

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего
образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»
Профиль подготовки: «Вычислительные машины, комплексы, системы и сети»

БАКАЛАВРСКАЯ РАБОТА

на тему

ВЕБ-ПРИЛОЖЕНИЕ ОПРЕДЕЛЕНИЯ
СТОЙКОСТИ ПОЛЬЗОВАТЕЛЬСКИХ
ПАРОЛЕЙ

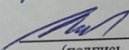
Студент

 28.05.2026 Уткин Михаил Максимович
(подпись, дата) (ФИО полностью)

Руководитель

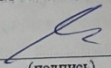
 28.05.2026 Токарев А.Н.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр

 28.05.2026 Бычков А.С.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 22.05.2026 г. № 14)

Заведующий кафедрой

 Митрохин М.А.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой хорошо (протокол заседания ГЭК от 11.06.2026 № 7)

Секретарь ГЭК

 Гурин Е.И.
(подпись) (фамилия, инициалы)

Пенза 2026

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«7» апреля 2026 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Уткин Михаил Максимович гр. 22ВВВ1 факультета ФВТ

2. Тема работы: «Веб-приложение определения стойкости пользовательских паролей»

Тема утверждена приказом ПГУ № 347/о от «7» апреля 2026 г.

3. Руководитель бакалаврской работы к.т.н., доцент кафедры ВТ ПГУ Токарев А.Н.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать «Веб-приложение определения стойкости пользовательских паролей». Приложение должно обладать графическим и интуитивно понятным интерфейсом для взаимодействия с пользователем. Система должна предоставлять возможность анализа пользовательских паролей с учетом их длины, структуры, используемых символов и наличия распространенных шаблонов. Разрабатываемое приложение должно выполнять вычисление энтропии пароля, проверку по локальным базам утечек, оценку времени подбора при различных типах атак, а также формировать рекомендации по повышению уровня безопасности. Дополнительно система должна поддерживать генерацию стойких паролей и использование модуля машинного обучения для выявления предсказуемых комбинаций.

5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметной области оценки стойкости пользовательских паролей: изучить существующие решения, выделить их достоинства и недостатки. Выбрать язык программирования, серверный фреймворк и технологии для создания графического интерфейса. Определить архитектуру приложения и вспомогательные средства разработки. Изучить методы и подходы для анализа паролей. Реализовать модуль машинного обучения. Разработать пользовательский интерфейс и обеспечить взаимодействие клиентской и серверной частей приложения.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	30	07.04.2026 – 10.04.2026	
2	Анализ аналогов, обоснование актуальности, постановка задачи	24	10.04.2026 – 13.04.2026	
3	Выбор и настройка инструментов разработки	8	13.04.2026 – 14.04.2026	
4	Проектирование архитектуры приложения	40	14.04.2026 – 20.04.2026	
5	Разработки модуля анализа стойкости пароля	60	20.04.2026 – 28.04.2026	
6	Разработка модуля проверки по локальным базам утечек	45	28.04.2026 – 07.05.2026	
7	Разработка модуля генерации пользовательских паролей	30	07.05.2026 – 13.05.2026	
8	Разработка модуля машинного обучения	30	13.05.2026 – 15.05.2026	
9	Реализация пользовательского интерфейса	17	15.05.2026 – 16.05.2026	
10	Тестирование и отладка	20	16.05.2026 – 17.05.2026	
11	Оформление пояснительной записки	20	18.05.2026 – 21.05.2026	

Дата выдачи задания «07» апреля 2026г.

Руководитель бакалаврской работы
(подпись, дата)

Токарев А.Н.

Задание к исполнению принял студент
(подпись, дата)

Уткин М.М.

Работу к защите допустить
Декан факультета

Фионова Л.Р.

03.06.2026 (подпись, дата)

Реферат

Бакалаврская работа содержит 105 листов, 20 рисунков, 1 таблицу, 12 источников, 3 приложения.

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ, ПАРОЛЬ, СТОЙКОСТЬ ПАРОЛЯ,
ЭНТРОПИЯ, ВЕБ-ПРИЛОЖЕНИЕ, DJANGO, МАШИННОЕ ОБУЧЕНИЕ

Цель работы - разработать веб приложение, обеспечивающее комплексный анализ стойкости пользовательских паролей. Самым простым и эффективным путем достижения поставленной задачи является создание приложения, которое реализует объединение нескольких методов оценки стойкости, включая математический анализ, эвристические алгоритмы и методы машинного обучения. В процессе выполнения работы были рассмотрены существующие подходы к оценке стойкости паролей, проведён анализ аналогичных решений и разработана архитектура программы и реализованы основные алгоритмы анализа. Они включают вычисление информационной энтропии, выявление предсказуемых шаблонов, проверку паролей по базам утечек данных и оценку времени подбора.

В итоге было разработано веб-приложение, которое демонстрирует возможность комплексной оценки паролей, выявляет уязвимости и предоставляет пользователю рекомендации по повышению уровня безопасности. Дополнительно реализован модуль генерации надёжных паролей и модуль интеллектуального анализа. Разработанное приложение может использоваться как в практических целях для повышения безопасности учётных записей, так и в образовательной среде для формирования навыков создания надёжных паролей. Реализованный на данный момент функционал приложения протестирован и готов к эксплуатации.

					ПГУ 1.090301.13.001 ПЗ			
Изм.	Лист	№ докум	Подпись	Дата	Веб-приложение определения стойкости пользовательских паролей Пояснительная записка	Литера	Лист	Листов
Разраб		Уткин М.М.		28.05.24				
Пров		Токарев А.Н.		28.05.24			2	105
Н. Контр.		Бычков А.С.		28.05.24		ФВТ, каф. ВТ гр. 22ВВВ1		
Утв		Митрохин М.А.		28.05.24				

Реферат

Бакалаврская работа содержит 105 листов, 20 рисунков, 1 таблицу, 12 источников, 3 приложения.

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ, ПАРОЛЬ, СТОЙКОСТЬ ПАРОЛЯ, ЭНТРОПИЯ, ВЕБ-ПРИЛОЖЕНИЕ, DJANGO, МАШИННОЕ ОБУЧЕНИЕ

Цель работы - разработать веб приложение, обеспечивающее комплексный анализ стойкости пользовательских паролей. Самым простым и эффективным путем достижения поставленной задачи является создание приложения, которое реализует объединение нескольких методов оценки стойкости, включая математический анализ, эвристические алгоритмы и методы машинного обучения. В процессе выполнения работы были рассмотрены существующие подходы к оценке стойкости паролей, проведён анализ аналогичных решений и разработана архитектура программы и реализованы основные алгоритмы анализа. Они включают вычисление информационной энтропии, выявление предсказуемых шаблонов, проверку паролей по базам утечек данных и оценку времени подбора.

