

README.md

Задание

Hasker: Poor Man's Stackoverflow

Задание: Написать Q&A сайт, аналог stackoverflow.com. Это будет Django приложение, покрытое тестами и для которого (опционально) реализовано API

Цель задания: получить навык создания веб-приложений и использования фреймворков.

Критерии успеха: задание **обязательно**, критерием успеха является работающий согласно заданию код, который проходит тесты, для которого проверено соответствие пер8, написана минимальная документация с примерами запуска. Также проект должен разворачиваться одним из способов, описанных в секции Deploy. Далее успешность определяется code review.

Основные сущности

1. Пользователь – электронная почта, никнейм, пароль, аватарка, дата регистрации.
2. Вопрос – заголовок, содержание, автор, дата создания, тэги.
3. Ответ – содержание, автор, дата написания, флаг правильного ответа.
4. Тэг — слово тэга.

Основные страницы и формы

Warning: мокапы - это ориентир, если что-то противоречит здравому смыслу, то придерживайтесь здравого смысла.

1. Index

Листинг вопросов с пагинацией по 20 штук на странице с сортировкой по дате добавления и рейтингу (2 вида сортировки).

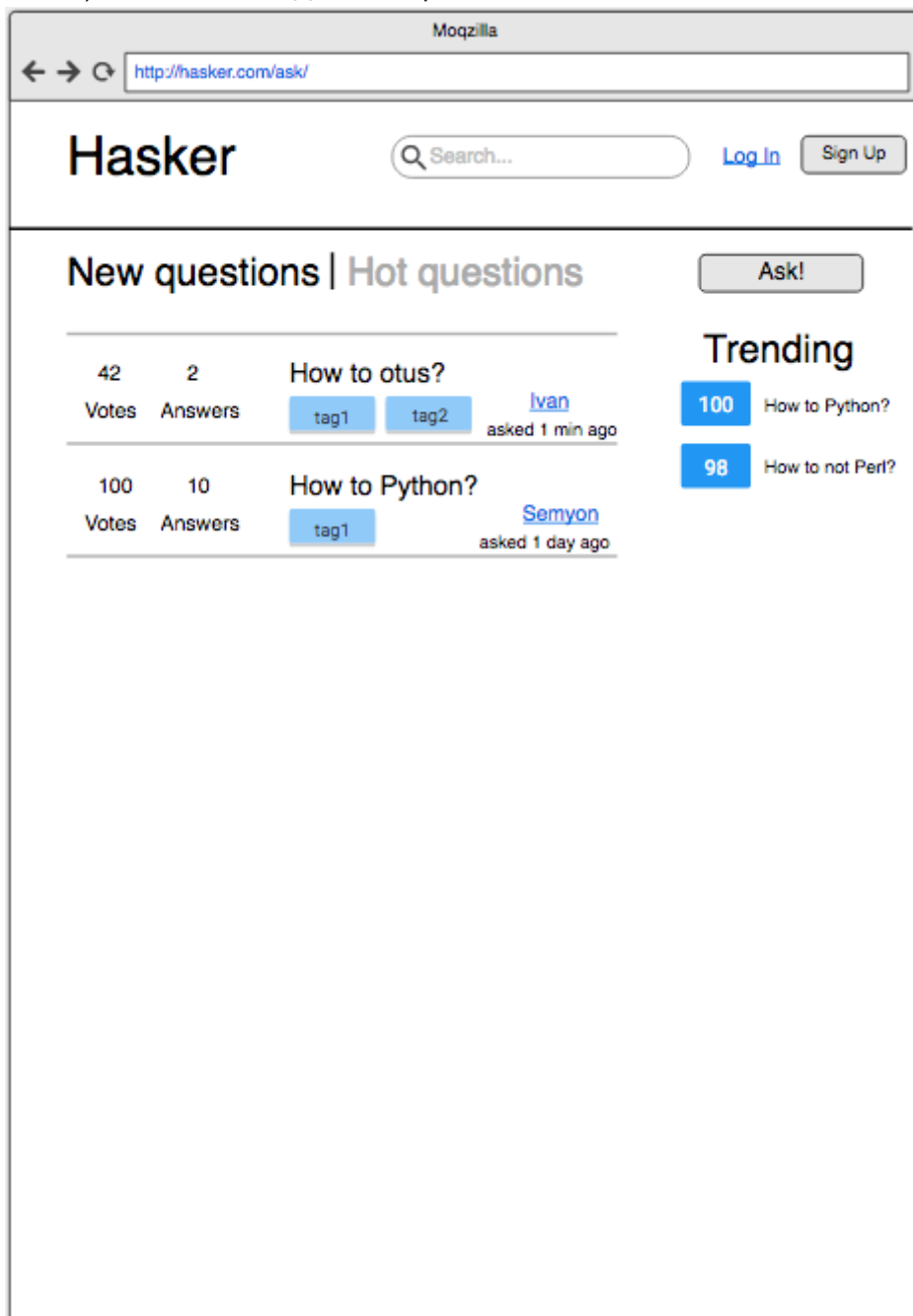
В шапке сайта находятся:

