МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики
Кафедра анализа данных и искусственного интеллекта

КУРСОВАЯ РАБОТА

РАЗРАБОТКА САЙТА СРЕДСТВАМИ PYTHON

Работу выполнила _____ М.А. Наталенко
(подпись)

Направление подготовки 09.03.03 Прикладная информатика

Направленность (профиль) Прикладная информатика в экономике

Научный руководитель
ст. преподаватель _____ Е.В. Казаковцева
(подпись)

Нормоконтролер
канд. физ.-мат. наук, доц. _____ Г.В. Калайдина
(подпись)

Краснодар
2021

РЕФЕРАТ

Курсовая работа 23 страницы, 6 рисунков, 1 таблица, 4 источника.

САЙТЫ, PYTHON, DJANGO, РАЗРАБОТКА

В данной работе была изучена теоретическая сторона создания сайта средствами Python и рассмотрены его фреймворки.

Цель данной курсовой работы – познакомиться с языками программирования и фреймворками для создания сайтов.

Задачи:

– проанализировать существующие языки программирования для создания сайтов и выбрать оптимальный;

– изучить фреймворки выбранного языка программирования и выбрать оптимальный;

– рассмотреть основные конструкции выбранного фреймворка.

Объект исследования – сайт. Предмет исследования – инструменты для создания сайтов. Итог проделанной работы – нахождение и выбор подходящих языка программирования и фреймворка для дальнейшего создания сайта.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современное информационное общество и развитие технологий определяет вступление России в эпоху информационного общества, основными составляющими которого являются разнообразные веб-системы. Они определяют его возможности, влияют на решения, способствуют улучшению уровня жизни. В этой ситуации знание механизмов создания и функционирования подобных систем очень актуально и позволяет ставить правильные задачи при их создании.

Существует большое количество составляющих глобальной сети Интернет, где свое место находят сайты организаций и предприятий, личные странички пользователей, социальные сети и множество других веб-сервисов. Все это позволяет пользователю общаться с людьми, находящимися в любой точке земного шара, быстро и комфортно отыскивать любую необходимую информацию, публиковать для всеобщего сведения данные, которые он хотел бы сообщить всему миру.

Любая организация, коммерческая или некоммерческая, в эпоху информационного общества зависит от информационных ресурсов, важнейшим из которых сейчас является веб-сайт. Для коммерческих компаний присутствие в Интернете – это возможность рассказать о своих товарах и услугах, найти потенциальных партнеров и клиентов, а также снизить издержки за счет Интернет-торговли, использования «облачных» сервисов. Для некоммерческих компаний веб-сайт дает возможность быстрее находить волонтеров, расширять сферу деятельности, при этом не затрачивая больших средств.

В связи с этим, можно сказать, что наличие сайта является важным элементом развития в современном информационном обществе и знание технологий создания сайтов является необходимым условием при планировании, проектировании и разработке веб-систем, в число которых

могут входить веб-сайты, мобильные приложения, информационные системы, интернет-магазины.

Цель данной курсовой работы – познакомиться с языками программирования и фреймворками для создания сайтов.

Задачи:

– проанализировать существующие языки программирования для создания сайтов и выбрать оптимальный;

– изучить фреймворки выбранного языка программирования и выбрать оптимальный;

– рассмотреть основные конструкции выбранного фреймворка.

Итог проделанной работы – нахождение и выбор подходящих языка программирования и фреймворка для дальнейшего создания сайта.

1 Анализ языков программирования

Основным инструментом разработки сайта является язык программирования. Его необходимо выбрать в первую очередь, так как от этого будет зависеть выбор дальнейших средств разработки. В процессе выбора языка программирования необходимо провести анализ существующих языков программирования. Ниже рассмотрены языки программирования, которые чаще всего используются при создании сайтов, такие как Ruby, Python и PHP, и проводится их сравнение.

1.1 PHP

PHP (hypertext preprocessor) – скриптовый язык программирования общего назначения. PHP используется для написания сценариев, которые будут выполняться на стороне сервера. Данный язык программирования может выполнять различные операции с HTML-документами, перенаправлять на другие страницы веб-сайта, управлять аутентификацией пользователей, обрабатывать файлы, загружаемые на сервер, и предоставлять доступ к базе данных [1, 2, 3].

Основные преимущества PHP:

– достаточно легок в освоении;

– свободно распространяемый;

– очень популярен, поддерживается большим сообществом программистов;

– включает в себя поддержку основных возможностей объектно-ориентированного программирования;

– имеет большое количество расширений и библиотек.

