**Введение**

В современном мире информация имеет определенную, а часто и очень высокую ценность. В связи с этим ее необходимо различными способами защищать.

Одной из наиболее распространенных причин потери информации является несанкционированный доступ к ней. К мероприятиям по защите информации от несанкционированного доступа относят такие мероприятия, как защита данных паролем, шифрование, двухфакторная аутентификация и другие. В связи с тем, что технологии не стоят на месте, ни идеального, ни универсального способа защиты данных не существует.

Одной из наиболее актуальных проблем в сети интернет являются так называемые боты.

Бот – это специальная программа, выполняющая некоторые рутинные действия, такие, как регистрация нового пользователя, рассылка комментариев и сообщений, сбор данных и т.д.

К основным вредоносным действиям относят:

Сбор личных данных, электронных адресов и мобильных телефонов из контактных форм;

DoS- и DDoS-атаки;

Некоторые вирусы;

Перегрузка интернет-каналов потоками ненужной информации;

Одним из наиболее популярных методов противодействия интернет-ботам является методология CAPTCHA.

CAPTCHA – автоматизированный публичный тест Тьюринга для различения компьютеров и людей. Это аудиовизуальный инструмент, с помощью которого администраторы защищают сайты от автоматических регистраций на сайтах интернет-ботов и выполнения ими поставленных недоброжелателями целей.

Актуальность проблемы противодействия интернет-ботам с развитием и популяризацией интернет ресурсов становится все острее, так как технологии активно развиваются и злоумышленники придумывают новые способы кражи информации и нанесения различного рода вреда.

Целью работы является написание одного из вариантов алгоритма CAPTCHA.

Для выполнения проекта были поставлены следующие задачи:

* Рассмотреть понятие CAPTCHA;
* Рассмотреть основные виды технологии CAPTCHA;
* Изучить инструменты редактирования кода для веб-разработки;
* Реализовать действующее веб-приложение с применением одного из алгоритмов CAPTCHA.

**Глава 1. Теоретическая часть**

* 1. Основные методологии защиты информации

Во избежание потери данных люди, работающие с информацией, применяют различные методы защиты. Рассмотрим некоторые из них:

**Технические средства защиты информации**

Технические средства защиты данных совмещают в себе аппаратные и программные средства. К ним относят:

* Резервное копирование данных;
* Удаленное хранение данных;
* Обеспечение возможности использовать резервные системы электропитания;
* Обеспечение как противопожарной безопасности, так и защиты оборудования от повреждения водой;
* Установка программного обеспечения, ограничивающего доступ к данным и защищающего от несанкционированного доступа.

**Программные средства защиты информации**

Программные средства включают в себя программное обеспечение, с помощью которого реализуется контроль над доступом к информации, а также ее защита. Некоторые из них:

* Программы для идентификации и аутентификации пользователей;
* Программы для шифрования данных;
* Программы для удаления остаточной информации;
* Генераторы паролей;
* Хранилища паролей;
* Методы защищенного обмена информации.
  1. Основные технологии создания веб-приложения

При создании любого веб-приложения обязательно используют HTML, CSS, JavaScript и PHP. При помощи HTML, CSS и JavaScript (или их аналогов) структуру, внешний вид и динамику интерфейса, а также взаимосвязь между его элементами, если она должна присутствовать в приложении.

На языке PHP реализуют взаимосвязь между клиентом и сервером, на котором расположен интернет-ресурс, связь приложения с базой данных и проводят вычислительные операции.

* 1. Анализ основных технологий CAPTCHA

**Распознавание защищенного текста**

Распознавание делится на 2 основных этапа:

Определение местоположения и границ каждого символа;

Собственно распознавание символа.

Если символы постоянно расположены на одних и тех же позициях, из написанных выше этапов остается только второй. Следовательно, координаты символов нужно изменять динамически.

Также, для повышения уровня защищенности, символы не должны по цвету сильно отличаться от цвета фона. По возможности, символы должны быть разных цветов.

Таким образом, необходимо создать визуальный шум, трудноотделимый от символов, либо размещать символы близко друг к другу или накладывая их друг на друга.

