генерации отчёта

Ticket (Заявка)

+ ticketId: int — идентификатор заявки

+ title: string — заголовок заявки

+ status: string — статус заявки

+ priority: string — приоритет заявки

+ createdAt: date — дата создания

+ updatedAt: date — дата последнего обновления

# attachments: array — вложения (protected)

+ addComment(): void — метод добавления комментария

BackupArchive (Архив/Бэкап)

+ backupId: int — идентификатор резервной копии

+ fileName: string — имя файла резервной копии

+ createdAt: date — дата создания копии

+ size: float — размер файла в МБ

# storagePath: string — путь к хранилищу (protected)

+ restoreBackup(): void — метод восстановления данных

Notifications (Уведомления)

+ notificationId: int — идентификатор уведомления

+ message: string — текст уведомления

+ status: string — статус (прочитано/не прочитано)

+ createdAt: date — дата создания уведомления

# userContext: object — контекст пользователя (protected)

+ markAsRead(): void — отметить как прочитанное

FileManager (Менеджер файлов)

+ fileId: int — идентификатор файла

+ fileName: string — имя файла

+ fileSize: float — размер файла

+ filePath: string — путь к файлу

# permissions: object — права доступа (protected)

+ uploadFile(): void — метод загрузки файла

+ deleteFile(): void — метод удаления файла

AuditLog (Журнал аудита)

+ logId: int — идентификатор записи аудита

+ userId: int — идентификатор пользователя

+ action: string — выполненное действие

+ tableName: string — таблица, в которой произведено действие

+ recordId: int — идентификатор записи

+ createdAt: date — дата и время действия

# auditDetails: object — детали аудита (protected)

+ generateAuditReport(): void — метод генерации отчёта

ErrorLog (Журнал ошибок)

+ errorId: int — идентификатор ошибки

+ errorMessage: string — сообщение об ошибке

+ errorCode: int — код ошибки

+ file: string — файл, в котором возникла ошибка

+ line: int — строка кода с ошибкой

+ createdAt: date — дата и время ошибки

# stackTrace: string — трассировка стека (protected)

+ logError(): void — метод записи ошибки в журнал

SystemSettings (Системные настройки)

+ settingId: int — идентификатор настройки

+ settingName: string — имя настройки

+ settingValue: string — значение настройки

+ updatedAt: date — дата последнего обновления

# defaultValue: string — значение по умолчанию (protected)

+ applySetting(): void — применить настройку

SessionManager (Менеджер сессий)