В итоге было разработано веб-приложение, которое демонстрирует возможность комплексной оценки паролей, выявляет уязвимости и предоставляет пользователю рекомендации по повышению уровня безопасности. Дополнительно реализован модуль генерации надёжных паролей и модуль интеллектуального анализа. Разработанное приложение может использоваться как в практических целях для повышения безопасности учётных записей, так и в образовательной среде для формирования навыков создания надёжных паролей. Реализованный на данный момент функционал приложения протестирован и готов к эксплуатации.

					ПГУ 1.090301.13.001 ПЗ						
Изм.	Лист	№ докум	Подпись	Дата	Веб-приложение определения стойкости пользовательских паролей Пояснительная записка			Литера	Лист	Листов	
Разраб		Уткин М.М									
Пров		Токарев А.Н							2	105	
Н. Контр.		Бычков А.С.						ФВТ, каф. ВТ гр. 22BBB1			
Утв		Митрохин М.А.									

Содержание

Перечень сокращений и обозначений.....	5
Введение	6
1 Веб-приложение определения стойкости пользовательских паролей.....	8
1.1 Анализ предметной области.....	8
1.2 Анализ актуальности разработки.....	10
1.3 Анализ аналогичных решений.....	11
1.4 Постановка задачи.....	15
1.5 Анализ решения поставленной задачи	17
2 Разработка.....	19
2.1 Описание архитектуры проекта.....	19
2.2 Выбор инструментов разработки.....	21
2.2.1 Выбор языка программирования и технологий разработки.....	21
2.2.2 Выбор методов и алгоритмов оценки стойкости пароля.....	22
2.2.3 Среда разработки и программные инструменты.....	24
2.3 Установка и настройка инструментов разработки.....	24
2.4 Описание структуры проекта	27
2.5 Разработка программы	32
2.5.1 Проектирование структуры веб-приложения.....	32
2.5.2 Реализация анализа стойкости пароля.....	33
2.5.3 Реализация генератора паролей	36
2.5.4 Реализация модуля машинного обучения.....	37
2.5.5 Реализация пользовательского интерфейса.....	39
2.6 Описание доступных функций.....	41
2.6.1 Анализ стойкости пароля.....	42
2.6.2 Проверка пароля на утечки данных.....	43
2.6.3 Прогноз времени подбора пароля	44
2.6.4 Интеллектуальный анализ с использованием машинного обучения	44
2.6.5 Генератор надёжных паролей.....	45

2.6.6 Формирование итогового отчёта.....	45
3 Тестирование разработанного приложения	47
4 Анализ результатов.....	52
4.1 Результаты разработки приложения	52
4.2 Оценка эффективности и ограничения системы	53
4.3 Рекомендации к дальнейшей разработке.....	54
5 Безопасность жизнедеятельности.....	56
5.1 Правила безопасности при использовании ПК.....	56
Приложение А – UML-диаграммы.....	60
Приложение Б – Листинг программы	65
Приложение В – Презентация	100

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

ИБ – информационная безопасность

ПО – программное обеспечение

API (Application Programming Interface) – программный интерфейс приложения

CPU (Central Processing Unit) – центральный процессор

CSRF (Cross-Site Request Forgery) – межсайтовая подделка запроса

CSS (Cascading Style Sheets) – каскадные таблицы стилей

GPU (Graphics Processing Unit) – графический процессор

HTML (HyperText Markup Language) – язык разметки гипертекста

HTTP (HyperText Transfer Protocol) – протокол передачи гипертекста

JSON (JavaScript Object Notation) – формат обмена данными

JS (JavaScript) – язык программирования для создания интерактивных веб-интерфейсов

ML (Machine Learning) – машинное обучение

MVT (Model–View–Template) – архитектурный шаблон веб-приложений фреймворка Django

ORM (Object-Relational Mapping) – технология объектно-реляционного отображения данных

SHA-1 (Secure Hash Algorithm 1) – криптографическая хеш-функция

Введение

Защита персональных данных пользователей одна из ключевых задач в современном мире. Одним из вариантов для ограничения доступа к информации являются механизмы аутентификации. Среди них наиболее распространённым остаётся использование паролей, несмотря на появление альтернативных способов подтверждения личности, таких как биометрия и многофакторная аутентификация. Такое широкое распространение объясняется простотой, универсальностью и отсутствием необходимости в дополнительном оборудовании. Пароли используются в веб-сервисах, корпоративных системах, мобильных приложениях и мессенджерах, однако их надёжность напрямую зависит от того какие пароли используют пользователи. На практике они довольно часто в своих паролях используют простые комбинации, которые содержат распространённые слова, последовательности символов или свою личную информацию. Даже если в них соблюдены формальные требования такие пароли могут быть уязвимыми и поддаваться подбору. Ситуация усложняется тем, что методы подбора паролей постоянно совершенствуются. К ним относятся полный перебор, словарные атаки и использование баз утекших паролей. Современные вычислительные мощности позволяют выполнять такие атаки значительно быстрее, чем раньше, из-за чего даже относительно сложные пароли могут быть взломаны за короткое время.

Для оценки надёжности паролей существуют различные подходы. Наиболее распространённые из них это расчёт энтропии, анализ структуры пароля и проверка по базам утечек, но каждый из этих методов по отдельности не всегда даёт точную оценку. Поэтому всё чаще используется их комбинация. Также одним из перспективных направлений повышения точности анализа является использование методов машинного обучения. Благодаря этому решению можно выявлять скрытые закономерности в структуре паролей. Это даёт возможность обнаруживать комбинации, которые выглядят сложными, но остаются предсказуемыми. В связи с этим разработка программных

решений,обеспечивающих комплексную оценку стойкости паролей, является актуальной задачей.

В рамках данной выпускной работы рассматривается разработка веб-приложения, предназначенного для анализа стойкости пользовательских паролей с использованием совокупности методов. Эти методы включают вычисление энтропии, анализ структуры и применение методов машинного обучения.

Для достижения поставленной цели необходимо решить следующие задачи:

- ~ исследовать существующие методы оценки стойкости паролей;
- ~ проанализировать угрозы, связанные с использованием слабых паролей;
- ~ разработать алгоритмы анализа структуры пароля;
- ~ реализовать механизм вычисления энтропии;
- ~ обеспечить проверку паролей по базам скомпрометированных данных;
- ~ разработать и применить модель машинного обучения;
- ~ реализовать веб-приложение с пользовательским интерфейсом;
- ~ провести тестирование разработанной системы;
- ~ сформулировать рекомендации по дальнейшему развитию системы.

Объектом исследования является процесс обеспечения безопасности аутентификации пользователей в информационных системах.