Для того, чтобы алгоритмам автоматического распознавания было труднее расшифровывать текст, можно также применять для генерации надписи различные шрифты, так как автоматизированные системы распознавания нередко используют наложение маски и эталонных шрифтов.

**Аудио-CAPTCHA**

Аудио-капча предлагает пользователю прослушать некоторую фразу, набор букв или цифр, после чего ее ввести. Фраза обычно произносится с изменяемой тональностью, паузами и фоновым шумом. На такую капчу сможет ответить человек с нарушениями зрения. Но пользователь должен иметь оборудование для воспроизведения звука.

Создание звуковых шумов требовательна к ресурсам сервера. Также, ее трудно реализовать недостаточно квалифицированному программисту.

Из-за трудностей реализации и требований к ресурсам сервера данную капчу применяют редко.

Одним из видов аудио-капчи являются задаваемые пользователю задачи и загадки. Ответ чаще всего предлагают выбрать из списка, или ввести в поле. Но выбор ответа из предложенного списка увеличивает вероятность правильного ответа, так как правильный ответ – обязательно есть в списке. Поэтому пользователю задают не один вопрос, а несколько.

Еще одним минусом данного метода является необходимость посетителю сайта быть ознакомленным с алфавитом и языком, на котором задаются вопросы.

**Распознавание предметов**

При использовании данного метода пользователя просят узнать показанные на картинке (или картинках) предметы.

Данный метод делят на два вида:

Показ на картинке предмета и просьба ввести в поле его название;

Указание пользователю названия предмета и просьба выбрать несколько изображений, на которых он изображен, из списка.

Пользователю легче отличить один предмет от другого, чем сильно искаженные, похожие друг на друга буквы. Боту – наоборот. Для автоматического распознавания злоумышленники чаще всего должны прибегать к использованию нейронных сетей. Нейросети сильно ограничены в функциональности и требовательны к ресурсам компьютера злоумышленника, что затрудняет их применение, поэтому данный метод можно считать наиболее безопасным из рассмотренных.

**Методы, позволяющие усложнить распознавание**

Одним из дополнительных способов усложнения разгадывания капчи автоматически является ограничение количества запросов с одного IP-адреса. Данный метод прост и малоэффективен из-за развития прокси и подставных IP-адресов. Ограничение через cookie тоже неэффективно, так как многие пользователи их игнорируют или удаляют и получают возможность новых попыток.

Благодаря тому, что пользователь при заполнении форм заполняет только те поля, которые видит и смотрит на визуальную часть сайта, а не на разметку, был разработан еще один метод – создание невидимых пользователь полей ввода. Боты обычно смотрят в разметку страницы и обнаруживают поля, которые автоматически заполняют. Это позволяет идентифицировать бота и ограничить ему доступ к сайту.

Также можно ограничить ввод данных по времени, так как компьютер всегда тратит на работу и вывод данных меньше времени, чем человек, этим себя выдавая.

**Глава 2. Практическая часть**

2.1. Выбор редактора кода

Для разработки данного веб-приложения был выбран Visual Studio Code. Данный редактор полностью бесплатен. Также, он наделен большей частью необходимых функций:

* Поддержка большинства популярных языков программирования;
* Подсветка синтаксиса;
* Возможность одновременной работы с несколькими проектами;
* Интегрирована система подсказок;
* Обширная библиотека плагинов, позволяющих расширить функционал редактора (Emmet – значительно ускоряет написание текста на языках HTML, CSS, JS и другие).

Существует множество других редакторов кода и сред разработки для разработки приложений:

* WebStorm – полноценная среда разработки веб-приложений. Множество полезных функций, таких как работа с git через удобный интерфейс, умный анализатор кода и плагины идут «из коробки». Однако, все это не бесплатно. Пользователь должен оформить подписку, либо получить студенческую лицензию;
* Atom – приятный и простой в использовании редактор. Поддерживает все основные языки программирования. Главные недостатки – потребляет много ресурсов системы и не имеет большого количества плагинов, сложнее настроить под себя;
* Sublime Text – некогда популярная программа, высокая производительность и огромное количество уроков по настройке под различные проекты. Однако сейчас его функционал не настолько широк, как у того же VS Code. Подойдет для создания простых сайтов.
* PHPStorm – среда разработки от компании JetBrains, имеющая значительно расширенный функционал среды WebStorm: поддержку языка PHP и большинства его библиотек и фреймворков. Платный.