+ sessionId: string — идентификатор сессии

+ userId: int — идентификатор пользователя

+ ipAddress: string — IP-адрес пользователя

+ createdAt: date — дата начала сессии

+ lastActivity: date — дата последней активности

# sessionData: object — данные сессии (protected)

+ terminateSession(): void — завершить сессию

1. Методы с атрибутами

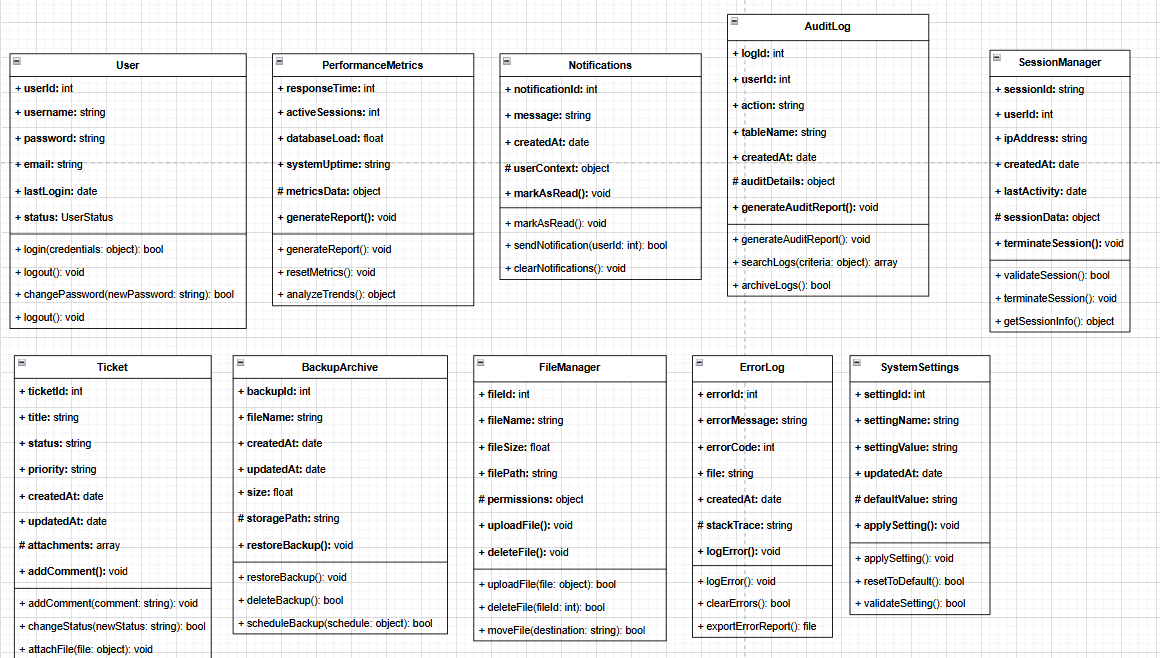


Рисунок № 37 – Методы с атрибутами

User (Пользователь):

+ login(credentials: object): bool — выполняет вход пользователя

+ logout(): void — выполняет выход пользователя

+ changePassword(newPassword: string): bool — меняет пароль пользователя

+ updateProfile(data: object): bool — обновляет профиль пользователя

PerformanceMetrics (Метрики производительности):

+ generateReport(): void — генерирует отчёт о производительности

+ resetMetrics(): void — сбрасывает метрики

+ analyzeTrends(): object — анализирует тренды метрик

Ticket (Заявка):

+ addComment(comment: string): void — добавляет комментарий к заявке

+ changeStatus(newStatus: string): bool — изменяет статус заявки

+ attachFile(file: object): void — прикрепляет файл к заявке

BackupArchive (Архив/Бэкап):

+ restoreBackup(): void — восстанавливает данные из резервной копии

+ deleteBackup(): bool — удаляет резервную копию

+ scheduleBackup(schedule: object): bool — планирует резервное копирование

Notifications (Уведомления):

+ markAsRead(): void — отмечает уведомление как прочитанное

+ sendNotification(userId: int): bool — отправляет уведомление пользователю

+ clearNotifications(): void — очищает уведомления

FileManager (Менеджер файлов):

+ uploadFile(file: object): bool — загружает файл

+ deleteFile(fileId: int): bool — удаляет файл

+ moveFile(destination: string): bool — перемещает файл

AuditLog (Журнал аудита):

+ generateAuditReport(): void — генерирует отчёт об аудите

+ searchLogs(criteria: object): array — ищет записи аудита

+ archiveLogs(): bool — архивирует старые записи аудита

ErrorLog (Журнал ошибок):

+ logError(): void — записывает ошибку в журнал

+ clearErrors(): bool — очищает ошибки

+ exportErrorReport(): file — экспортирует отчёт об ошибках

SystemSettings (Системные настройки):

+ applySetting(): void — применяет настройку

+ resetToDefault(): bool — сбрасывает настройку к значению по умолчанию

+ validateSetting(): bool — проверяет корректность настройки

SessionManager (Менеджер сессий):