Предметом исследования являются методы анализа и оценки стойкости пользовательских паролей.

Практическая значимость работы заключается в создании инструмента, который помогает пользователям оценить надёжность своих паролей и при необходимости улучшить их.

1 Веб-приложение определения стойкости пользовательских паролей

1.1 Анализ предметной области

Для разработки качественного приложения необходимо провести анализ основных аспектов обеспечения парольной безопасности и существующих подходов к ее реализации. То, насколько устойчивыми оказываются пароли, которые создают пользователи, напрямую определяет безопасность учетных записей. Распространенное мнение о том, будто достаточно увеличить длину пароля и добавить разные типы символов не всегда соответствует действительности. Как показывает практика всё немного сложнее. Влияние на уровень защиты оказывает не только набор символов, но и как именно эти символы организованы внутри пароля. Если в основе пароля лежат предсказуемые шаблоны, использование букв разного регистра, цифр и специальных символов не поможет и пароль может остаться уязвимым. Такие шаблоны легко учитываются в алгоритмах подбора, что существенно упрощает задачу злоумышленнику. Отдельно обратить внимание стоит на проблему утечек данных. Регулярно происходят случаи, когда базы пользователей различных сервисов оказываются в открытом доступе. В результате в сети накапливаются огромные списки логинов и паролей, которые затем используются при атаках. Один из распространённых сценариев автоматическая проверка уже известных комбинаций на других сайтах. Из-за этого даже сложный пароль перестаёт быть надёжным, если он уже когда-то был раскрыт. Также важным инструментом, применяемым при взломе, являются так называемые радужные таблицы. Они представляют собой заранее вычисленные наборы хешей. Они позволяют значительно ускорить процесс восстановления исходного пароля несмотря на то, что современные системы защиты применяют дополнительные меры, такие как добавление «соли» к паролям, полностью исключить подобные методы атак невозможно. Это ещё раз показывает, что защита должна быть комплексной. Дополнительно следует рассмотреть основные методы подбора

					ПГУ 1.090301.13.001 ПЗ	Лист
						8
Изм.	Лист	№докум.	Подпись	Дата		

паролей, применяемые на практике. Наиболее простым вариантом является полный перебор (brute force), при котором последовательно проверяются все возможные комбинации символов. Эффективность такого подхода напрямую зависит от длины пароля и мощности используемого оборудования. Но чаще используются более продвинутые методы, одними из таких методов стали словарные атаки. При таких видах атак применяются заранее подготовленные списки популярных паролей и их возможные вариации. Типичное поведение пользователей, замену букв на символы, добавление цифр в конце и другие распространённые приёмы учитываются в них. Даже на первый взгляд сложный пароль может быть подобран достаточно быстро. Злоумышленники часто совмещают различные методы, которые включают использование утёкших баз данных и автоматизированные сценарии перебора. По этой причине при оценке стойкости пароля важно учитывать устойчивость к реальным сценариям атак. Кроме того, важную роль играет уровень технических возможностей потенциального нарушителя. С техническим развитием скорость подбора паролей значительно увеличилась. В результате требования к сложности постоянно повышаются. Решения, которые считались надёжными несколько лет назад, сегодня уже могут не обеспечивать надёжный уровень защиты[9].

Таким образом, надёжность пароля определяется не одним параметром, а их совокупностью. Она должна учитывать, как количественные параметры, такие как длину, разнообразие символов, так и качественные аспекты, включая предсказуемость структуры, уникальность и устойчивость к современным методам атак.

В результате анализа стало понятно, что «Веб-приложение для определения стойкости пользовательских паролей» может повысить уровень осведомленности пользователей о значимости выбора надежного пароля.

1.2 Анализ актуальности разработки

Актуальность разработки инструментов анализа стойкости паролей подтверждается многочисленными инцидентами утечек данных в крупных интернет-сервисах. Одним из известных примеров был инцидент с социальной сетью LinkedIn. В результате этого происшествия в открытый доступ попали более 160 миллионов паролей. Анализ этих данных показал, что значительная часть пользователей использовала простые и легко угадываемые комбинации. Похожая ситуация наблюдалась и при утечках данных компании Yahoo, где многие учётные записи оказались скомпрометированы из-за слабых или повторно используемых паролей.

Современные методы атак позволяют за секунды проверить миллионы вариантов. С учётом роста вычислительных мощностей такие атаки позволяют проверять огромное количество вариантов за очень короткое время. В результате даже те пароли, которые выглядят сложными, могут оказаться ненадёжными. В большинстве систем проверка паролей выполняется на основе простых критериев. Обычно учитываются минимальная длина пароля и необходимость использования различных типов символов. Такой подход прост в реализации, но он не всегда позволяет точно оценить уровень безопасности. Даже при полном соответствии установленным требованиям пароль может оставаться уязвимым. Причиной этого становятся очевидная структура и применение распространенных шаблонов. Существует еще одна проблема, она связана с недостаточной осведомленностью пользователей, ведь зачастую пользователи не всегда понимают, насколько надёжным является их пароль и сколько времени потребуется для его подбора. Именно поэтому результаты проверки важно представлять понятной форме. Например, показывать примерное время взлома или указывать на конкретные слабые места в пароле. Использование веб-приложения для анализа паролей обладает рядом преимуществ. Проверка может выполняться без установки дополнительного ПО, что делает такой инструмент

					ПГУ 1.090301.13.001 ПЗ	Лист
						10
Изм.	Лист	№докум.	Подпись	Дата		

удобным как для личного применения, так и в образовательных целях. Помимо итоговой оценки система способна формировать рекомендации по улучшению пароля. Такое решение помогает лучше понять пользователю принципы создания безопасных паролей. Еще одним преимуществом является доступность на различных устройствах независимо от операционной системы. Отдельной функцией системы выступает моделирование сценариев атак. С его помощью определяется примерное время, необходимое для взлома пароля при различных вычислительных возможностях злоумышленника. За счет такой визуализации различия между слабыми и надёжными паролями становятся более наглядными.

Таким образом, разработка такого приложения для комплексной оценки стойкости паролей представляет собой актуальную задачу. Разработка подобного инструмента позволяет повысить уровень информационной защищенности, снизить вероятность кражи учётных записей и сформировать у пользователей ответственное отношение к созданию надёжных паролей с учетом современных угроз.

1.3 Анализ аналогичных решений

Анализ существующих аналогичных решений является важным этапом разработки любой программы, поскольку позволяет изучить применяемые подходы к решению поставленной задачи, выявить их сильные и слабые стороны, а также выявить направления совершенствования разрабатываемой системы. В нашем случае я рассмотрел программные и онлайн-решения, которые предназначены для оценки паролей.