2.2. Структура и функционал приложения

После того, как была определена среда разработки, необходимо определить структуру и функционал приложения. Ранее упоминалось о том, что структура сайта должна быть простой и дружелюбной в использовании.

В нашем случае структура приложения состоит из одной веб-страницы с формой и текстовым блоком CAPTCHA, которая при правильно заполненном поле капчи сообщает, что все введено верно. В противном случае выводится сообщение об ошибке с просьбой повторить попытку ввода.

Далее определим функционал приложения:

* Повторная генерация изображения с текстом капчи с целью избежания повтора задачи и снижения риска прохождения ее ботом.

2.3. Проектирование интерфейса

Определим состав страницы нашего приложения:

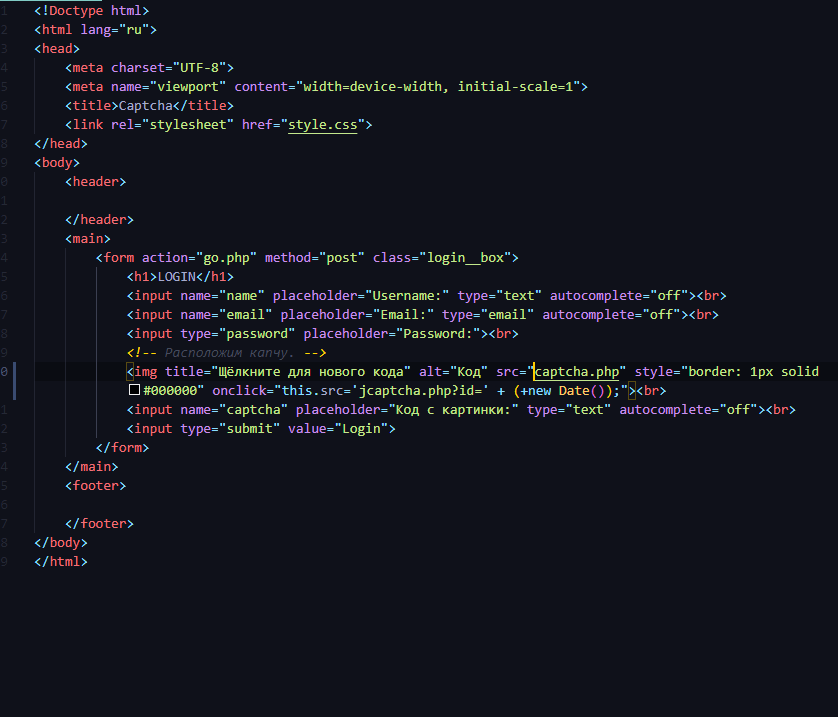
Чтобы показать визуально цель применения капчи, страница будет состоять из формы, включающей в себя такие поля, как:

* Поле ввода логина пользователя;
* Поле ввода электронной почты;
* Поле ввода пароля;
* Генерируемую картинку с капчей;
* Поле ввода для самой капчи.

2.4. Процесс разработки

После проектирования функционала и структуры приложения, а также его интерфейса, следует процесс написания кода.

Для начала создаем HTML-разметку. Необходимо верстать пустую страницу, в которую поместить форму.



Данный код представляет собой четыре поля формы, описанные ранее и кнопку, при нажатии на которую начинается проверка введенных данных, а также контентные блоки header, main и footer, не несущие в данном случае особой нагрузки, но предусмотренных HTML5 как основные.

