**3. ПРОЕКТНО-КОНСТРУКТОРСКАЯ ЧАСТЬ**

1. **Разработка структуры приложения**

Для разработки приложения использовалась среда Microsoft Visual Studio Code, включающая в себя широкий набор функциональных инструментов.

При запуске веб-приложения пользователь сам решает, что ему необходимо сделать. Он может, как и просматривать каталог и всю информацию на сайте без входа и регистрации в роли гостя, так и войти на сайт и оформить бронирование или запросить уведомление может выбрать, какое действие ему совершить в роли клиента.

В любой момент работы с веб-приложением, пользователь может выйти из него.

Данные на сайте добавляются, удаляются и обновляются с помощью администратора. У администратора есть все необходимые для этого возможности. Помимо этого, администратор также может работать над существующими заказами и запросами.

Общее назначение программного средства – предоставление возможности просмотра информации о попугаях, клетках, поставщиках, а также произведение бронирования необходимых товаров.

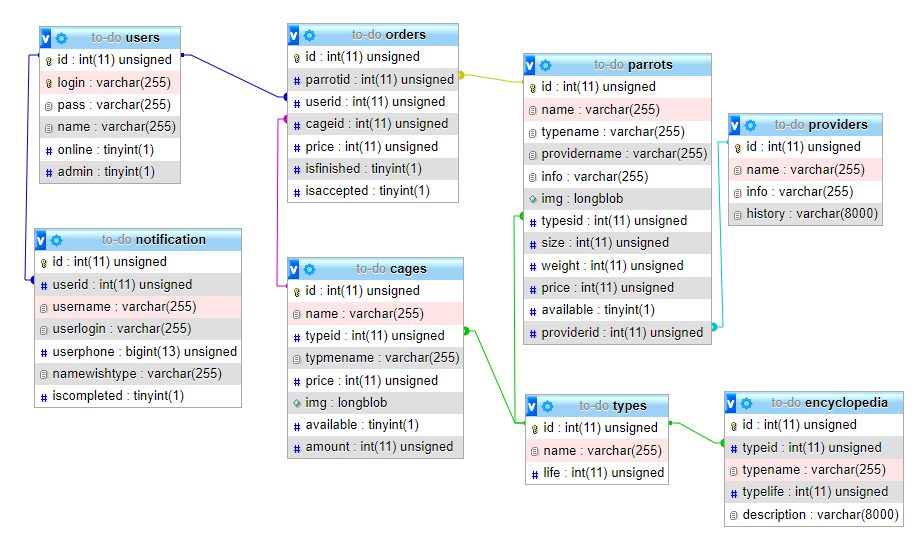
Реализуемая задача состоит в том, чтобы хранение всей информации, необходимой для существования веб-приложения осуществлялось в базе данных, а также производилось удобное взаимодействие с пользовательским интерфейсом.

Рисунок 3.1 – Физическая модель данных

1. **Логическая схема базы данных**

На рисунке 3.3 представлена логическая схема проектируемой базы данных.

Таблица «users» содержит информацию о пользователе (логин, пароль, имя, онлайн он или нет, является ли он администратором или нет).

Таблица «encyclopedia» содержит информацию о энциклопедии. Здесь осуществляется хранение видов, средней продолжительности жизни, описания попугаев.

Таблица «providers» содержит информацию о поставщиках. Здесь происходит хранение имени, описание, история поставщика.

Таблица «types» содержит информацию о видах попугаев. Здесь хранится имени и средней продолжительности жизни попугая.

Таблица «parrots» содержит информацию о попугаях. Здесь осуществляется хранение имени, вида, поставщика, описания, цены, изображения, доступности, размера и веса попугаев.

Таблица «cages» содержит информацию о клетках. Здесь происходит хранение имени, цены, изображения, доступности и количества клеток и вида попугая для этой клетки.

Таблица «orders» хранит информацию о заказе (бронировании). Здесь хранится id товар (либо id попугая, либо id клетки, чтобы не было конфликтов при создании заказов, в структуру базы данных был добавлен следующий запрос рис. 3.2), пользователя, который сделал заказ, цену на товар, состояние заказа (завершен ли он, успешно ли принят или отклонен).

Таблица «notification» содержит информацию о уведомлениях, которые сделали пользователи (например, запрос об уведомлении при появлении в наличии требуемого вида попугая). Здесь хранится информация пользователе, которой делал запрос уведомления, также имя желаемого (запрашиваемого) попугая и завершен или нет запрос (уведомление).

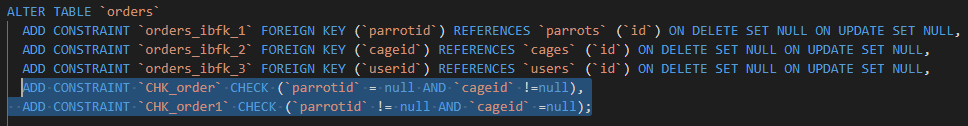


Рисунок 3.2 – Проверка на правильность входных данных

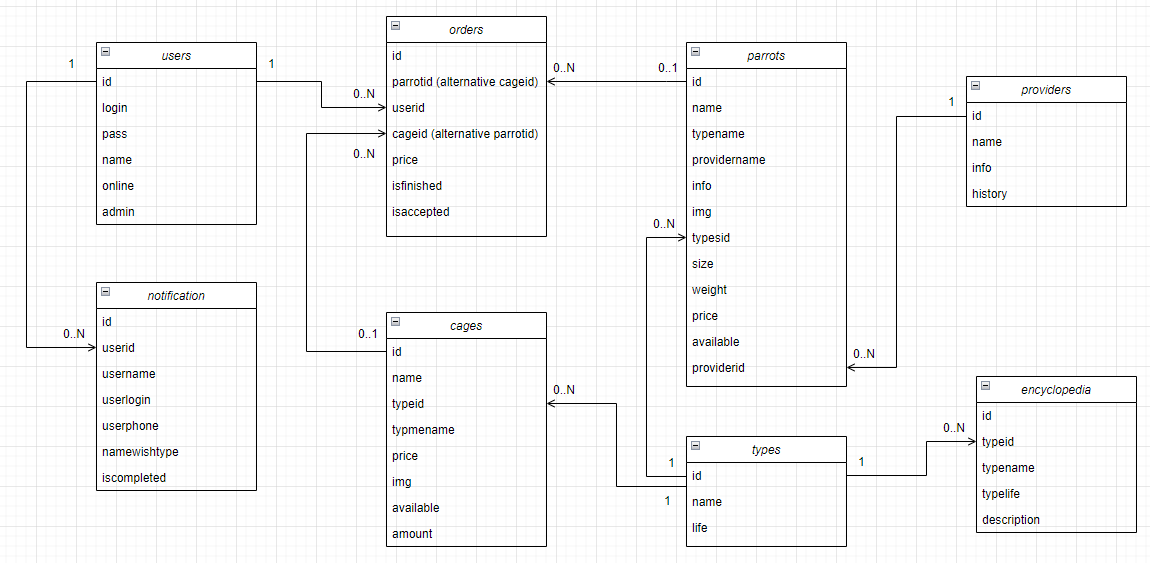


Рисунок 3.3 — Логическая схема проектируемой БД

1. **Разработка архитектуры приложения**

При разработке данного веб-приложения в курсовой работе было принято решение использовать монолитную архитектуру.