Первую группу аналогов составляют онлайн-сервисы проверки паролей. Они предоставляют базовую оценку их надёжности, как правило, такие сервисы анализируют длину вводимой комбинации, наличие символов различных типов, например, строчные и заглавные буквы, цифры, специальные символы, а также формируют условную шкалу сложности. Основными преимуществами данных решений являются простота использования, доступность и отсутствие

необходимости установки дополнительных программ, а также возможность работы на различных устройствах благодаря веб-интерфейсу.

Все это делает их удобными для широкого круга пользователей. Отображение аналога рассмотрено на рисунке 1.1. Вместе с тем функциональные возможности подобных сервисов ограничиваются преимущественно формальными критериями. В большинстве случаев они не учитывают анализ структуры пароля.

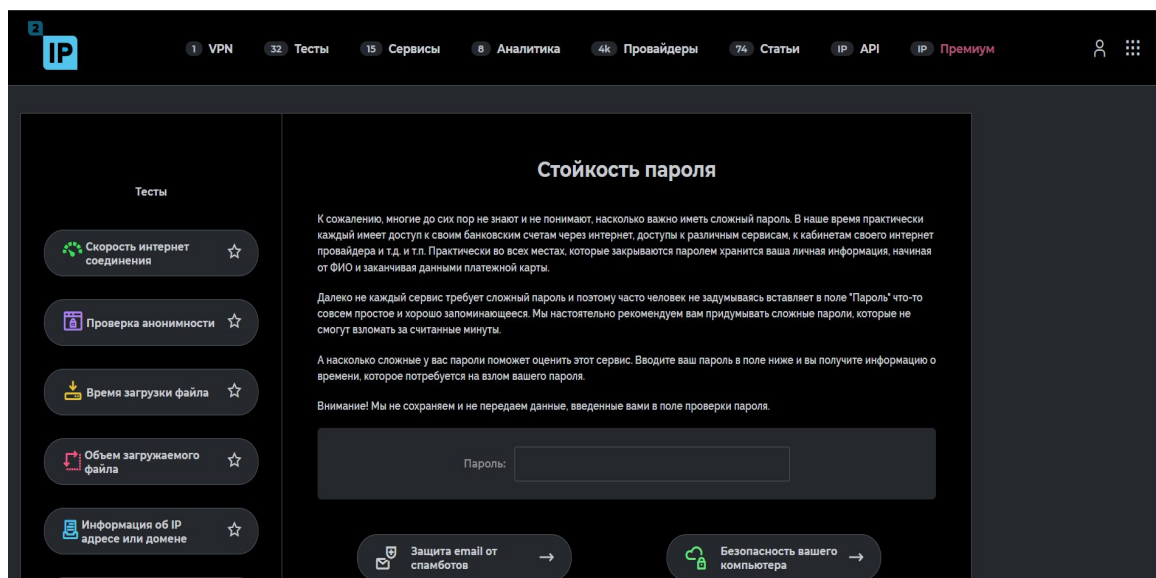


Рисунок 1.1 - Обзор аналога онлайн сервис проверки пароля

Вторую группу составляют сервисы, которые используют базы данных ранее скомпрометированных паролей, например, сервис «Have I Been Pwned». Такие инструменты позволяют определить, использовался ли введенный пароль ранее в известных утечках данных. Основным преимуществом такого подхода является высокая практическая значимость, поскольку повторное использование паролей существенно увеличивает риск утечки учетных данных. Отображение приложения рассмотрено на рисунке 1.2.

Тем не менее, подобные сервисы ориентированы преимущественно на проверку уже известных паролей и не предназначены для оценки новых комбинаций. Они не анализируют структуру пароля и не оценивают его устойчивость к различным методам подбора.

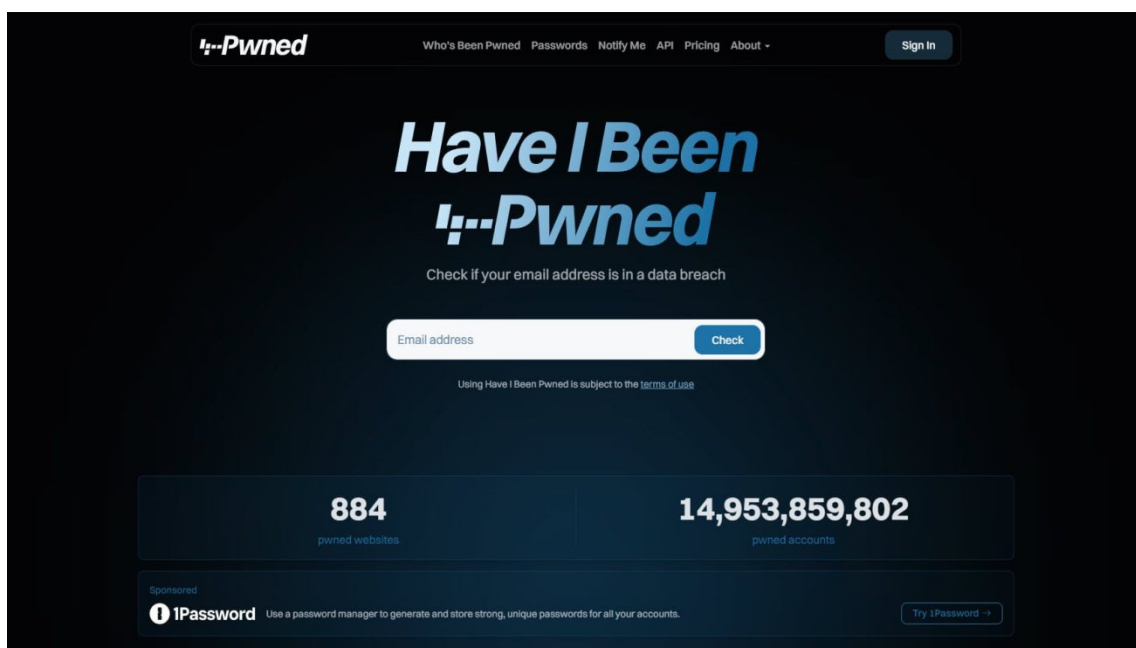


Рисунок 1.2 - Обзор аналога «Have I Been Pwned»

Третью группу составляют встроенные механизмы проверки паролей, реализованные в операционных системах, веб-браузерах и крупных интернет-сервисах. Такие механизмы автоматически предупреждают пользователя о недостаточной длине пароля или использовании распространённых комбинаций, а также могут предлагать альтернативные, более надёжные варианты. Несмотря на их удобство и высокий уровень интеграции, данные решения функционируют преимущественно как «чёрный ящик», то есть пользователь получает лишь итоговую оценку или предупреждение без детального объяснения причин, что снижает образовательную ценность и затрудняет понимание принципов формирования надёжных паролей.