Чтобы наша верстка приобрела привычный для пользователя вид, нужно добавить стили для страницы и ее элементов.

html {

height: 100%;

}

body {

font-family: sans-serif;

background: linear-gradient(-50deg, #970d1a, #4e204d);

height: 100%;

display: flex;

justify-content: center;

align-items: center;

overflow-y: hidden;

overflow-x: hidden;

}

.login\_\_box {

height: auto;

width: 300px;

padding: 40px;

/\* transform: translate(-50%, -50%); \*/

background: #191919;

text-align: center;

background: linear-gradient(-50deg, #970d1a, #4e204d);

box-shadow: 0 0 10px rgba(0,0,0,0.5);

border-radius: 5px;

}

.login\_\_box h1 {

color: #ffffff;

text-transform: uppercase;

font-weight: 500;

}

.login\_\_box input[type = "text"],

.login\_\_box input[type = "password"],

.login\_\_box input[type = "email"] {

border: 0;

background: none;

display: block;

margin: 8px auto;

text-align: center;

border: 2px solid #3498db;

padding: 14px 10px;

width: 200px;

outline: none;

color: #ffffff;

border-radius: 24px;

transition: 0.25s;

}

.login\_\_box input[type = "text"]:focus,

.login\_\_box input[type = "password"]:focus,

.login\_\_box input[type = "email"]:focus {

width: 280px;

border-color: #2ecc71;

}

.login\_\_box input[type = "submit"] {

border: 0;

background: none;

display: block;

margin: 15px auto;

text-align: center;

border: 2px solid #2ecc71;

padding: 14px 40px;

width: 200px;

outline: none;

color: #ffffff;

border-radius: 24px;

transition: 0.25s;

cursor: pointer;

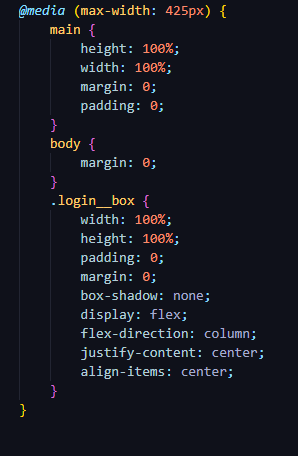
}

.login\_\_box input[type = "submit"]:hover {

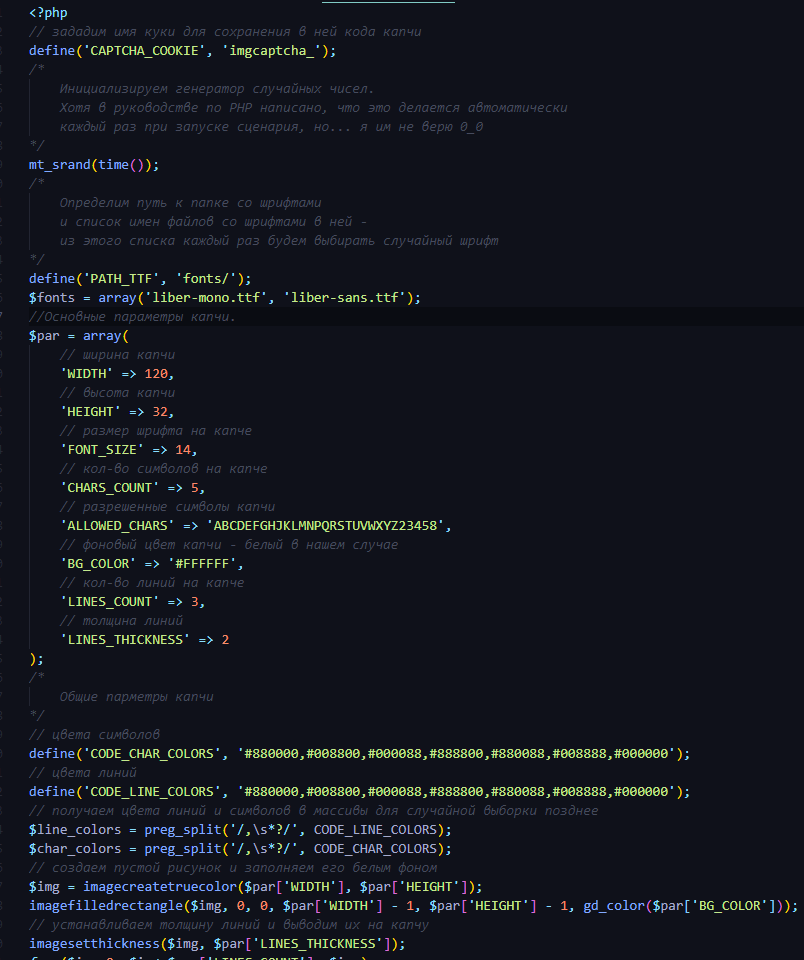
background: #2ecc71;