В программной инженерии монолитная модель относится к единой неделимой единице. Концепция монолитного программного обеспечения заключается в том, что различные компоненты приложения объединяются в одну программу на одной платформе. Обычно монолитное приложение состоит из базы данных, клиентского пользовательского интерфейса и серверного приложения. Все части программного обеспечения унифицированы, и все его функции управляются в одном месте. Компоненты монолитного программного обеспечения взаимосвязаны и взаимозависимы, что помогает программному обеспечению быть самодостаточным.

Преимущества использования монолитной архитектуры:

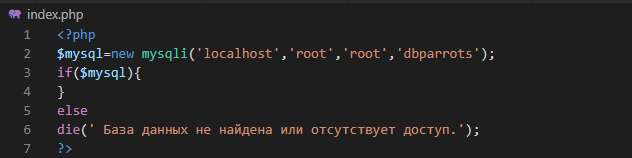
* большим преимуществом является более простая реализация. В монолитной архитектуре можно быстро начать реализовывать свою бизнес-логику, вместо того чтобы тратить время на размышления о межпроцессном взаимодействии, кроме того, все действия выполняются с одним каталогом, что упрощает развертывание;
* улучшенная производительность. Если учитывать, что приложения были собраны правильно, то одно и то же приложение при монолитной архитектуре будет более производительным, чем, например, при микросервисной. Это обеспечивается единым кодом программы и работой из «одного» места.
* при необходимости изменить элементы программы не нужно вносить изменения по отдельности в разных местах — все делается в одном месте.

В случае данного веб-приложения все исходные «.php» файлы имеют примерно одинаковую структуру.

Сначала происходит проверка соединения с БД, далее отображаются начальные HTML элементы приложения, затем происходит прямое обращение к БД с целью извлечения данных, после этого, при необходимости, сразу осуществляется обработка полученных данных и реализация бизнес-логики. Все страницы системы схожи своей структурой, с добавлениями небольших особенностей, которые необходимы для реализации задуманных возможностей веб-приложения.

Отсутствуют какие-либо методы или функции, описывающие реакцию взаимодействия пользователя с системой. Кроме того, отсутствуют методы или функции для описания моделей, контроллеров и представления.

Например, каждая страничка содержит одинаковое начало - необходимость проверки соединения с базой данных, для корректной работы всей системы (рис. 3.4), а также шапку веб-приложения, которая исполняет роль навигации по сайту (рис. 3.5).

Рисунок 3.4 – Проверка соединения с БД

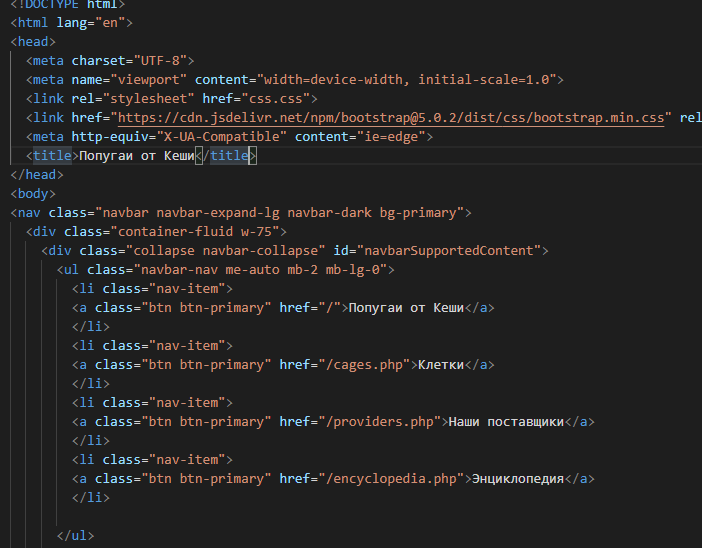


Рисунок 3.5 – Шапка веб-приложения

Далее, если взглянуть на структуру начальной страницы (такая структура присуща всем страницам приложения), после строчки «<?php» происходит работа на серверной части веб-приложения, идет прямое обращение к БД (запрос), извлекаются необходимые данные и при помощи ключевого слова «echo» полученные данные отправляются обратно на клиентскую часть веб-приложения в виде HTML кода (рис. 3.6). Никакие дополнительные методы или функции не создаются и не используются.

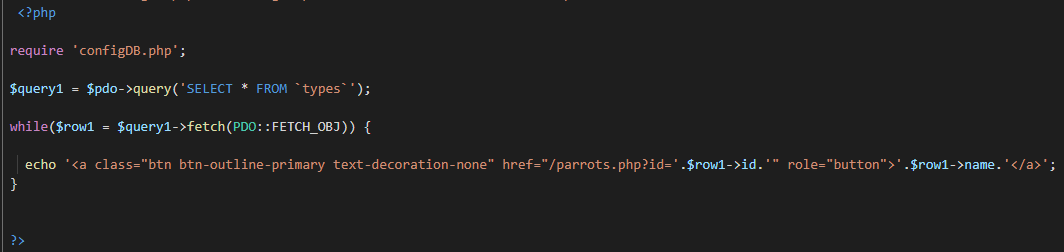


Рисунок 3.6 – Пример работы на серверной части приложения

1. **Разработка интерфейса взаимодействия пользователя с системой**

Для удобного взаимодействия пользователя был разработан следующий интерфейс.