- логотип
- поисковая строка (для быстрого поиска)
- кнопка задать вопрос (доступна только авторизованным).

В правой части шапки — юзерблок.

- Для авторизованного пользователя юзерблок содержит его
 - ник, который ведет на страницу с профилем
 - аватарку
 - ссылки на "выход"
- Для неавторизованных
 - ссылки "войти"
 - и "регистрация".

В правой колонке — информационный блок “Популярные вопросы” (описание ниже) и кнопка “Задать вопрос”.



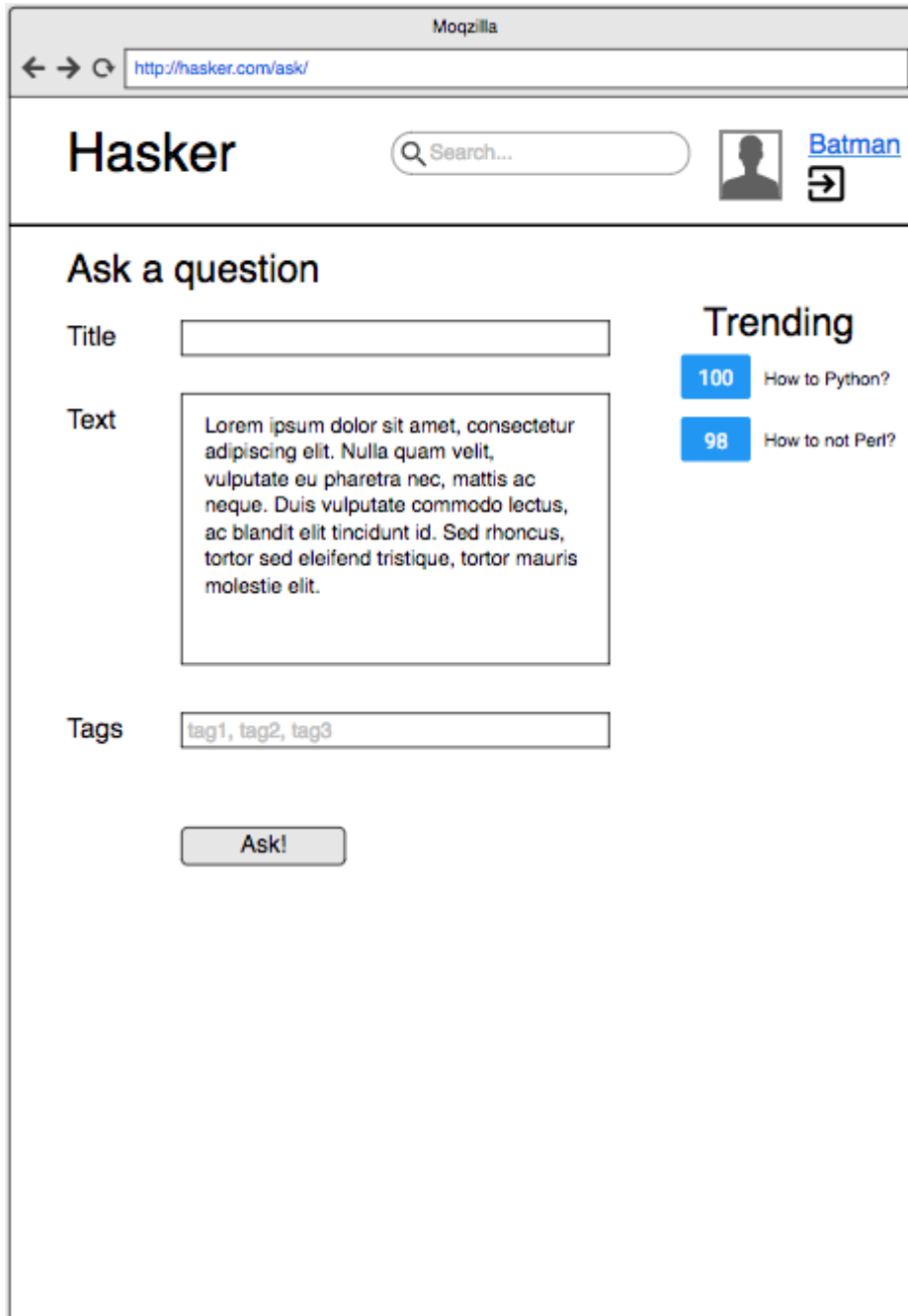
2. Ask Question

Страница добавления вопроса (можно сделать оверлеем). Доступна только для авторизованных пользователей.

В форма вводится

- заголовок
- текст вопроса
- тэги, через запятую.

С вопросом может быть связано не более 3 тегов. Для подсказки при выборе тега можно использовать готовый jquery плагин. При обработке формы обязательно проверка валидности данных. Если вопрос успешно добавлен — пользователя перебрасывает на страницу вопроса, если возникли ошибки — их нужно



The screenshot shows a web browser window with the URL `http://hasker.com/ask/`. The page has a header with the 'Hasker' logo, a search bar, a user profile icon, and the name 'Batman'. The main content area is titled 'Ask a question' and contains a form with the following fields:

- Title**: A text input field.
- Text**: A large text area containing placeholder text: 'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac neque. Duis vulputate commodo lectus, ac blandit elit tincidunt id. Sed rhoncus, tortor sed eleifend tristique, tortor mauris molestie elit.'
- Tags**: A text input field containing 'tag1, tag2, tag3'.
- Ask!**: A button to submit the question.

To the right of the form is a 'Trending' section with two items:

- 100 How to Python?
- 98 How to not Perl?

отобразить в форме.

3. Answer question

Страница вопроса со списком ответов. На странице вопроса можно добавить ответ.