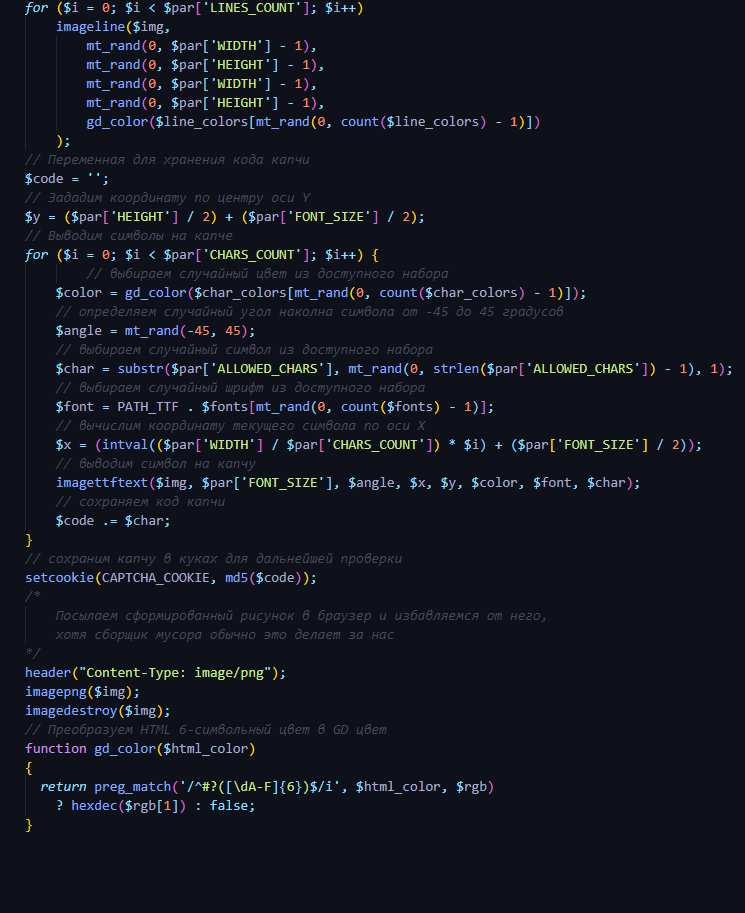
}

Далее нужно предусмотреть открытие пользователем страницы с мобильного устройства, то есть добавить на страницу адаптив.

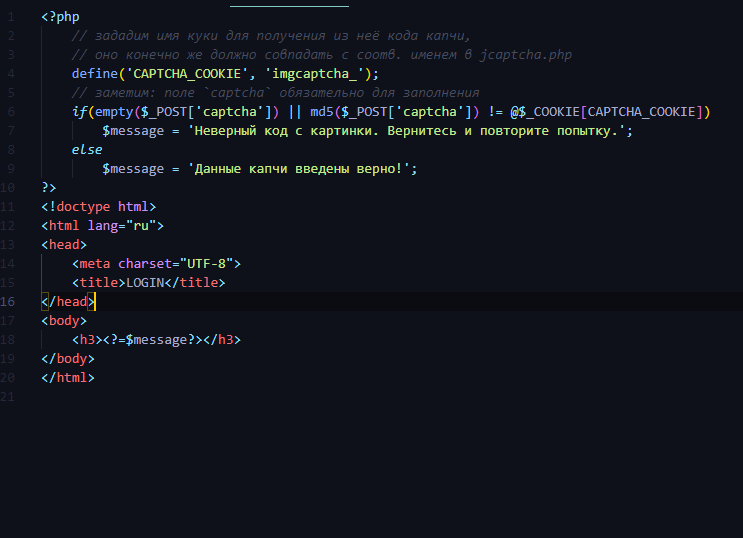


После завершения базовой разметки и стилизации, необходимо написать алгоритм для генерации изображения капчи при помощи языка PHP. Создаем файл captcha.php.