Если пользователь является гостем, ему предоставляется все те же возможности, как и зарегистрированному пользователю, за исключением функции бронирования попугаев или клеток, а именно – просмотр попугаев (рис. 3.7) и клеток (рис. 3.8), просмотр поставщиков (рис. 3.9) и информации о них (рис. 3.10), а также просмотр энциклопедии (рис. 3.11). Если же, не зарегистрированный пользователь захочет сделать бронирование (клетка или попугай, не важно), ничего не получится (рис 3.12). Для успешного бронирования необходимо либо зарегистрироваться под новым пользователем (рис. 3.13), либо зайти уже под существующим (рис. 3.14). После авторизации, пользователь сможет совершить бронирование (рис. 3.15). Также произойдет автоматическое заполнение полей имени и логина пользователя, под которым осуществилась авторизация.

Также для обновления информации на сайте, необходим администратор с возможностями управления веб-приложением. У администратора есть доступ к странице администрирования, при помощи нажатия на специальную кнопку, которая доступна только ему (рис. 3.16). Он обладает рядом возможностей для успешного управления сайтом, а именно просмотр заказов, запрашиваемых попугаев, редактирование энциклопедии, изменение количества клеток, добавление или удаление попугаев, клеток, а также поставщиков (рис. 3.17). Также стоит привести пример написания и обработки запроса для вывода попугаев с определенным id (рис. 3.18).