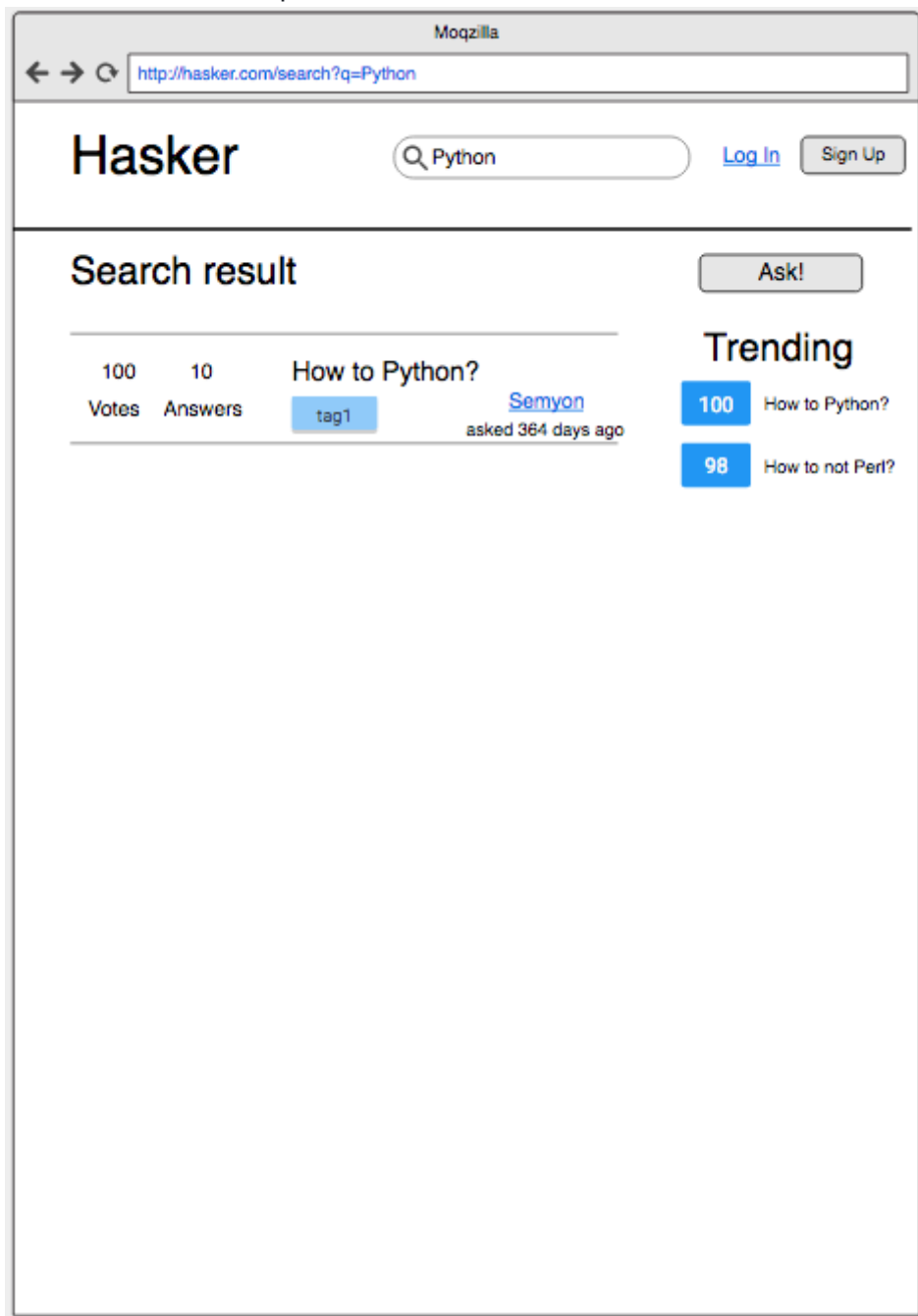
Ответы сортируются по рейтингу и дате добавления при равном рейтинге. Ответы разбиваются по 30 штук на странице.

Форма добавления ответа находится на странице вопроса. Отображается только для авторизованных пользователей. После добавления ответа, автор вопроса должен получить email с уведомлением от нового ответа. В этом письме должна быть ссылка для перехода на страницу вопроса. Автор вопроса может пометить один из ответов как правильный. Пользователи могут голосовать за вопросы и ответы с помощью лайков «+» или «-». Один пользователь может голосовать за 1 вопрос и ответ только 1 раз, однако может отменить свой выбор или переголосовать неограниченное число раз.

The screenshot shows a web browser window with the URL <http://hasker.com/question/how-to-otus/>. The page header includes the Hasker logo, a search bar, and a user profile for 'Batman'. The main content area displays a question titled 'How to otus?' with a score of 42. The question text is a placeholder Lorem Ipsum. Below the question are two tags: 'tag1' and 'tag2', and the user 'Ivan' who asked the question. To the right of the question is a 'Trending' section with two items: 'How to Python?' with a score of 100 and 'How to not Perl?' with a score of 98. Below the question are two answers. The first answer has a score of 21 and is by user 'Boris'. The second answer has a score of 11 and is by user 'Petr'. At the bottom of the page is a form for 'Your answer' with a text input field and a 'Submit' button.