Существуют также специализированные программные библиотеки для анализа, к примеру, библиотека `zxcvbn`. Подобные инструменты применяются разработчиками при создании систем аутентификации и позволяют учитывать словари, шаблоны и популярные комбинации символов, все это даёт возможность более точно оценить стойкость пароля и примерное время, необходимое для его подбора, но такие решения ориентированы на

разработчиков и не предназначены для использования обычными пользователями, они не обладают самостоятельным пользовательским интерфейсом и требуют интеграции в программные системы[9].

Рассмотрев все сильные и слабые стороны аналогов, мы можем подвести итог в виде краткого содержание всей информации об этих приложениях, что представлено в таблице 1.

Таблица 1 - Сравнение аналогичных решений

Тип решения	Преимущества аналога	Недостатки аналога
Онлайн-сервисы проверки паролей	Простота использования, доступность, отсутствие установки ПО	Формальный анализ, отсутствие глубокого анализа структуры пароля
Проверка по базам утечек	Высокая практическая значимость, выявление скомпрометированных паролей	Не оценивает новые пароли и их устойчивость к подбору
Встроенные проверки ОС и браузеров	Автоматизация, удобство использования	Отсутствие подробных пояснений и прозрачность анализа
Библиотеки оценки энтропии	Учет словарей и шаблонов, более точная оценка	Отсутствие пользовательского интерфейса, ориентированность на разработчиков

Проведённый анализ показал, что большинство существующих решений используют ограниченный набор методов оценки, таких как формальные правила сложности, проверка по базам утечек или расчёт энтропии. При этом редко применяется комплексный подход, который сочетает различные методы анализа и позволяет выявлять скрытую предсказуемость паролей. В целом все эти существующие аналоги не обеспечивают всестороннюю оценку стойкости пароля.

Это подтверждает целесообразность разработки веб-приложения, объединяющего различные методы анализа. Реализация такого подхода позволит повысить точность и информативность оценки стойкости паролей, а также улучшить понимание пользователями принципов безопасного формирования учётных данных.

1.4 Постановка задачи

Анализ предметной области и исследование существующих программных решений показали, что во многих современных средствах проверки паролей применяется ограниченный набор критериев оценки. Из-за этого часть факторов, которые влияют на реальную устойчивость к современным методам атак не учитываются. По этой причине пользователь получает неполное представление о степени защищённости своих учётных данных. Это подтверждает необходимость разработки программного средства, которое способно выполнять комплексный анализ с учётом различных аспектов[9].

Разрабатываемое веб-приложение должно показывать пользователю информацию о степени надежности вводимого пароля, обнаруженных уязвимостях и возможных способах их устранения. При этом результаты анализа важно представлять в понятной форме, не перегружая пользователя сложными терминами и пояснениями. Так как это напрямую влияет на восприятие и практическую пользу для пользователя. Дополнительно, приложение должно обеспечивать возможность генерации случайных паролей, которые соответствуют современным требованиям безопасности. Пользователь должен иметь возможность задать основные параметры, например, длина и используемые типы символов. Это позволит не только оценивать уже существующие пароли, но и создавать новые, которые будут соответствовать требованиям безопасности.

При разработке программного средства необходимо учитывать следующие требования. Система должна функционировать в виде веб-приложения, доступного через современные браузеры без установки дополнительного ПО.

					ПГУ 1.090301.13.001 ПЗ	Лист
						15
Изм.	Лист	№докум.	Подпись	Дата		

Это делает ее универсальной и более удобной для применения. Корректная работа должна обеспечиваться на распространённых операционных системах, включая Windows, Linux и macOS. Серверная часть должна поддерживать развёртывание в различных средах, включая как Linux-системы, например, Ubuntu или Debian и Windows Server. Отдельное внимание следует уделить производительности. Обработка пользовательских запросов должна происходить без заметных задержек, то есть максимально близко к режиму реального времени. Для стабильной работы системы достаточно стандартных серверных ресурсов, процессора с частотой около 2 ГГц, не менее двух ядер, оперативной памяти от 4 ГБ и порядка 10 ГБ свободного дискового пространства. Передача данных между клиентской и серверной частями должна осуществляться с использованием защищённых протоколов. При этом пароль не должен передаваться или храниться в открытом виде. Система должна корректно обрабатывать некорректные входные данные и обеспечивать стабильную работу при одновременном обращении нескольких пользователей.

Для описания взаимодействия пользователя с системой была разработана UML-диаграмма прецедентов использования, представленная на рисунке 1.3.

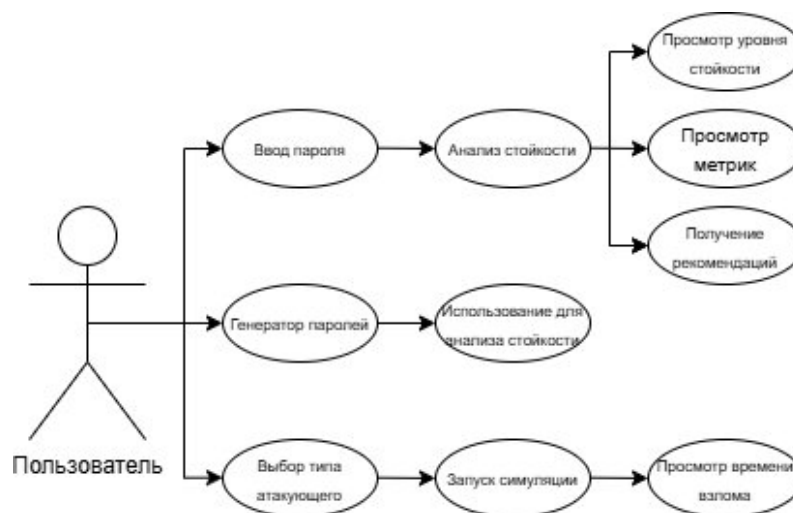


Рисунок 1.3 - Диаграмма прецедентов использования

К основным действиям пользователя относятся ввод пароля и получение результатов его анализа. В процессе обработки система выполняет оценку

стойкости пароля, вычисляет его характеристики, а также формирует рекомендации. Дополнительно в системе предусмотрена возможность моделирования различных сценариев атак. Пользователь имеет возможность выбрать тип атакующего и запустить процесс оценки времени подбора пароля, после чего система отображает результат в наглядной форме. Также предусмотрена функция генерации случайного пароля безопасного пароля.