+ validateSession(): bool — проверяет валидность сессии

+ terminateSession(): void — завершает сессию

+ getSessionInfo(): object — получает информацию о сессии

1. Отношения между атрибутами

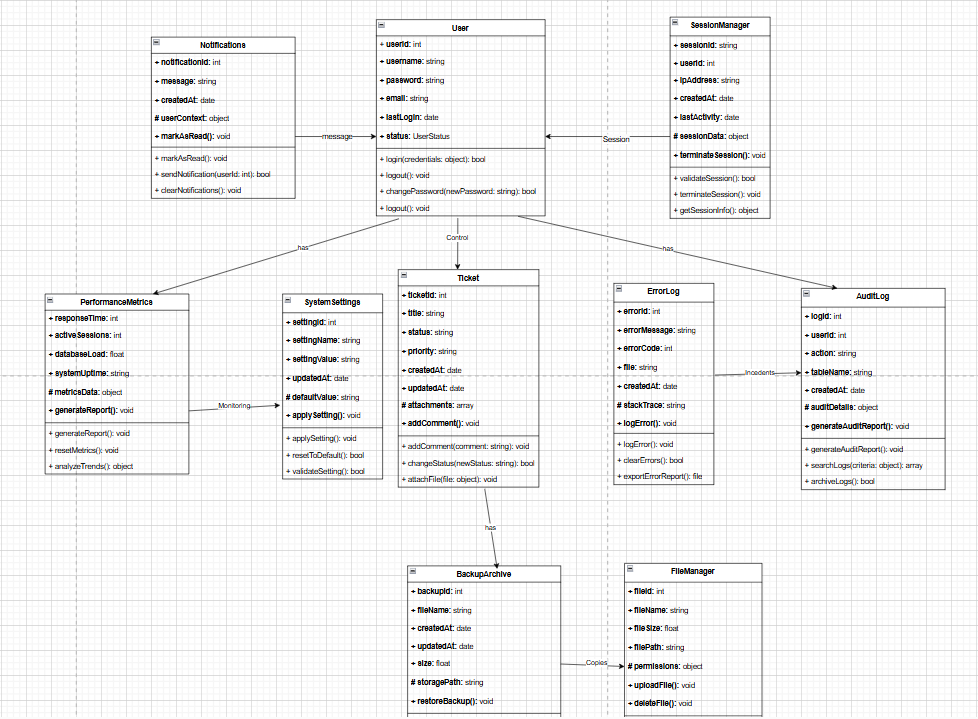


Рисунок № 38 – Отношения между атрибутами

Связи:

User --> Notifications: ассоциация, пользователь получает уведомления через Notifications.

User --> SessionManager: ассоциация, пользователь управляет сессией через SessionManager.

User --> Ticket: ассоциация, пользователь может иметь несколько заявок (Ticket).

Ticket --> BackupArchive: ассоциация, заявки могут быть сохранены в резервных копиях через BackupArchive.

Ticket --> FileManager: ассоциация, заявки могут иметь файлы, управляемые через FileManager.

Ticket --> ErrorLog: ассоциация, ошибки, связанные с заявками, фиксируются в ErrorLog.

ErrorLog --> AuditLog: ассоциация, ошибки регистрируются в журнале аудита AuditLog.

PerformanceMetrics --> SystemSettings: ассоциация, метрики производительности контролируются настройками системы через SystemSettings.

SystemSettings --> Ticket: ассоциация, системные настройки влияют на управление заявками.

AuditLog --> Ticket: ассоциация, журнал аудита фиксирует все действия, связанные с заявками.

SessionManager --> User: ассоциация, менеджер сессий управляет активными пользователями (User).

User --> PerformanceMetrics: ассоциация, пользователь взаимодействует с метриками производительности (PerformanceMetrics).

Notifications --> User: ассоциация, уведомления отправляются пользователю (User).

SessionManager --> Ticket: ассоциация, сессии могут контролировать доступ к заявкам (Ticket).

BackupArchive --> FileManager: ассоциация, файлы резервных копий управляются через FileManager.

ErrorLog --> Ticket: ассоциация, ошибки фиксируются в связанных заявках (Ticket).

AuditLog --> SessionManager: ассоциация, аудит включает информацию о сессиях (SessionManager).

SystemSettings --> AuditLog: ассоциация, настройки системы влияют на параметры аудита (AuditLog).

FileManager --> SystemSettings: ассоциация, менеджер файлов зависит от системных настроек (SystemSettings).

PerformanceMetrics --> AuditLog: ассоциация, метрики фиксируют ключевые параметры в аудите (AuditLog).