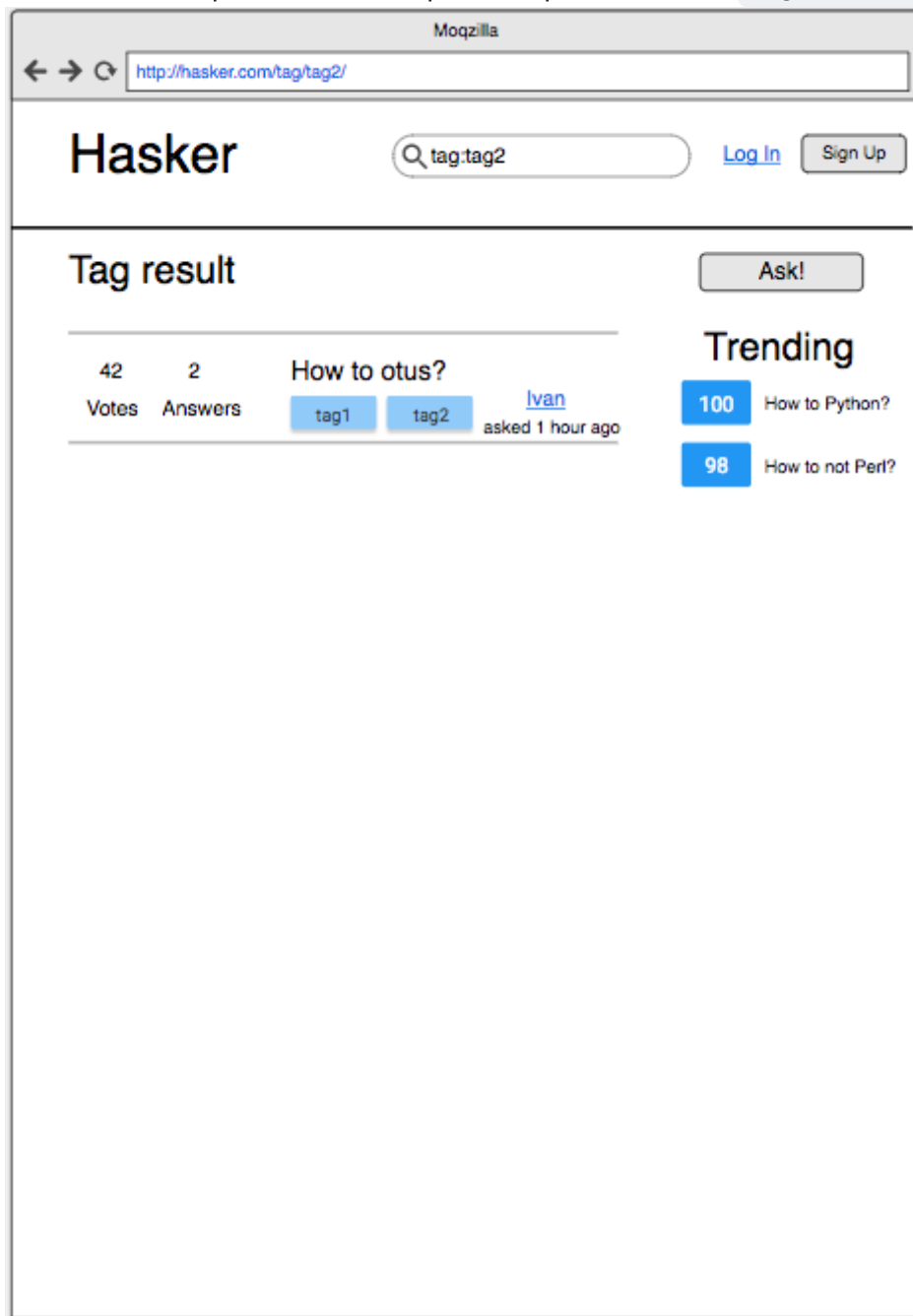
4. Search question

Страница результатов поиска по вопросам. Состоит из поисковой строки и листинга вопросов. Сортировка — по рейтингу (дате если рейтинг одинаковый), чем выше рейтинг и свежее вопрос, тем он выше в результатах. Пагинация по 20 вопросов. Поиск идет одновременно по текстам вопросов и их заголовкам, но в списке только вопросы.