1.5 Анализ решения поставленной задачи

Оценка стойкости пароля напрямую связана с определением того, насколько он устойчив к различным способам подбора. На практике пользователи часто создают простые или предсказуемые пароли. Это существенно повышает риск их взлома. Поэтому возникает необходимость в инструментах, которые позволяют оценить надёжность пароля непосредственно в момент его создания. Надёжность пароля нельзя определить по какому-то одному параметру. Она формируется за счёт совокупности характеристик. Увеличение длины и использование разных типов символов действительно усложняет подбор, но этого не всегда достаточно. Существенную роль играет то, как именно построен пароль. Пользователи нередко используют слова, связанные с личной информацией, или создают комбинации по привычным шаблонам, лишь немного их изменяя. В результате даже формально сложные пароли могут оставаться уязвимыми.

Существующие методы оценки стойкости паролей можно условно разделить на несколько групп. В одних решениях проверяется соответствие установленным требованиям, включая минимальную длину и наличие различных типов символов. Другие же основаны на вычислении энтропии. Также используются методы анализа структуры пароля, предназначенные для выявления повторяющихся элементов, последовательностей и распространённых шаблонов. Несмотря на существование различных методов, во многих программных решениях применяется лишь часть доступных способов. Из-за этого итоговая оценка может быть неточной или односторонней. В связи с этим

более целесообразно использовать комбинированный подход, при таком подходе проверка выполняется одновременно по нескольким направлениям. Такой способ анализа может учитывать не только количественные характеристики пароля, но и особенности его структуры. Помимо длины и мощности алфавита, необходимо определять наличие повторяющихся элементов, типовых шаблоны и других признаков. Для реализации данного подхода предлагается разработать веб-приложение, выполняющее многоуровневый анализ стойкости паролей. Система должна обеспечивать комплексный анализ вводимой комбинации с использованием различных методов.

В рамках разрабатываемого приложения анализ строится на следующих принципах:

- ~ вычисление энтропии как количественной оценки сложности перебора;
- ~ учёт длины и используемых классов символов;
- ~ выявление повторяющихся последовательностей и типовых шаблонов;
- ~ проверка совпадений с базами данных утекших паролей;
- ~ применение методов машинного обучения для выявления скрытых закономерностей в структуре пароля.

Таким образом, решение задачи заключается в разработке приложения, которое реализует комплексный анализ вводимой комбинации. Использование сочетания различных методов позволяет повысить точность результатов и приблизить их к условиям реальных атак.

2 Разработка

2.1 Описание архитектуры проекта

В рамках данной работы было принято решение реализовать веб-приложение с использованием клиент-серверной архитектуры. Такое решение обеспечивает удобство доступа к системе через браузер, независимость от используемой операционной системы, а также упрощает масштабирование и дальнейшее развитие проекта.

Клиент-серверная архитектура подразумевает разделение приложения на две основные части. Клиентскую часть и серверную часть. Клиентская часть отвечает за взаимодействие с пользователем и отображение результатов. А серверная реализует основную вычислительную логику. Подобное разделение позволяет изолировать обработку данных от пользовательского интерфейса, что положительно влияет на поддержание и структуру проекта. Доступ к системе осуществляется через веб-браузер. Серверная часть приложения реализована с использованием фреймворка Django, на архитектурном шаблоне MVT. В рамках данного подхода Model отвечает за работу с данными и их структуру, View обеспечивает обработку пользовательских запросов и формирование логики приложения и Template используется для формирования пользовательского интерфейса. Применение шаблона MVT обеспечивает разделение логики обработки данных, представления и пользовательского интерфейса, что позволяет повысить читаемость кода и упростить масштабирование приложения.

Архитектура разрабатываемого веб-приложения включает основные компоненты, такие как пользовательский интерфейс, серверную часть обработки

					ПГУ 1.090301.13.001 ПЗ	Лист
						19
Изм.	Лист	№докум.	Подпись	Дата		

запросов, модуль анализа стойкости паролей, модуль интеллектуального анализа на основе методов машинного обучения и вспомогательные модули для работы с базами утекших паролей. Общая архитектура проекта представлена на рисунке 2.1.