1. Виджеты

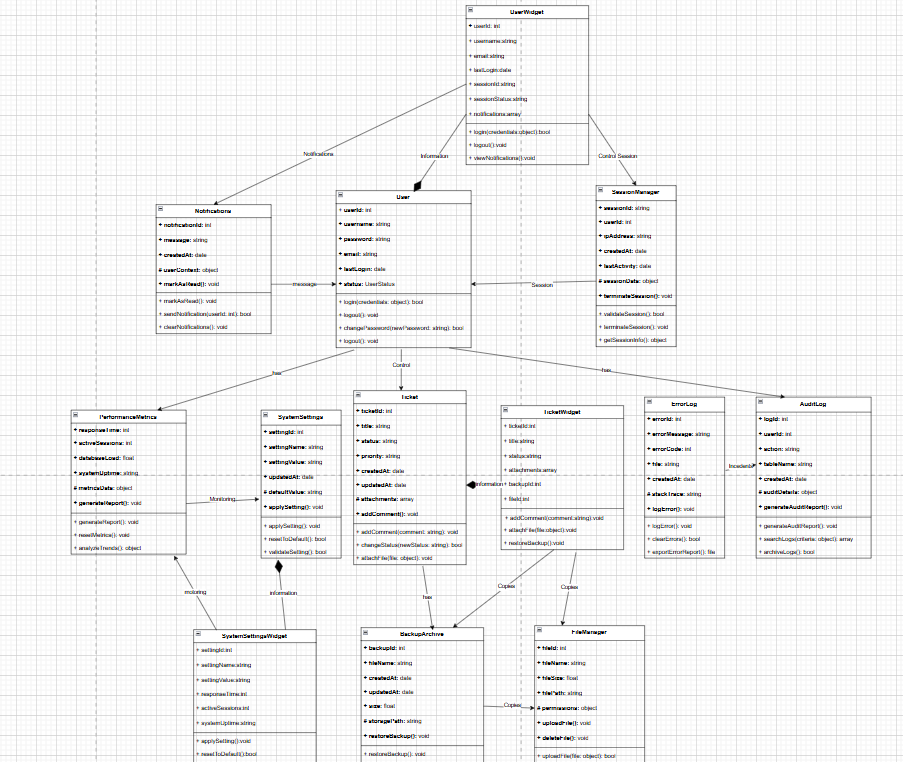


Рисунок № 39 – Виджеты

Добавлены виджеты:

UserWidget:

userId:int - Идентификатор пользователя

username:string - Имя пользователя

email:string- Электронная почта

lastLogin:date - Дата последнего входа

sessionId:string - Идентификатор сессии

sessionStatus:string - Статус сессии

notifications:array - Список уведомлений

Методы:

login(credentials:object):bool - Выполняет вход пользователя

logout():void - Завершает сессию пользователя

viewNotifications():void - Отображает уведомления

validateSession():bool - Проверяет валидность сессии

Отношения:

UserWidget → User: Управляет информацией о пользователе.

UserWidget → SessionManager: Управляет пользовательскими сессиями.

UserWidget → Notifications: Отображает уведомления пользователя.

TicketWidget:

ticketId:int - Идентификатор заявки

title:string - Заголовок заявки

status:string - Статус заявки

priority:string - Приоритет заявки

attachments:array - Вложения заявки

backupId:int - Идентификатор бэкапа

fileId:int - Идентификатор файла

Методы:

addComment(comment:string):void - Добавляет комментарий к заявке

attachFile(file:object):void - Прикрепляет файл к заявке

restoreBackup():void - Восстанавливает данные из бэкапа

viewBackupDetails():object - Просматривает детали бэкапа

Отношения:

TicketWidget - Ticket: Управляет заявками.

TicketWidget - BackupArchive: Управляет резервным копированием заявок.

TicketWidget - FileManager: Управляет файлами, связанными с заявками.