5. Search tag

Листинг вопросов по тэгу. На этой странице выводятся все вопросы содержащие некоторый тэг. Сортировка — по рейтингу (дате если рейтинг одинаковый), чем выше рейтинг и свежее вопрос, тем он выше в результатах. Пагинация по 20 вопросов. Пользователи попадают на эту страницу кликая по одному из тэгов в описании вопроса или набирая в строке поиска `tag:<имя тэга>`



6. Sign Up

Страница регистрации. Любой пользователь может зарегистрироваться на сайте, заполнив форму с электронной почтой, никнеймом, аватаркой и паролем. Аватарка загружается на сервер и отображается рядом с вопросами и ответами

The screenshot shows a web browser window with the address bar displaying `http://hasker.com/signup/`. The page title is "Hasker". Below the title is a search bar and two buttons: "Log In" and "Sign Up". The main content area is titled "Sign Up" and contains a registration form. The form has the following fields and controls:

- Login:** A text input field containing the text "Ivan".
- Email:** A text input field containing the text "ivan@otus.ru".
- Password:** A text input field with masked characters (dots).
- Repeat Password:** A text input field with masked characters (dots).
- Avatar:** A text input field containing "me.jpg" and an "Upload" button next to it.
- Buttons:** An "Ask!" button is located to the right of the "Sign Up" title. An "Enter" button is located below the "Avatar" field.

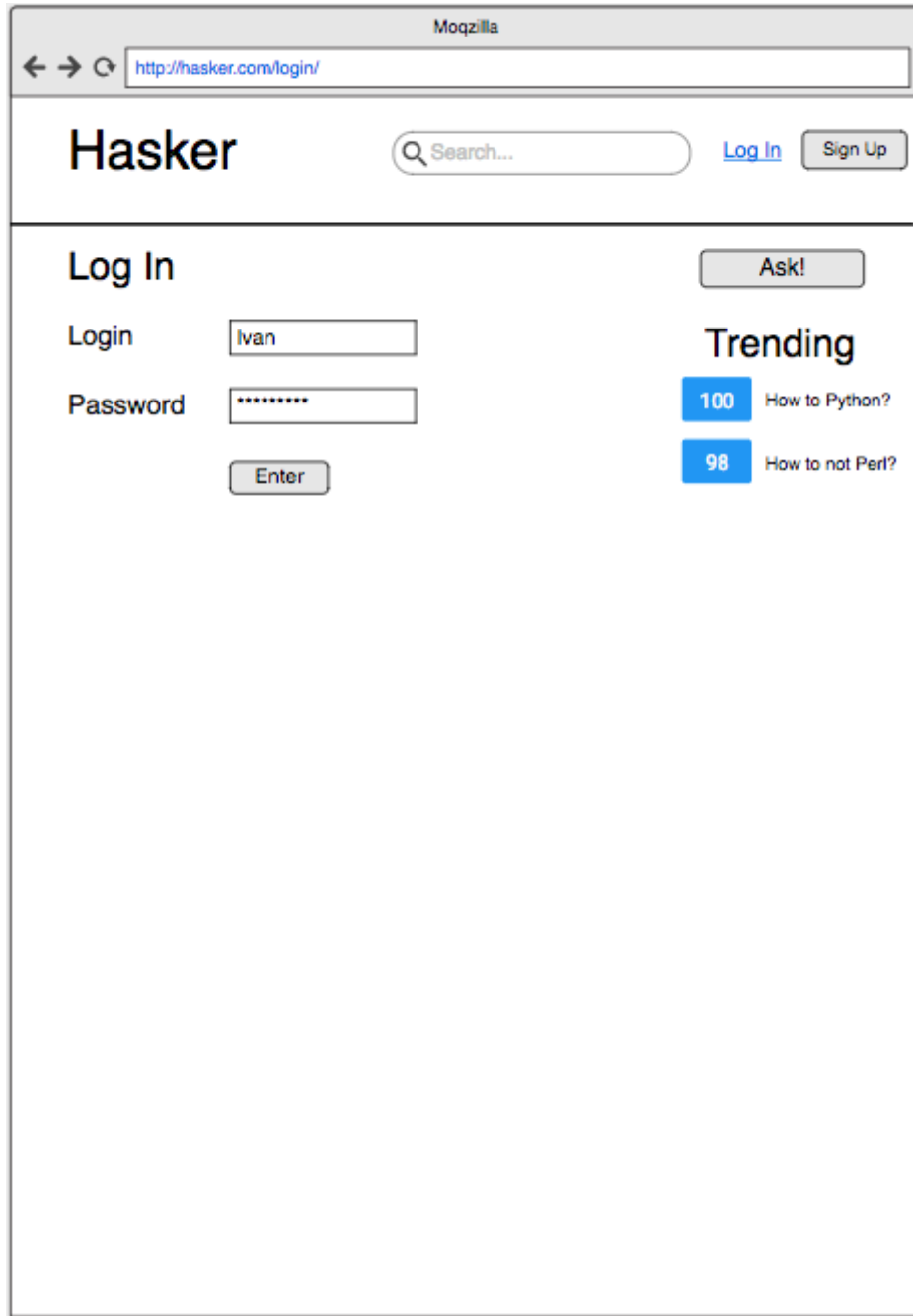
To the right of the form, there is a "Trending" section with two items:

- 100** How to Python?
- 98** How to not Perl?

пользователя.

7. Log In

Форма авторизации. Состоит из поля логин и пароль. Дополнительно есть ссылка на форму регистрации. При успешной авторизации пользователь перебрасывается на исходную страницу, при неуспешной авторизации — ему показывается ошибка. Для авторизованных пользователей вместо этой формы должна показываться

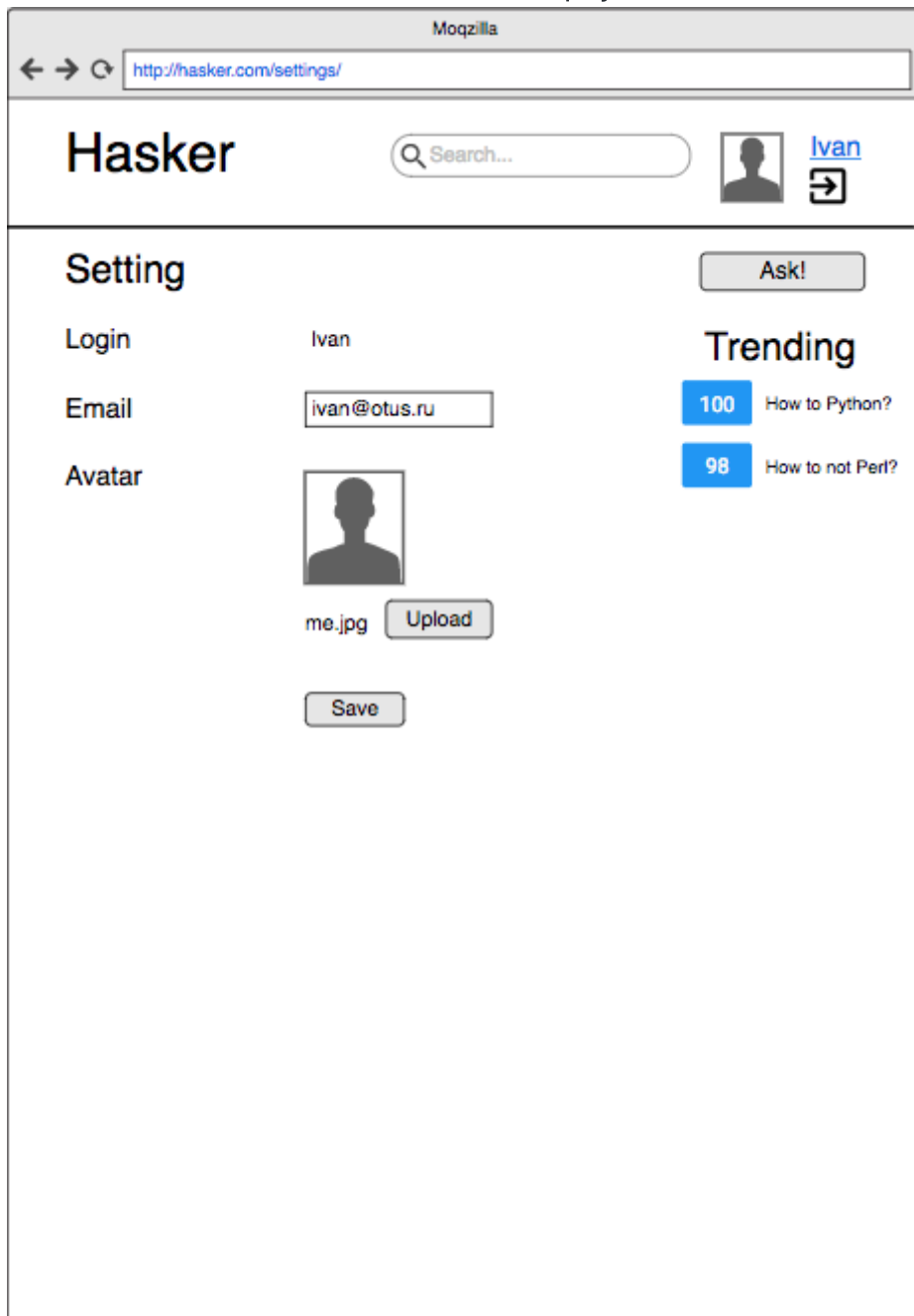


The screenshot shows a web browser window titled "Mozilla" with the address bar displaying "http://hasker.com/login/". The page header features the "Hasker" logo, a search bar with the placeholder "Search...", and links for "Log In" and "Sign Up". The main content area is divided into two sections. On the left, the "Log In" section contains a "Login" field with the text "Ivan", a "Password" field with masked characters "*****", and an "Enter" button. On the right, there is an "Ask!" button and a "Trending" section. The "Trending" section lists two items: "100 How to Python?" and "98 How to not Perl?".

кнопка "Выйти".

8. User Settings

Страница пользователя содержит его настройки — email, nick и аватарку. Каждый пользователь может смотреть только свою страницу. У пользователя должна быть возможность изменить email и аватарку.



9. Trending

В правой колонке сайта находится список из 20 наиболее популярных вопросов, т.е. вопросы с наивысшим рейтингом

API

Опциональное API создаем с помощью Django REST Framework. Что оно должно уметь:

- аутентификация: basic или jwt (лучше)
- получить index (аналогично корню сайта с пагинацией), trending
- сделать поисковый запрос
- получить вопрос
- получить ответы к вопросу

Что еще нужно:

- тесты
- swagger схема

Конкретная структура API выбирается самостоятельно на основании знаний, полученных на занятии.

Stack

- Последняя CentOS или Ubuntu
- Nginx
- uWSGI