					ПГУ 1.090301.13.001 ПЗ	Лист
						20
Изм.	Лист	№докум.	Подпись	Дата		

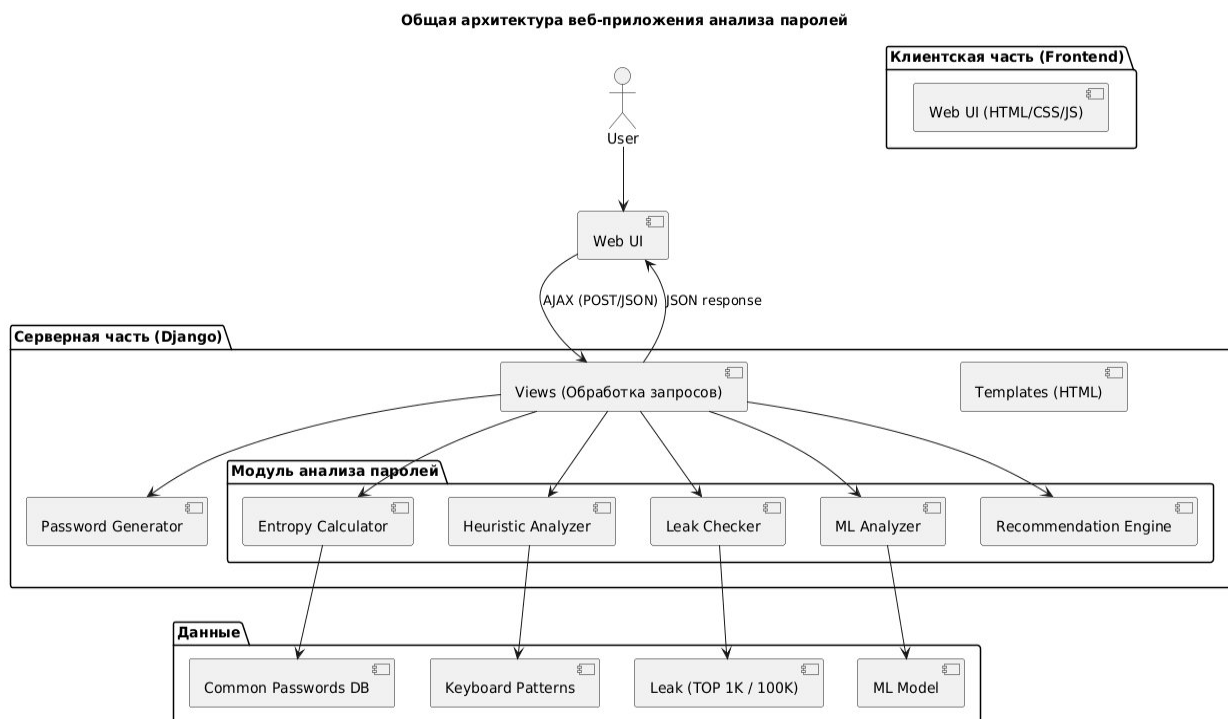


Рисунок 2.1 - Общая архитектура приложения

Пользовательский интерфейс предназначен для ввода комбинаций для анализа, выбора типа атаки и отображения результатов. Через интерфейс пользователь получает информацию о длине пароля, размере используемого набора символов, уровне стойкости, предполагаемом времени подбора, а также перечень факторов, которые снижают его безопасность. Дополнительно отображаются поясняющие комментарии и результаты интеллектуального анализа. Серверная часть реализует основную вычислительную логику приложения. После получения запроса от клиента сервер обрабатывает введённый пароль, сначала вычисляет его энтропию, определяет наличие повторяющихся элементов, последовательностей, типичных шаблонов, применяет системы штрафных корректировок, выполняет проверку по локальным базам утечек, также моделирует время подбора с учётом различных вычислительных возможностей атакующей стороны и производит дополнительный анализ с использованием методов машинного обучения. Все вычисления выполняются на стороне сервера, после чего результаты передаются клиенту в структурированном виде. Модуль анализа стойкости паролей

реализован в виде отдельного программного компонента и является ключевым элементом. Он включает алгоритмы расчёта энтропии, выявления предсказуемых структур и применения системы корректировок. В отличие от упрощённых моделей, в данном случае используется комбинированный подход, при котором оценка изменяется с помощью ограничивающих механизмов. Особое внимание уделяется проверке пароля на наличие в базах утечек, для этого используются локальные базы. Проверка выполняется без передачи исходного пароля в открытом виде. Отдельным компонентом является модуль интеллектуального анализа, реализующий методы машинного обучения. Дополнительно используется этап нормализации. Он позволяет ограничить итоговую энтропию при наличии характерных уязвимостей.

Выбранная архитектура обеспечивает удобство использования приложения, высокую производительность обработки запросов и возможность дальнейшего расширения функциональности, что полностью соответствует целям и задачам данной выпускной работы.

2.2 Выбор инструментов разработки

В соответствии с выбранной архитектурой приложения, особое значение приобретает обоснованный выбор инструментов разработки. От используемых технологий зависят такие характеристики системы, как производительность, безопасность, удобство использования и возможность дальнейшего масштабирования. Для реализации поставленных задач необходимо определить инструментарий приложения.

2.2.1 Выбор языка программирования и технологий разработки

Современные веб-приложения реализуются с использованием различных языков программирования и технологий, среди них наибольшее распространение получили JavaScript, Python, PHP и Java. Выбор конкретного набора инструментов должен учитывать особенности разрабатываемой системы и требования к её функциональности. Для реализации клиентской части веб-приложения используется язык JavaScript. Этот язык является стандартом для

создания интерактивных веб-интерфейсов. Его применение позволяет обеспечить динамическую обработку пользовательского ввода, визуализацию уровня стойкости пароля и оперативную обратную связь без перезагрузки страницы. Серверная часть приложения реализована на языке Python. Выбор данного языка обусловлен его высокой читаемостью, развитой экосистемой библиотек и широким применением в области анализа данных и информационной безопасности. Это делает его подходящим для реализации алгоритмов оценки стойкости паролей и интеграции машинного обучения. В качестве основного инструмента разработки серверной части выбран веб-фреймворк Django. Данный фреймворк реализует архитектурный шаблон MVT. Этот шаблон обеспечивает чёткое разделение компонентов системы и упрощает процесс разработки. Он предоставляет встроенные средства обработки HTTP-запросов, маршрутизации, работы с базами данных и реализации механизмов безопасности.

Для создания пользовательского интерфейса используются стандартные веб-технологии. HTML для формирования основной структуры страниц, CSS для их оформления и JavaScript для реализации интерактивных элементов на странице. Такой подход обеспечивает доступ к приложению из любого браузера и его доступность для широкого круга пользователей.

2.2.2 Выбор методов и алгоритмов оценки стойкости пароля

Важным этапом разрабатываемого веб-приложения является выбор методов, которые позволяют оценить стойкость пароля корректно. Существуют различные методы такой оценки, например, анализ длины, проверка состава символов, сравнение с распространёнными комбинациями, и вычисление информационной энтропии. Но применение каждого из этих методов по отдельности не даёт полной картины. Так, длинный пароль может оставаться уязвимым из-за предсказуемой структуры, а использование разных типов символов не гарантирует устойчивость к атакам, если пароль построен по шаблону. В рамках данной работы был выбран комбинированный подход. При таком решении оценка формируется на основе совокупности нескольких

взаимодополняющих методов. В первую очередь выполняется вычисление базовой энтропии, оно учитывает длину пароля и мощность используемого алфавита. Данный показатель позволяет оценить теоретическую сложность перебора, исходя из предположения о равномерном распределении символов. Но реальное поведение пользователей часто отклоняется от этой модели. Поэтому дополнительно применяется набор эвристических правил, которые направлены на выявление признаков предсказуемости. Они снижают практическую стойкость пароля. К таким признакам относятся повторяющиеся символы, последовательности чисел и символов клавиатуры, типовые шаблоны, например, слово с добавлением цифр или года, а также повторяющиеся фрагменты строки. Выявление подобных шаблонов позволяет скорректировать оценку с учётом реальных сценариев атак. Одним из этапов анализа является проверка пароля по локальным базам скомпрометированных данных. Если пароль уже встречался в известных утечках, он автоматически считается небезопасным. Важной особенностью реализованного алгоритма является способ применения штрафов. В отличие от простых моделей, где итоговая оценка уменьшается фиксированным значением, в данной системе используется более гибкий подход. Часть факторов напрямую снижает рассчитанную энтропию, тогда как другие влияют на итоговую оценку косвенно через механизм нормализации. Это позволяет избежать чрезмерного занижения результата и одновременно учитывать наличие уязвимых конструкций. Дополнительно в системе реализован модуль интеллектуального анализа. Он основан на методах машинного обучения, при этом модель применяется не ко всем входным данным, а только к тем паролям, которые уже прошли базовые проверки и обладают достаточной длиной и сложностью. Все перечисленные методы реализованы в виде независимых компонентов, что обеспечивает модульность системы и упрощает её дальнейшее развитие. При необходимости отдельные алгоритмы могут быть доработаны или заменены без изменения общей логики.

Таким образом, такой набор методов позволяет учитывать, как формальные характеристики пароля, так и особенности его структуры и распространённости.

Это обеспечивает более реалистичную оценку стойкости по сравнению с подходами, основанными только на одном критерии, и повышает практическую ценность получаемых результатов.