SystemSettingsWidget:

settingId:int - Идентификатор настройки

settingName:string - Название настройки

settingValue:string - Значение настройки

responseTime:int - Время отклика системы

activeSessions:int - Количество активных сессий

systemUptime:string - Время работы системы

Методы:

applySetting():void - Применяет настройки

resetToDefault():bool - Сбрасывает настройки

monitorPerformance():void - Отображает метрики

viewSystemStatus():string - Просматривает статус системы

Отношения:

SystemSettingsWidget - SystemSettings: Управляет настройками системы.

SystemSettingsWidget - PerformanceMetrics: Отображает и управляет метриками производительности.

1. Отношения к виджетам

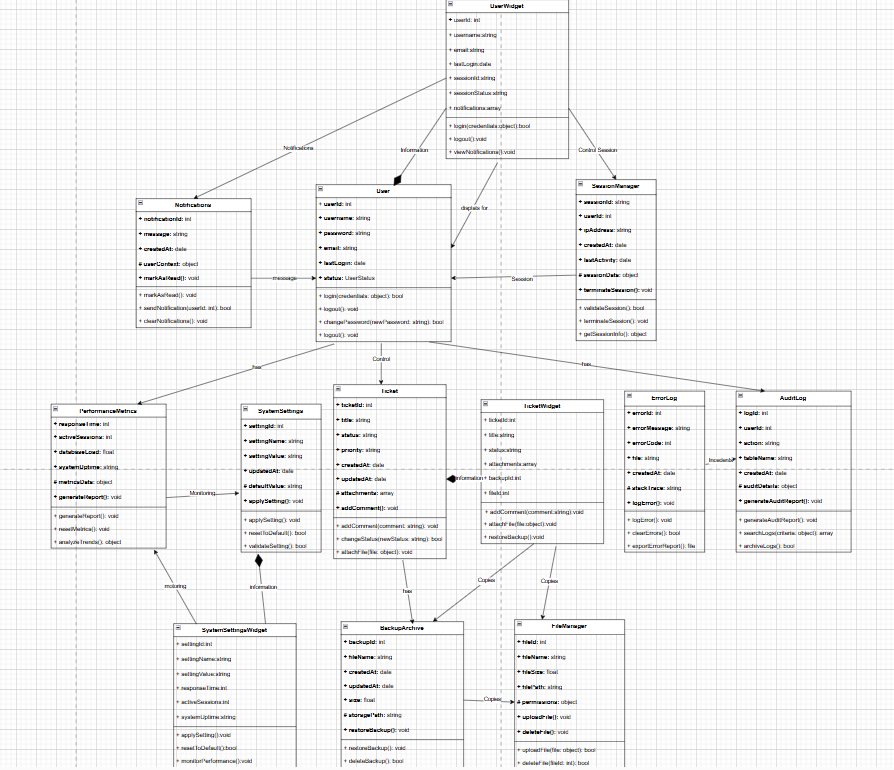


Рисунок № 40 – Виджеты

Добавленные отношения:

UserWidget - SessionManager:

UserWidget управляет пользовательскими сессиями.

UserWidget --> SessionManager: UserWidget управляет сессиями пользователя

SystemSettingsWidget → PerformanceMetrics:

Ассоциация

SystemSettingsWidget --> PerformanceMetrics: SystemSettingsWidget отображает метрики производительности

4. Произвести проверку правильности построения диаграммы, используя выполненный пункт 1.