Основные недостатки PHP:

– невозможность асинхронной и многопоточной работы приложения;

– низкое качество кода;

– несогласованность синтаксиса функций;

– веб-сайты, которые написаны на PHP, часто имеют проблемы с безопасностью.

1.2 Ruby

Ruby – объектно-ориентированный язык программирования высокого уровня. Язык обладает независимой от операционной системы реализацией многопоточности, а также строгой динамической типизацией. С помощью Ruby можно быстро и просто программировать, так как нет необходимости объявлять переменные и их типизировать. Есть хорошая поддержка операций со строками и регулярными выражениями. Существует очень мощный фреймворк Ruby-On-Rails для создания веб-приложений [2, 3].

Основные преимущества Ruby:

– возможность расширения возможностей языка с помощью множества библиотек;

– интегрируемость с различными СУБД, такими как: MySQL, DB2, Oracle;

– простой и удобный интерфейс;

– имеет средства для работы с массивами и строками;

– наличие фреймворков для создания веб-сайтов.

Основные недостатки Ruby:

– сложность в освоении;

– отсутствие необходимого количества документации;

– является менее производительным языком программирования, в сравнении с Python и PHP.

1.3 Python

Python, как и Ruby, является объектно-ориентированный языком программирования высокого уровня. Синтаксис этого языка минималистичен и прост, стандартные библиотеки имеют очень большое количество полезных функций [2]. Python используется для решения различных задач, в том числе для создания веб-сайтов. С помощью Python можно динамически генерировать HTML-код со стороны сервера [3].

Основные преимущества Python:

- удобный и легко понятный программный код;
- удобное и быстрое создание моделей системы;
- большое количество документации, а также быстрорастущее сообщество программистов, использующий Python;
- объектно-ориентированный язык;
- огромное количество различных библиотек;
- разнообразие фреймворков для создания веб-сайтов.

Основные недостатки Python:

- не самый популярный язык программирования;
- несовершенство многопоточности;
- начальная ограниченность средств работы с базой данных, но впоследствии устранимая.

1.4 Сравнение Ruby, PHP и Python

В таблице 1(Таблица 1) представлено сравнение языков Ruby, PHP и Python по другим параметрам, некоторые из которых влияют на процесс разработки продукта.

Таблица 1 – Сравнение Ruby, PHP и Python

Параметр	Python	PHP	Ruby
Год релиза	1991	1995	1995
Сложность изучения	Самый простой	Сложнее Python	Самый сложный
Объектно-ориентированность	Да	Да	Да
Автоматический сборщик «мусора»	Да	Да	Да
Возможность компиляции	Да	Да	Да
Множественное наследование	Да	Нет	Нет
Наличие различных библиотек	Да	Да	Да
Многомерные массивы	Да	Да	Да
Наличие различных фреймворков	Да, больше, чем у конкурентов	Да	Да
Многомерные массивы	Да	Да	Да
Удобство поддержки кода	Самый удобный	Самый неудобный	Удобнее PHP

Исходя из сравнения языков программирования было принято решение использовать язык программирования Python. Этот язык наиболее прост в изучении, а также обладает хорошей интеграцией с СУБД.

1.5 Анализ фреймворков языка Python

После того как языком программирования был выбран Python необходимо выбрать фреймворк для создания веб-сайтов. Фреймворк облегчает разработку программного продукта и объединяет разные компоненты программной системы.

Для выбора фреймворка необходимо проанализировать существующие средства разработки веб-сайтов на языке Python. Существует несколько решений: Django, Flask, Pyramid и web2py. Однако, Django и Flask являются

наиболее популярными среди разработчиков, поэтому анализировать и сравнивать необходимо именно эти фреймворки.

1.6 Сравнение Django и Flask

Flask создан австрийским разработчиком Армином Ронахером в 2010 году. Django – фреймворк, который разработан Django Software Foundation и выпущен в 2005 году. Стоит отметить, что оба фреймворка являются бесплатными с открытым программным кодом [2, 3].

Необходимо выделить основные различия между Flask и Django [4]. Во-первых, Django чаще всего используют при создании сложных веб-сайтов, в то время как Flask предназначен для небольших веб-сайтов с одной-двумя функциями.