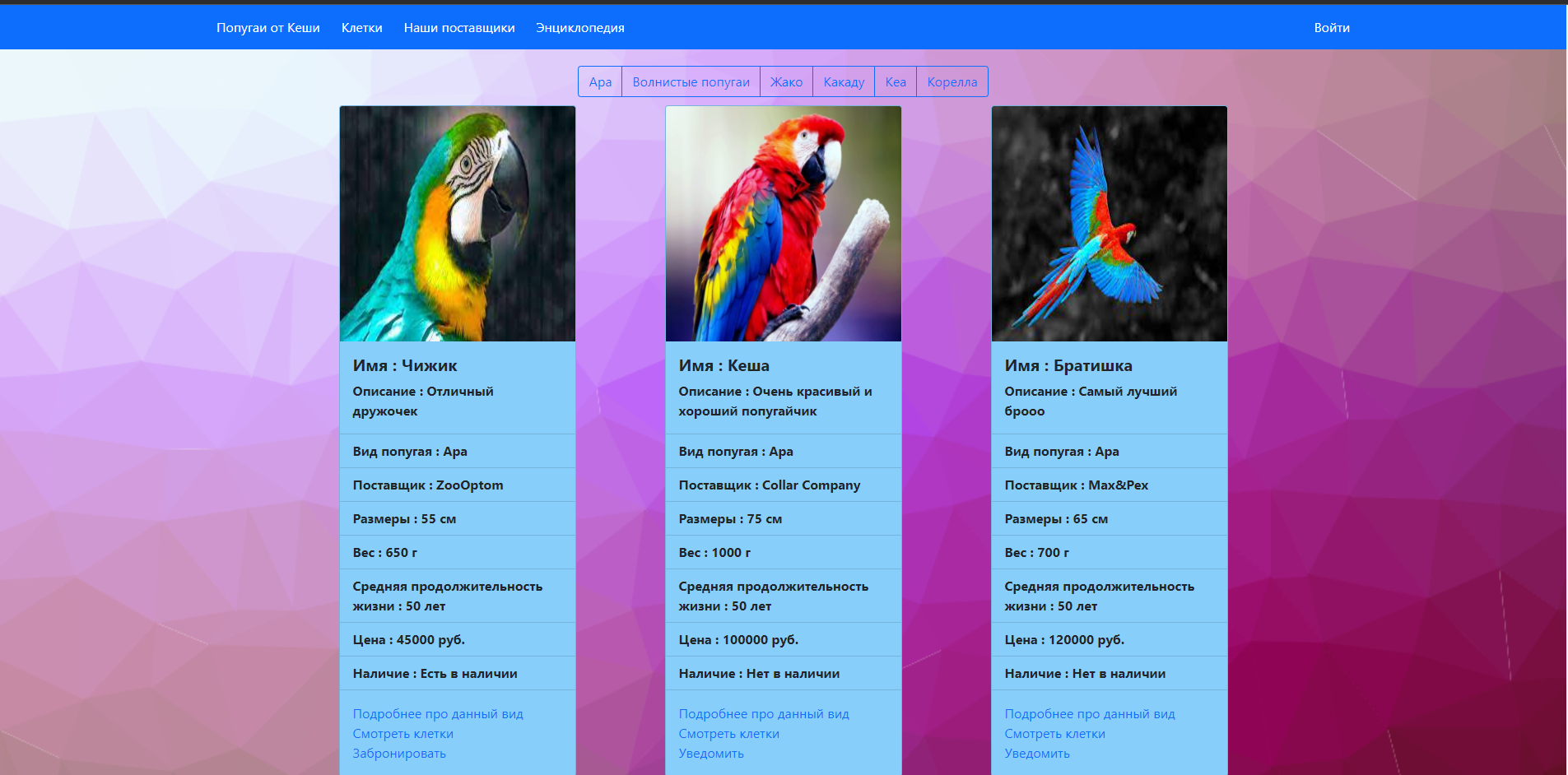


Рисунок 3.7 – Просмотр попугаев

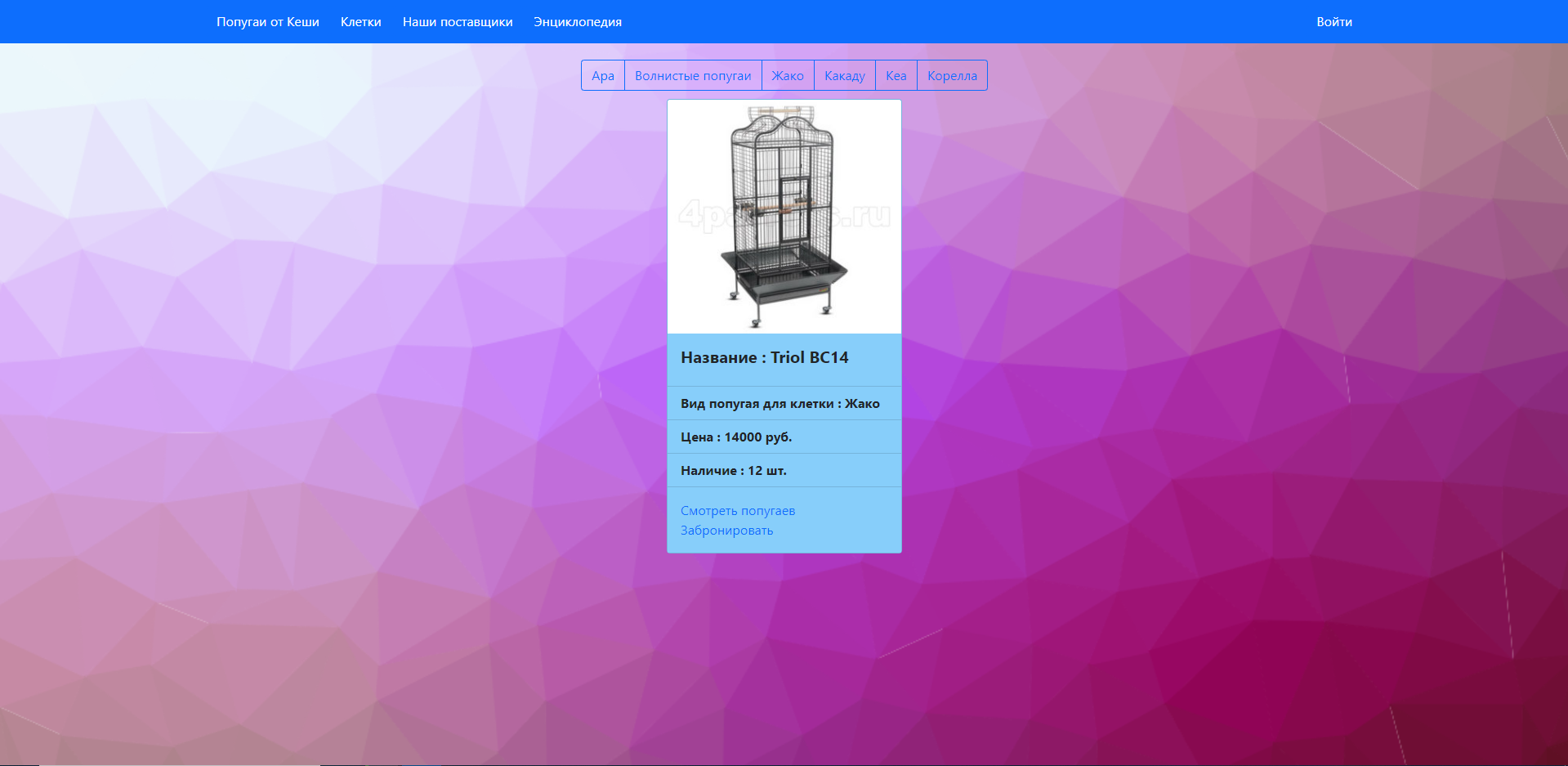


Рисунок 3.8 – Просмотр клеток

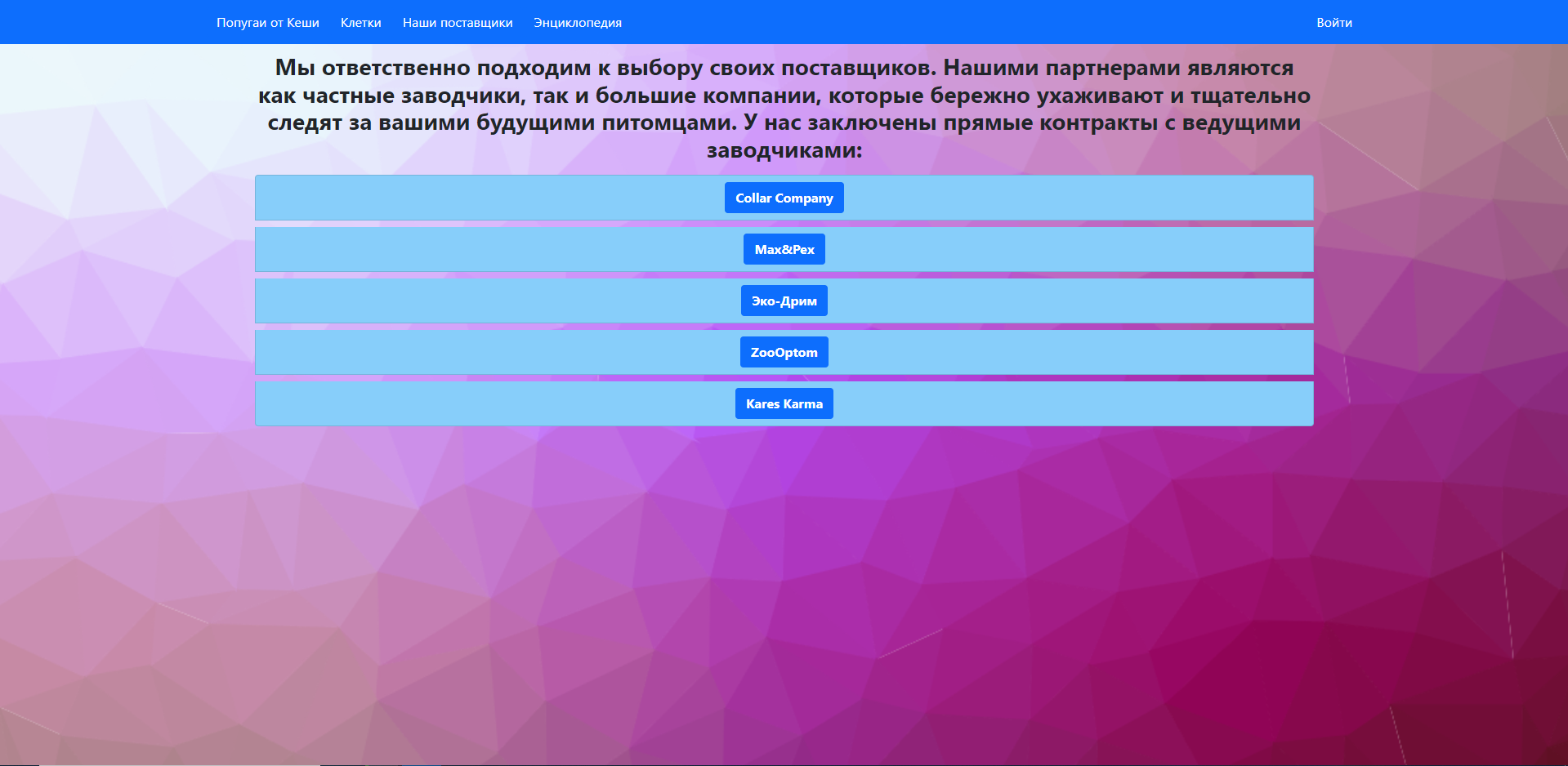


Рисунок 3.9 – Просмотр поставщиков