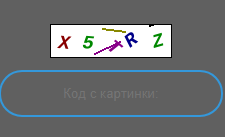
После создания текста капчи нужно написать алгоритм ее проверки после введения текста пользователем. Для этого создаем файл с именем go.php.



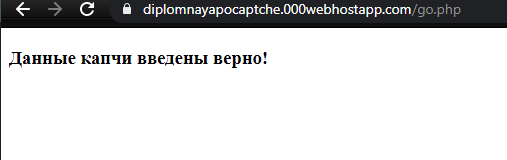
Далее нужно реализовать или саму генерацию капчи, или разместить наше приложение на стороннем сервисе с целью проверки работы нашего алгоритма.

В связи с сильно ограниченными ресурсами было выбрано размещение приложения на бесплатном хостинге. Из возможных вариантов было решено использовать хостинг “000webhost.com”, так как данный хостинг имеет простую панель администратора и не имеет отзывов о потере данных, что указывает на его надежность.

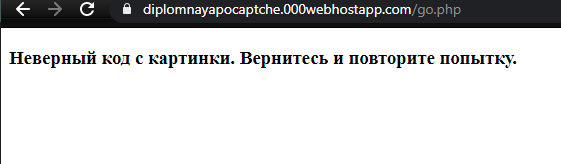
Проверяем работу нашей капчи:



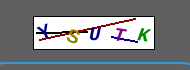
В случае правильно введенного текста высвечивается сообщение об успешном завершении:



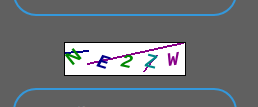
Проверим поведение приложения при неправильно введенном тексте капчи:



Также, несколько раз проверим поведение приложения при клике по картинке капчи в форме. Текст в изображении должен меняться. При множественных нажатиях текст становиться менее удобочитаемым не должен. Несколько вариантов перегенерируемого текста:



Капча после 5 попытки заменить текст.



Капча после 27 попытки.

При дальнейших повторных генерациях заметных ухудшений в читаемости текста не наблюдалось.

**2.5. Перспектива развития**

Как и любое программное обеспечение, данный алгоритм метода CAPTCHA можно развивать как в плане безопасности и оптимизации, так и в плане функционала.

Из функций можно добавить поддержку различных шрифтов для усложнения подбора маски к тексту, а также поддержку применения нескольких шрифтов за одну генерацию одновременно.

Из оптимизаций – использовать современные стандарты верстки и оформления кода с целью упрощения его поддержания и изменения.

Для повышения безопасности алгоритма и приложения можно переписать код с применением современных библиотек или фреймворков. Это также позволит сократить объем памяти, который занимает код приложения. Можно применить дополнительные алгоритмы шифрования и распределенное хранение данных чтобы усложнить доступ к ним извне. Это также защитит данный алгоритм.

Идей по доработке и добавлению новых возможностей много. Однако, так как CAPTCHA – это алгоритм защиты данных от несанкционированного доступа, улучшение его является не менее важным.

**Заключение**

Итогом выполненной работы является веб-приложение и действующий алгоритм, имитирующий работу алгоритма CAPTCHA. В приложении предусмотренны:

* Генерация текста, состоящего из букв латиницы;
* Проверка вводимых пользователем символов на соответствие со сгенерированным текстом;
* Возможность использования приложения с мобильного устройства.

В процессе проектирования были изучены редакторы кода веб-разработки – WebStorm, Sublime Text, Atom, PHPStorm и Visual Studio Code.

Разработка приложения выполнена с помощью языка PHP.

Получившийся проект полностью работоспособен, прошел этап тестирования и готов к работе.

Проект имеет следующие особенности:

* Простой функционал – пользователям для работы с ним достаточно обычной клавиатуры с поддержкой латинского алфавита;
* Приложение способно функционировать на любом устройстве, имеющем браузер и доступ к сети интернет.

Таким образом, следует отметить:

* Поставленные задачи выполнены,
* Итоговая цель достигнута.

**Список литературы**