Во-вторых, при знакомстве с новыми средствами разработки необходимо обратить внимание на их документацию. Документация Django намного подробнее и ее больше, чем документация Flask. Это говорит о том, что процесс изучения Django будет понятнее и легче.

В-третьих, Django имеет встроенную ORM – виртуальную объектную базу данных, которая позволяет работать с различными видами баз данных. Flask имеет возможность подключения сторонней ORM, например, SQLAlchemy, но всегда удобнее использовать уже готовое встроенное решение.

В-четвертых, при создании сайта с помощью Django автоматически создается строго структурированная система директорий. Используя Flask, разработчик сам создает структуру своего сайта. Может показаться, что в данном случае удобнее и проще использовать Flask, но это не так. В случае создания сложного сайта с множеством функций с помощью Flask гораздо труднее структурировать код в целом при отсутствии специальных средств для этой цели.

Одним из самых мощных преимуществ Django является автоматически создаваемый интерфейс администратора базы данных. С помощью данного интерфейса доверенные пользователи могут удобно управлять базой данных веб-сайта.

Из сравнения Django и Flask видно, что намного лучше для решения задачи подходит фреймворк Django, поэтому он и будет использоваться при разработке веб-сайта.

2 Основные строительные блоки Django

На традиционном информационном веб-сайте веб-приложение ожидает HTTP-запросы от веб-браузера (или другого клиента). Когда запрос получен, приложение разрабатывает то, что необходимо на основе URL-адреса и, возможно, данных в POST или GET запросах. В зависимости от того, что требуется, далее он может читать или записывать информацию из базы данных или выполнять другие задачи, необходимые для удовлетворения запроса. Затем приложение вернёт ответ веб-браузеру, часто динамически создавая HTML-страницу для отображения в браузере, вставляя полученные данные в HTML-шаблон [4].

Веб-приложения, написанные на Django, обычно группируют код, который обрабатывает каждый из этих шагов, в отдельные файлы (Рисунок 1):

– URLS: хотя можно обрабатывать запросы с каждого URL-адреса с помощью одной функции, гораздо удобнее писать отдельную функцию для обработки каждого ресурса. URL-маршрутизатор используется для перенаправления HTTP-запросов в соответствующее представление на основе URL-адреса запроса. Кроме того, URL-маршрутизатор может извлекать данные из URL-адреса в соответствии с заданным шаблоном и передавать их в соответствующую функцию отображения (view) в виде аргументов.

– View: view (англ. «отображение») – это функция обработчика запросов, которая получает HTTP-запросы и возвращает ответы. Функция view имеет доступ к данным, необходимым для удовлетворения запросов, и делегирует ответы в шаблоны через модели.

– Models: модели представляют собой объекты Python, которые определяют структуру данных приложения и предоставляют механизмы для управления (добавления, изменения, удаления) и выполнения запросов в базу данных.

– Templates: template (англ. «шаблон») – это текстовый файл, определяющий структуру или разметку страницы (например HTML-

страницы), с полями для подстановки, которые используются для вывода актуального содержимого. View может динамически создавать HTML-страницы, используя HTML-шаблоны и заполняя их данными из модели (model). Шаблон может быть использован для определения структуры файлов любых типов, не обязательно HTML.