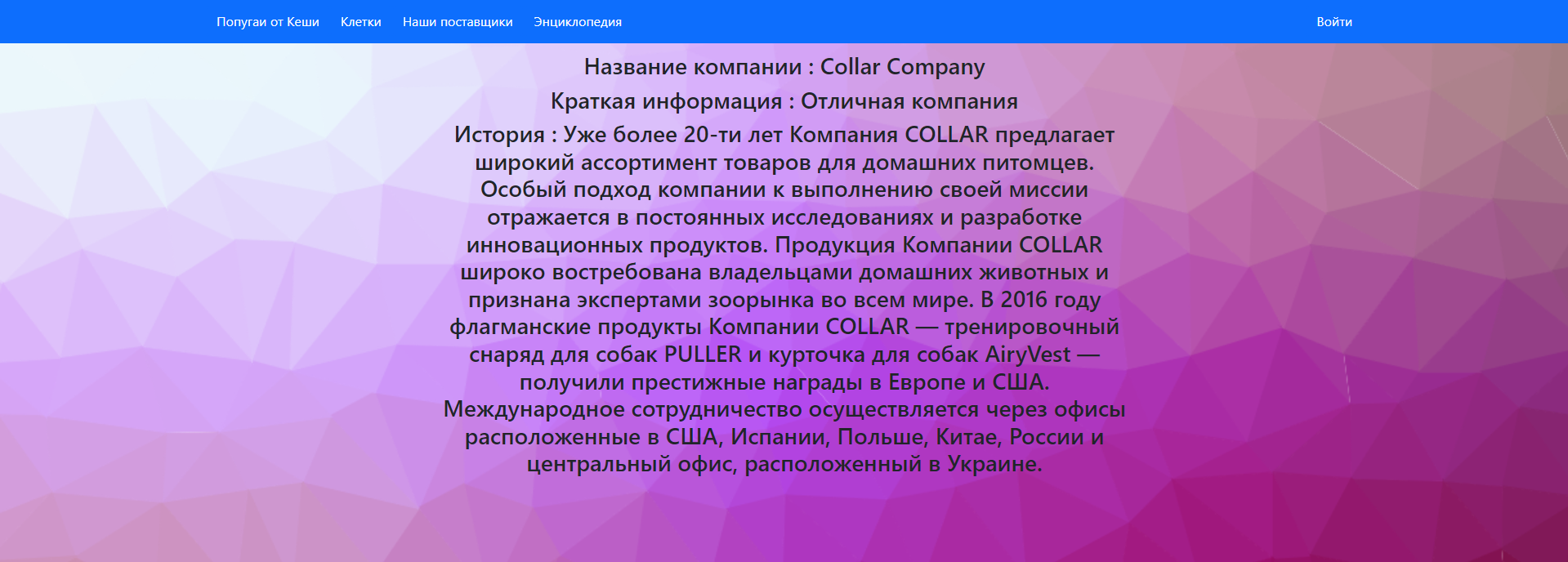


Рисунок 3.10 – Просмотр информации о поставщике

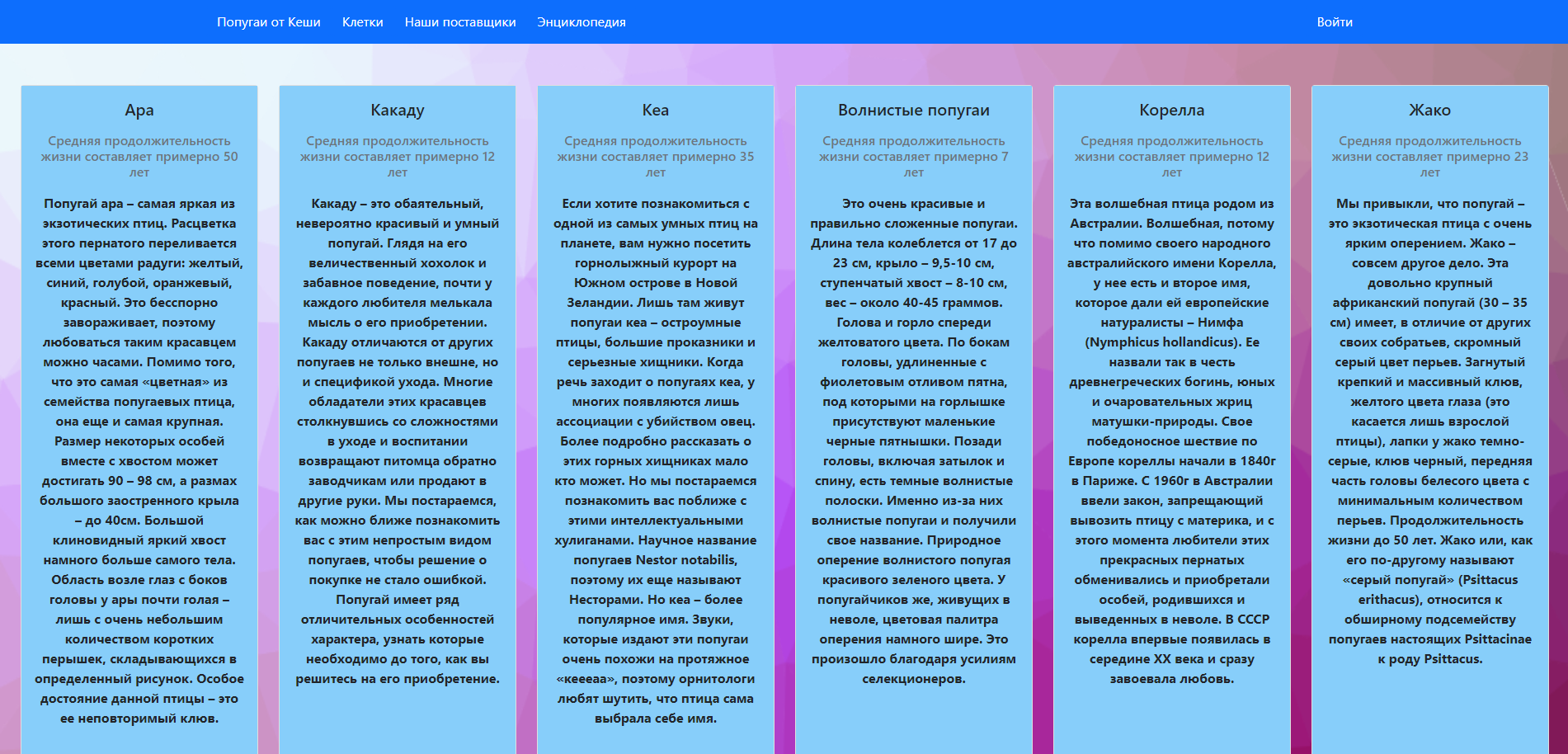


Рисунок 3.11 – Просмотр энциклопедии

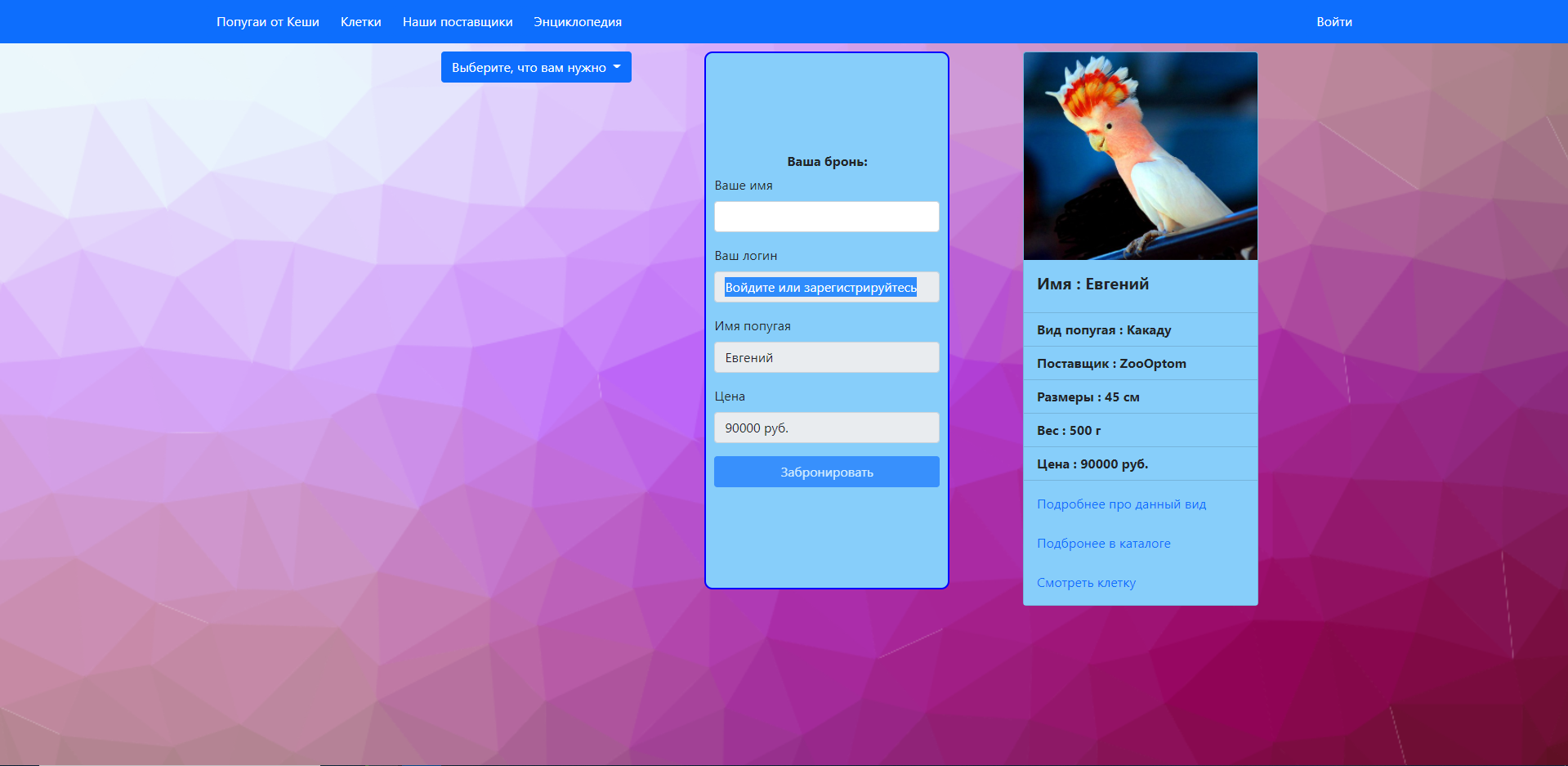