- Django
 - без сторонних app'ов
 - django-debug-toolbar можно
- PostgreSQL/MySQL
- Twitter Bootstrap
- Javascript, jQuery

Deploy

На лекции мы обсуждали layout проекта на Django, в том числе там был Makefile. Проект должно уметь запускаться по команде make prod. Т.е. предполагается следующая последовательность действий:

1. создается контейнер с маппингом 80 порта контейнера на 8000 хоста: `docker run --rm -it -p 8000:80 /bin/bash`
2. клонируется ваш гит `git clone <ваш гит>`
3. заходим в директорию проекта `cd hasker` (именно такую директорию)
4. запускаем команду `make prod`, которая настроит все, что нужно, исходя из того, что у нас голый контейнер
 - установит нужные пакеты
 - запустит демоны

В итоге, на 8000 порту хоста должен открываться корень сайта.

В качестве альтернативы (и это был бы идеальный вариант), можно поднять своеобразный стейджинг пользуясь этой инструкцией <https://simonwillison.net/2017/Oct/17/free-continuous-deployment/>. Также может пригодиться <http://www.eidel.io/2017/07/10/dockerizing-django-uwsgi-postgres/>, http://p.agnihotry.com/post/the_free_stack_aws/, <https://www.mattlayman.com/understand-django/deploy-site-live/>.

Еще, может кому-то понадобится: <https://ngrok.com/>.

Deadline

Проект должен быть сдан целиком через 3 недели. Нарушение дедлайна (пока) не карается, пытаться сдать ДЗ можно до конца курса. Но код, отправленный с опозданием, когда по плану предполагается работа над более актуальным ДЗ, будет рассматриваться в более низком приоритете без гарантий по высокой скорости проверки

Обратная связь

Студент коммитит все необходимое в свой github/gitlab репозиторий. Далее необходимо зайти в ЛК, найти занятие, ДЗ по которому выполнялось, нажать "Чат с преподавателем" и отправить ссылку. После этого ревью и общение на тему ДЗ будет происходить в рамках этого чата.