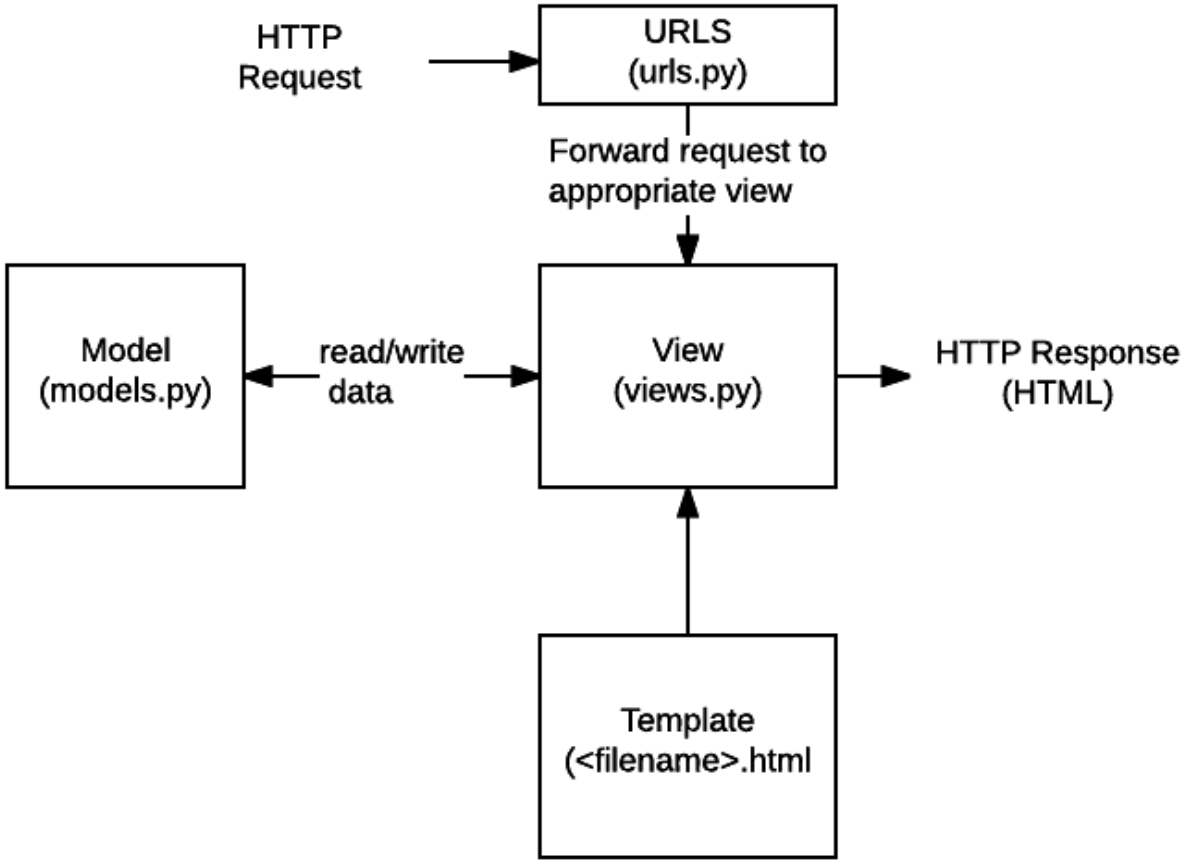

Рисунок 1 – Схема файлов кода

Django реализует уровневую архитектуру "Model View Template (MVT)". Она имеет много общего с более известной архитектурой Model View Controller.

Следующие разделы дадут понимание того, как выглядят основные части Django.

2.1 Отправка запроса в правильное view (urls.py)

Сопоставитель URL-адресов обычно содержится в файле urls.py. В примере ниже сопоставитель (urlpatterns) определяет список сопоставлений между маршрутами (определёнными URL-шаблонами) и соответствующими функциями отображения (view). Если получен HTTP-запрос, который имеет URL-адрес, соответствующий определённому шаблону, то затем будет вызвана связанная функция отображения (view) и передана в запрос (Рисунок 2).

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('book/<int:id>/', views.book_detail, name='book_detail'),
    path('catalog/', include('catalog.urls')),
    re_path(r'^([0-9]+)/$', views.best),
]
```

Рисунок 2 – Скриншот 1

Объект urlpatterns является списком функций path() и/или re_path() (в Python списки определяются с помощью квадратных скобок, внутри которых элементы разделены запятыми и могут содержать необязательную завершающую запятую. Например: [item1, item2, item3,]).

Первый аргумент в обоих методах - маршрут (шаблон), который будет сопоставлен. В методе path() угловые скобки используются для определения частей URL-адреса, которые будут захвачены и переданы в функцию отображения (view) в качестве именованных аргументов. Функция re_path() использует гибкий подход к сопоставлению с шаблоном, известный как регулярное выражение.

Второй аргумент – это ещё одна функция, которая будет вызываться при сопоставлении шаблона. Обозначение views.book_detail указывает, что

функция называется book_detail() и может быть обнаружена в модуле с именем views (т.е. внутри файла с именем views.py).

2.2 Обработка запроса (views.py)

Отображения (views) – это сердце веб-приложения, принимающего HTTP-запросы от веб-клиентов и возвращающего HTTP-ответы. Между этим они используют другие ресурсы фреймворка для доступа к базам данных, шаблонам визуализации и т. д.

В приведённом ниже примере показана минимальная функция представления index(), которая могла быть вызвана нашим сопоставителем URL-адресов в предыдущем разделе. Как и все функции отображения (view), она получает объект HttpRequest в качестве параметра (request) и возвращает объект HttpResponse. В этом случае ничего не делаем с запросом, и ответ просто возвращает жёстко запрограммированную строку запрос (Рисунок 3).