Рисунок 3.12 – Невозможность бронирования незарегистрированному пользователю

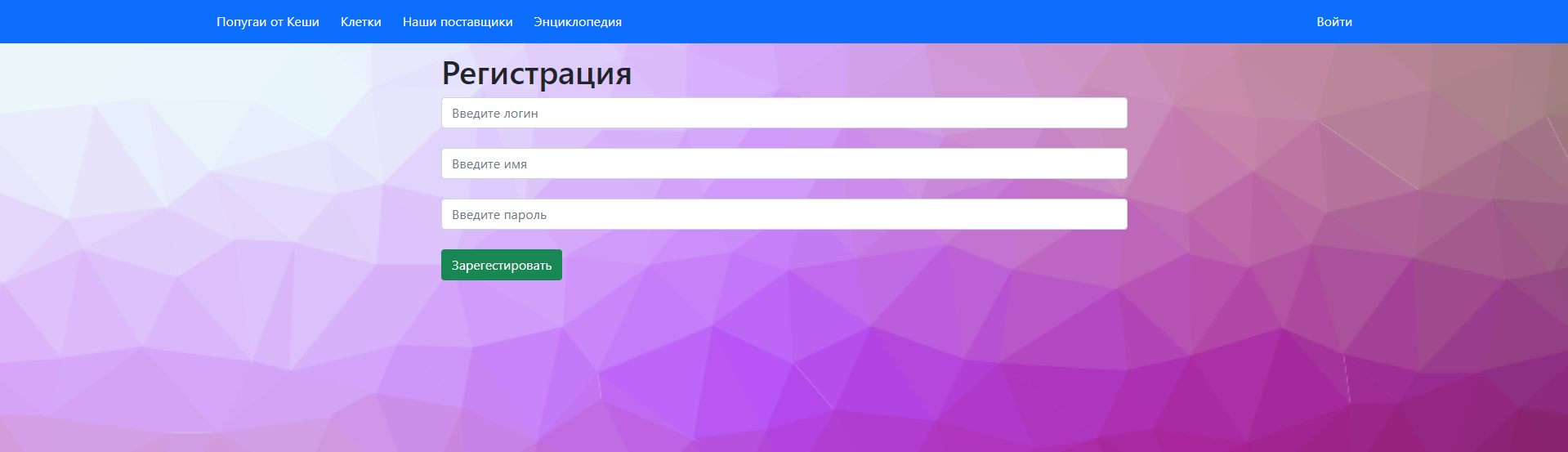


Рисунок 3.13 – Регистрация нового пользователя

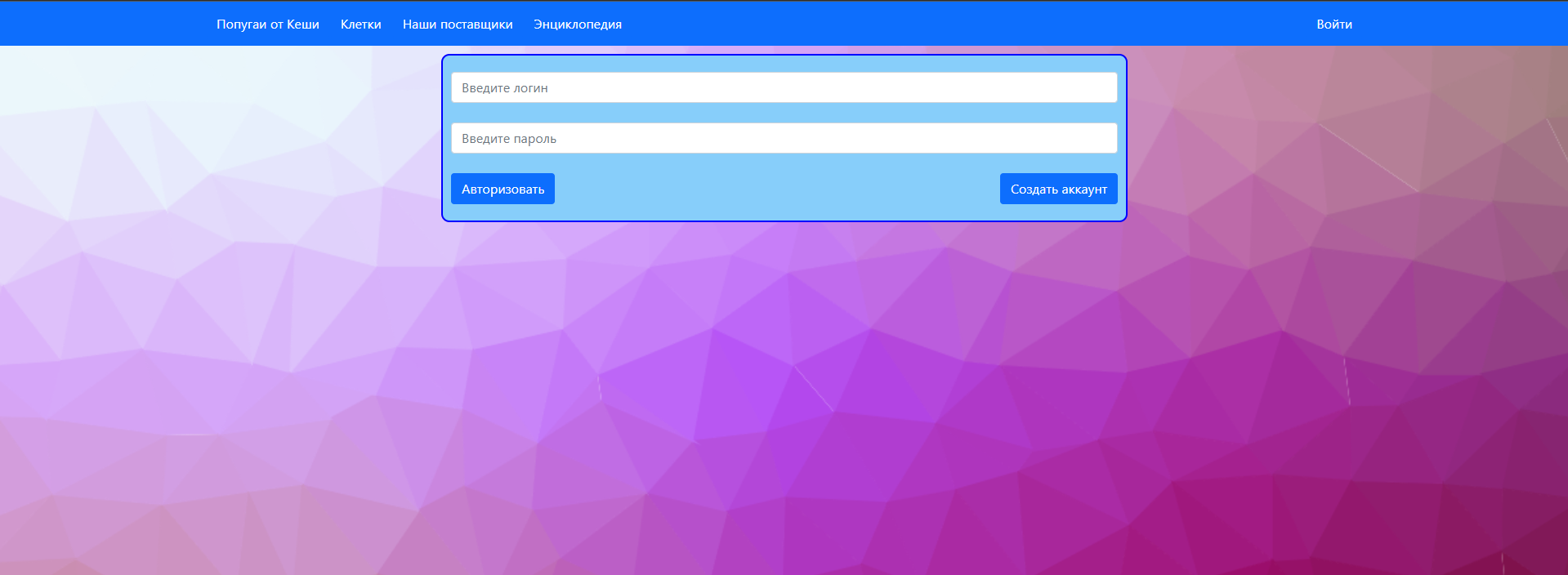