Проверяющий – Владислав Шашков

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Правило** | **Наличие ошибок** | **Описание** |
| 1 | Класс представляет собой элемент диаграммы, обозначающий множество объектов с одинаковой структурой, поведением и отношениями с другими классами. | - | Классы корректно изображены и структурированы. |
| 2 | Имя класса должно быть существительным в единственном числе, уникальным в пределах диаграммы. Если имя состоит из нескольких слов, оно записывается в стиле CamelCase. | - | - |
| 3 | Статический класс — это класс, который содержит только статические поля и методы. На его основе нельзя создавать объекты. Статический класс помечается стереотипом «utility». | - | - |
| 4 | Абстрактный класс отличается от обычного тем, что его имя записывается курсивом. | + | User может быть абстрактным, но его имя не выделено курсивом. Это создает неоднозначность. |
| 5 | Поле класса описывается с указанием уровня видимости, который может быть: "+" (public), "-" (private), "#" (protected). | - | - |
| 6 | Идентификатор поля — это его уникальное имя, обязательно указывается для каждого поля класса. | - | - |
| 7 | Тип поля показывает тип данных, который хранится в данном поле (например, int, string и т.д.). | - | - |
| 8 | Кратность поля показывает диапазон значений массива, если поле является массивом (например, [m..n] — от m до n элементов, [0..\*] — неограниченное количество элементов). | - | На диаграмме кратность для полей не указана, но это не ошибка, т.к. не все поля являются массивами. |
| 9 | Методы класса описываются с указанием уровня видимости, аргументов и возвращаемого значения. | - | - |
| 10 | Отношение ассоциации показывает связь между классами. Один класс использует функционал другого, ассоциация изображается стрелкой от класса пользователя к классу владельца. Стрелка ассоциации направлена от класса пользователя к классу владельцу используемой функциональности. | - | - |
| 11 | Кратность ассоциации, может быть, не указана. По умолчанию считается, что кратность [0..\*]. | - | - |
| 12 | Если класс — статический, кратность ассоциации не указывается (считается [1]). | - | - |
| 13 | Отношение зависимости используется для указания на то, что изменение одного класса вызывает изменения в другом. Стрелка зависимости указывает от зависимого класса к независимому. Визуализируется пунктирной линией с замкнутым наконечником и иллюстрирует от зависимого класса к независимому. | - | - |
| 14 | Отношение наследования показывает, что один класс является потомком другого (родительский класс — базовый, а потомок — производный). Указывается сплошной линией без закрашенного наконечника и направлена от производного класса к базовому. | - | - |
| 15 | В наследовании класс потомок наследует все поля и методы родительского класса. | - | - |
| 16 | Отношение агрегации показывает, что один класс включает в себя другой как составную часть. Однако класс-часть может существовать обособленно от класса-целого. Закрашивается сплошной линией с ромбовидным наконечником. От составляющих классов до класса, включающегося в себя несколько классов-частей. | - | - |
| 17 | Отношение композиции — частный случай агрегации. В композиции класс-часть не может существовать отдельно от класса-целого. | - | - |
| 18 | В композиции удаление класса-целого автоматически влечет за собой удаление класса-части. | - | - |
| 19 | Стрелки агрегации и композиции отличаются визуально: в агрегации — пустой ромб у класса-целого, в композиции — закрашенный ромб у класса-целого. | - | - |
| 20 | Интерфейсы изображаются на диаграммах как классы, но с добавлением стереотипа «interface». | - | На диаграмме нет интерфейсов |
| 21 | Интерфейс — это набор абстрактных методов, которые должны быть реализованы классами, которые этот интерфейс реализуют. | - | На диаграмме нет интерфейсов |
| 22 | Класс может реализовывать несколько интерфейсов. | - | На диаграмме нет интерфейсов |
| 23 | Связь «реализация» изображается пунктирной стрелкой с треугольником от класса к интерфейсу. | - | На диаграмме нет интерфейсов |
| 24 | Открытое поле («+») доступно из любой части программы. | - | - |
| 25 | Закрытое поле («-») доступно только внутри класса, к которому оно принадлежит. | - | - |
| 26 | Защищённое поле («#») доступно внутри класса и производных классов. | - | - |
| 27 | Параметры методов указываются в формате: имя\_параметра: тип\_параметра. | - | - |
| 28 | Возвращаемое значение метода указывается после его имени через двоеточие, например: имя\_метода(): возвращаемый\_тип. | - | Возвращаемые типы указаны корректно. |
| 29 | Отношение композиции подчеркивает жесткую связь между объектами, когда жизнь одного объекта зависит от жизни другого. | - | Композиция отображена корректно. |
| 30 | Обобщение (наследование) изображается с помощью стрелки с пустым треугольником, указывающим на родительский класс. | - | Наследование не применено, что допустимо. |

Вывод:

1.1. Закрепил теоретические знания по принципам создания диаграммы классов.

1.2. Получил практические навыки по построению диаграммы классов.