```
## filename: views.py (Django view functions)

from django.http import HttpResponse

def index(request):
    # Получить HttpRequest – параметр запроса
    # Выполнить операции, используя информацию из запроса.
    # Вернуть HttpResponse
    return HttpResponse('Hello from Django!')
```

Рисунок 3 – Скриншот 2

Модули Python – это библиотеки функций, сохранённые в различных файлах, которые можно использовать в коде. Здесь импортируем только объект HttpResponse из модуля django.http, чтобы использовать его в нашем

отображении (view): from django.http import HttpResponse. Также есть другие способы импортирования некоторых или всех объектов модуля.

Функции объявляются с помощью ключевого слова def, как показано выше, с именованными параметрами, перечисленными в скобках после имени функции; строка завершается двоеточием. Следующие строки содержат отступы. Отступы важны, так как они определяют, какие строки кода находятся внутри конкретного блока (обязательные отступы – это ключевая особенность Python и одна из причин, почему код на Python так легко читать). Отображения (view) обычно содержатся в файле views.py.

2.3 Определение данных модели (models.py)

Веб-приложения Django обрабатывают и запрашивают данные через объекты Python, называемые моделями. Модели определяют структуру хранимых данных, включая типы полей и, возможно, их максимальный размер, значения по умолчанию, параметры списка выбора, текст справки для документации, текст меток для форм и т. д. Определение модели не зависит от используемой базы данных – модели будут работать в любой из них. После того как выбрали базу данных, которую хотим использовать, не нужно напрямую обращаться к ней – просто пишем свою структуру модели и другой код, а Django выполняет всю «грязную работу» по обращению к базе данных за нас.

В приведённом ниже фрагменте кода показана очень простая модель Django для объекта Team. Класс Team наследуется от класса models.Model. Он определяет имя команды и командный уровень в качестве полей символов и задаёт максимальное количество символов, которые могут быть сохранены для каждой записи. Team_level может быть одним из нескольких значений, поэтому определяем его как поле выбора и предоставляем сопоставление между отображаемыми вариантами и хранимыми данными вместе со значением по умолчанию (Рисунок 4).

```python
# filename: models.py

from django.db import models

class Team(models.Model):
    team_name = models.CharField(max_length=40)

    TEAM_LEVELS = (
        ('U09', 'Under 09s'),
        ('U10', 'Under 10s'),
        ('U11', 'Under 11s'),
        ...  #список других командных уровней
    )
    team_level = models.CharField(max_length=3,choices=TEAM_LEVELS,default='U11')
```

Рисунок 4 – Скриншот 3

Python поддерживает «объектно-ориентированное программирование», то есть стиль программирования, в котором мы организуем наш код в объекты, которые включают связанные данные и функции для работы с этими данными. Объекты также могут наследовать / расширять / выводить из других объектов, позволяя использовать одинаковое поведение между связанными объектами. В Python используем ключевое слово class, чтобы определить «скелет» для объекта. Можем создать несколько конкретных экземпляров типа объекта на основе модели в классе.

Так, например, имеем класс Team, который происходит от класса Model. Это означает, что эта модель будет содержать все методы модели, но также можем дать ей специализированные возможности. В нашей модели определяем поля нашей базы данных, в которой будем хранить данные, присваивая им конкретные имена. Django использует эти определения, включая имена полей, для создания основной базы данных.

2.4 Запросы данных (views.py)

Модель Django предоставляет простой API запросов для поиска в базе данных. Поиск может осуществляться по нескольким полям одновременно, используя различные критерии (такие как exact («точный»), case-insensitive («без учёта регистра»), greater than («больше чем») и т. д.), и может поддерживать сложные выражения (например, можем указать поиск в командах U11, у которых есть имя команды, начинающееся с «Fr» или заканчивается на «al»).

Фрагмент кода показывает функцию view (обработчик ресурсов) для отображения всех команд U09. Выделенная жирным строка показывает, как мы можем использовать модель API-запросов для того, чтобы отфильтровать все записи, где поле team_level в точности содержит текст 'U09' (эти критерии передаются функции filter() в качестве аргумента с именем поля и типом соответствия, разделённым двойным подчёркиванием: team_level__exact) (Рисунок 5).