Рисунок 3.14 – Авторизация существующего пользователя

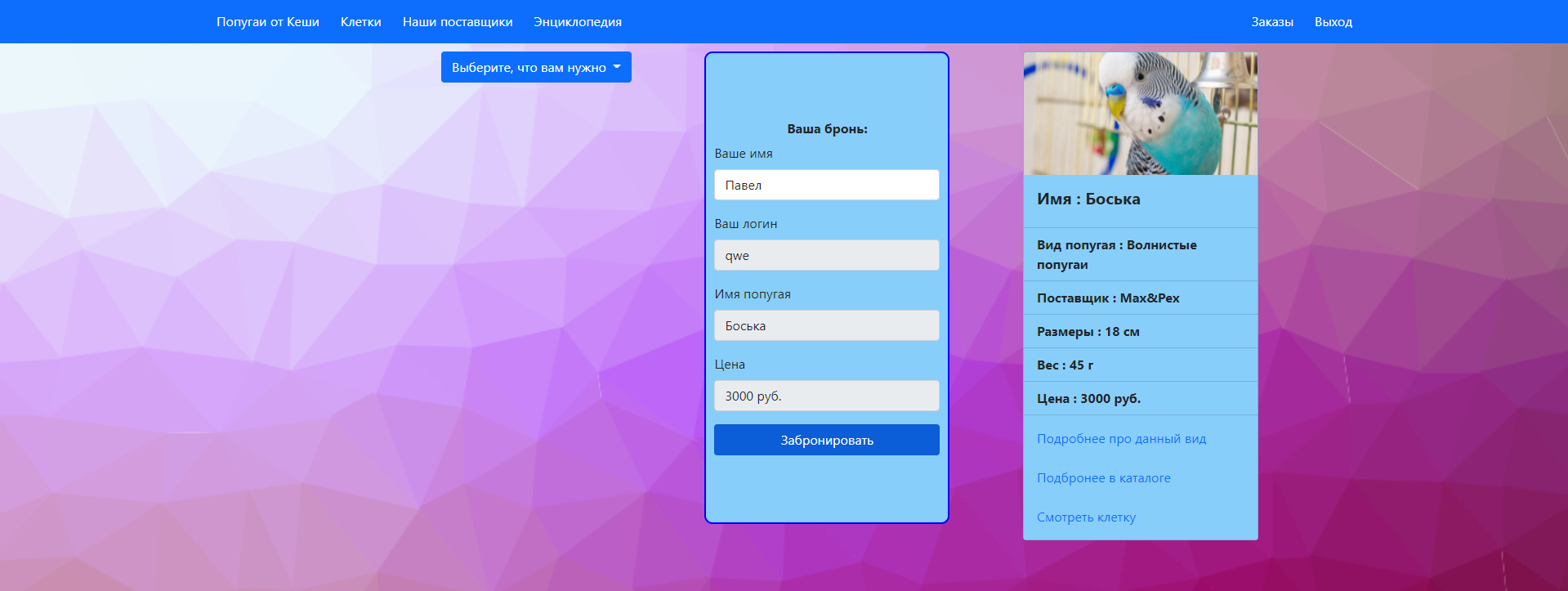


Рисунок 3.15 – Бронирование

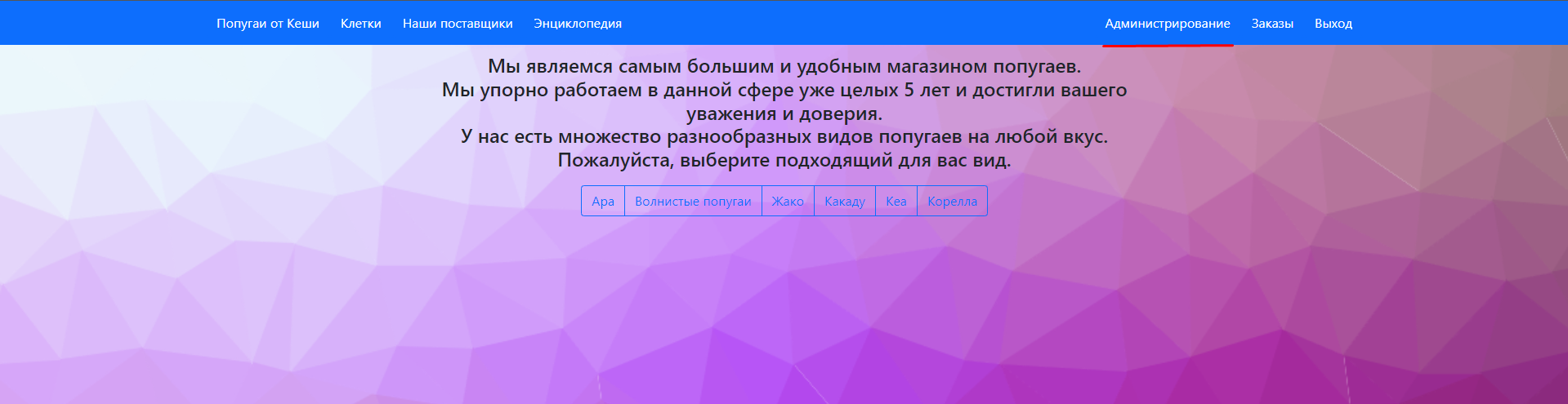


Рисунок 3.16 – Кнопка администрирования

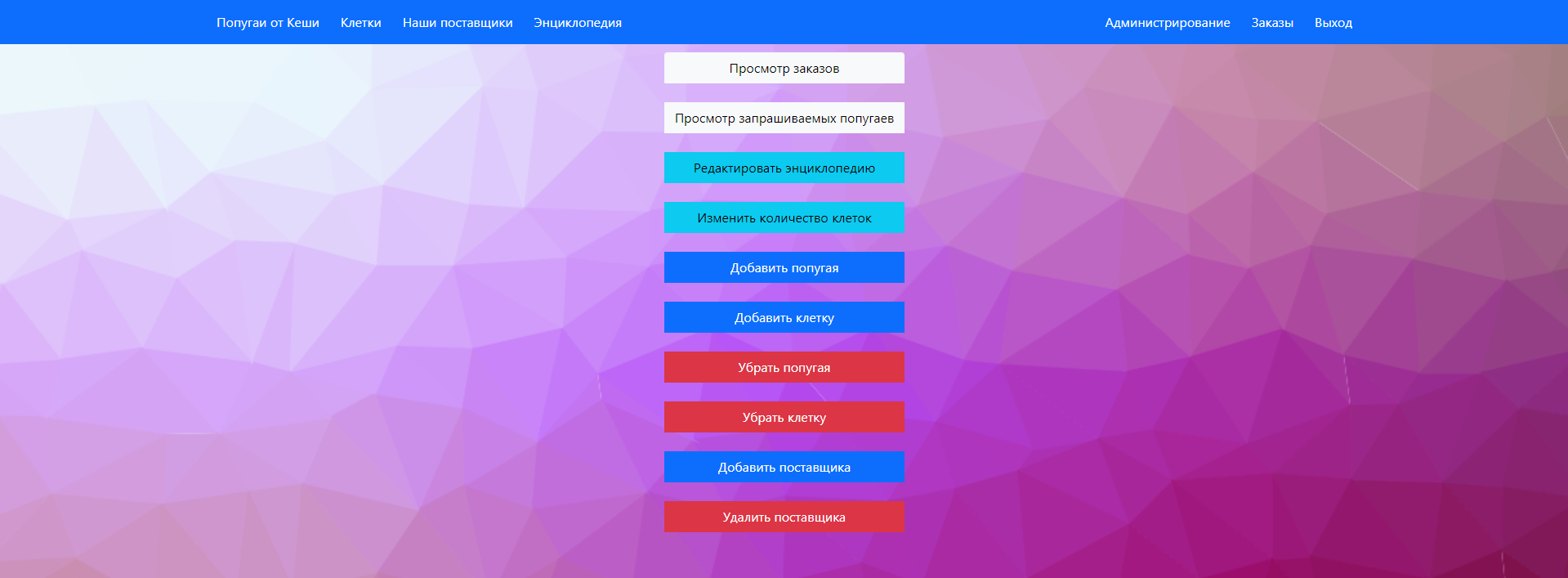