```
## filename: views.py

from django.shortcuts import render
from .models import Team

def index(request):
    list_teams = Team.objects.filter(team_level__exact="U09")
    context = {'youngest_teams': list_teams}
    return render(request, '/best/index.html', context)
```

Рисунок 5 – Скриншот 4

Данная функция использует функцию render() для того, чтобы создать HttpResponse, который будет отправлен назад браузеру. Эта функция

является ярлыком; она создаёт HTML-файл, комбинируя указанный HTML-шаблон и некоторые данные для вставки в шаблон (предоставляется в переменной с именем «context»). В следующем разделе покажем как данные вставляются в шаблон для создания HTML-кода.

2.5 Вывод данных (HTML-шаблоны)

Системы шаблонов позволяют указать структуру выходного документа, используя заполнители для данных, которые будут вставлены при генерировании страницы. Шаблоны часто используются для создания HTML, но также могут создавать другие типы документов. Django «из коробки» поддерживает как собственную систему шаблонов, так и другую популярную библиотеку Python под названием Jinja2 (она также может быть использована для поддержки других систем, если это необходимо).

Фрагмент кода показывает, как может выглядеть HTML-шаблон, вызванный функцией render() из предыдущего раздела. Этот шаблон был написан с предположением, что во время отрисовки он будет иметь доступ к переменной списка, названной youngest_teams (содержащейся в контекстной переменной внутри функции render() выше). Внутри скелета HTML имеем выражение, которое сначала проверяет, существует ли переменная youngest_teams, а затем повторяет её в цикле for. При каждом повторе шаблон отображает значение team_name каждой команды в элементе <li> (Рисунок 6).

```
## filename: best/templates/best/index.html

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Home page</title>
</head>
<body>
  {% if youngest_teams %}
    <ul>
      {% for team in youngest_teams %}
        <li>{{ team.team_name }}</li>
      {% endfor %}
    </ul>
  {% else %}
    <p>No teams are available.</p>
  {% endif %}
</body>
</html>
```

Рисунок 6 – Скриншот 5

2.6 Постановка задачи

В дальнейшем необходимо разработать сайт, который с помощью некоторых алгоритмов будет подбирать благозвучные мужские и женские имена под введённое пользователем отчество. Для этого понадобится база данных с мужскими и женскими именами. В Django уже реализовано большое количество классов, которые облегчат работу с базой данных, поэтому данный фреймворк как нельзя лучше подходит для осуществления поставленной задачи.

При вводе отчества и указании пола сайт будет проверять все мужские или женские имена из базы данных на соответствие следующим правилам:

– имя сына не должно совпадать с именем отца;

– первая буква отчества не должна повторять последнюю букву имени;

– имя не должно заканчиваться на тот же слог, с которого начинается отчество;

– необходимо избегать скопления согласных звуков на стыке имени и отчества;

– в имени и отчестве была только одна "р";

– если отчество длинное, то имя должно быть короткое, и наоборот;

– необходимо учитывать твердые и мягкие звуки. Если отчество мягкое, то имя должно быть твердым, и наоборот.

Удовлетворяющие всем правилам имена будут выведены пользователю как наиболее благозвучные под данное отчество.

ЗАКЛЮЧЕНИЕ

В результате проделанной работы нами была достигнута поставленная цель – изучить языки программирования для создания сайтов, познакомиться с их разновидностями, а также со средами для их реализации. Для выполнения поставленной цели были решены следующие задачи:

– Проведен анализ существующих языков программирования для создания сайтов и выбран оптимальный;

– изучены фреймворки выбранного языка программирования и выбран оптимальный;

– рассмотрены основные конструкции выбранного фреймворка.

Далее планируется разработать сайт для подбора благозвучного имени под введённое пользователем отчество.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Колисниченко Д. Н. PHP и MySQL. Разработка Web-приложений / Д. Н. Колисниченко. – СПб.: БХВ-Петербург, 2015. – 593 с.

2. Сафронов М. Н. Разработка веб-приложений в Yii 2 / М. Н. Сафронов. – М.: ДМК-Пресс, 2015. – 392 с.

3. Потапова Н. В. PHP, RUBY, PYTHON – на чем остановить свой выбор студенту при изучении веб-разработки / Н. В. Потапова. – СПб.: БХВ-Петербург, 2017. – 413 с.

4. Флэнаган Д. Язык программирования Ruby / Д.Флэнаган. – СПб.: Питер, 2011. – 496 с.