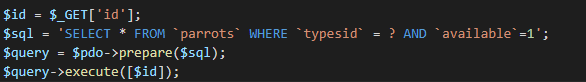


Рисунок 3.17 – Возможности администратора



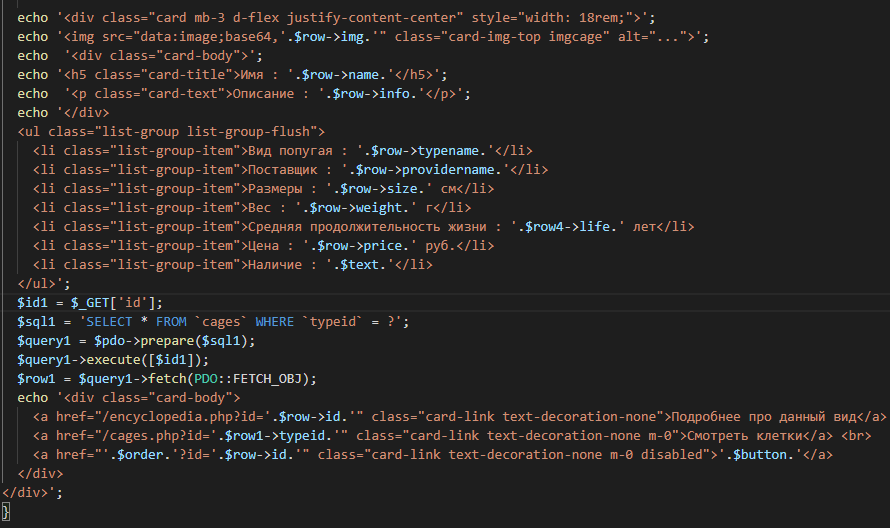


Рисунок 3.18 – Пример запроса получения и вывода попугаев с определенным id