2.2.3 Среда разработки и программные инструменты

Для создания проекта был выбран PyCharm в качестве основной среды разработки. Этот инструмент очень удобен для написания кода, тестирования и поиска ошибок. Он также отлично работает с языком программирования Python и фреймворком Django. Один из главных плюсов Django это его встроенные функции безопасности. Например, он помогает защищает сайт от SQL-инъекций, межсайтового скриптинга и подделки запросов. Это особенно важно при работе с данными пользователей, включая их пароли. Также используется виртуальное окружения Python, чтобы изолировать зависимости проекта. Это помогает обеспечить стабильную работу приложения, независимо от того, как настроена операционная система и какие программы установлены. Для управления исходным кодом используем систему контроля версий Git. Она позволяет отслеживать изменения, сохранять промежуточные результаты и возвращаться к предыдущим версиям нашего проекта, если это необходимо.

Выбранные инструменты для разработки, включая Python, Django, JavaScript и PyCharm, идеально подходят для нашей системы. Они делают разработку проще, обеспечивают высокую безопасность и позволяют расширять функциональные возможности нашего веб-приложения в будущем.

2.3 Установка и настройка инструментов разработки

В соответствии с выбранными инструментами разработки, рассмотренными в разделе 2.2, перед началом реализации необходимо выполнить установку и настройку инструментов разработки.

На первом этапе выполняется установка среды разработки. Для этого используется версия PyCharm Community Edition, которая доступна бесплатно для загрузки с официального сайта разработчика. Страница загрузки среды представлена на рисунке 2.2. После загрузки установочного файла выполняется

установка с использованием стандартного мастера установки. На данном этапе рекомендуется активировать параметры интеграции с операционной системой, включая добавление среды разработки в контекстное меню и ассоциацию файлов с расширением .py.

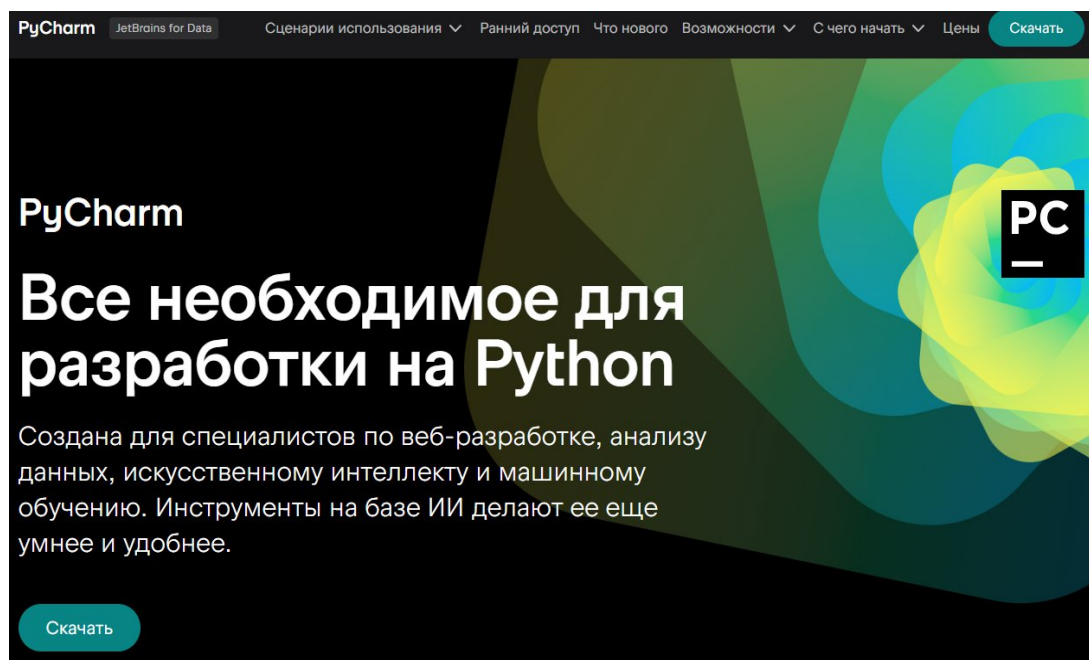


Рисунок 2.2 – Страница загрузки среды разработки PyCharm

После завершения установки необходимо убедиться, что установлен интерпретатор Python версии 3.x, который используется в качестве основного языка программирования проекта. Проверка установленной версии может быть выполнена средствами среды разработки или с использованием командной строки операционной системы. Для обеспечения изоляции зависимостей создаётся виртуальное окружение. Использование виртуального окружения позволяет локализовать используемые библиотеки в рамках проекта и исключить возможные конфликты с другими установленными пакетами.

На следующем этапе создаётся новый проект в среде разработки. Страница создания проекта показана на рисунке 2.3. В процессе создания задаются каталог размещения проекта, используемый интерпретатор и активируется опция автоматического создания виртуального окружения. В результате формируется

базовая структура проекта, которая необходима для дальнейшей разработки серверной части приложения.

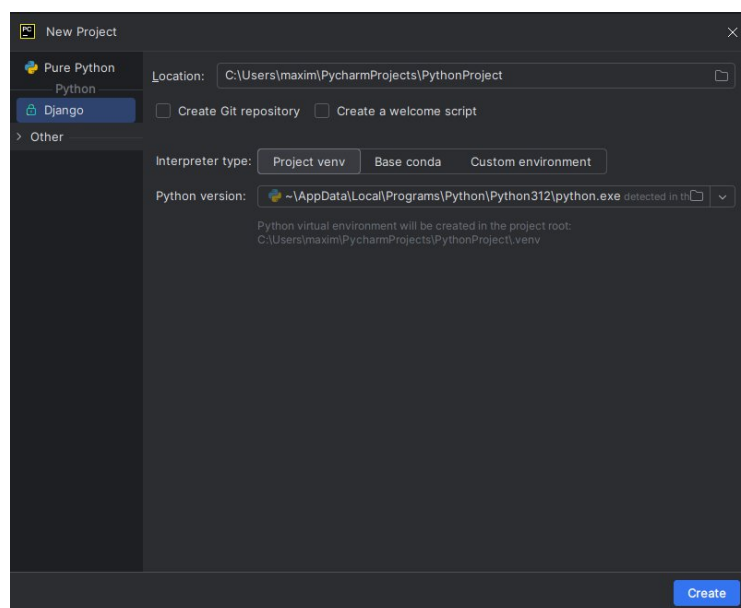


Рисунок 2.3 – Страница создания проекта

После создания проекта осуществляется установка веб-фреймворка Django с использованием пакетного менеджера `pip` внутри виртуального окружения. Установка выполняется с помощью соответствующей команды, после чего создаётся стандартная структура Django-проекта. Структура проекта показана на рисунке 2.4. Она включает конфигурационные файлы, маршрутизацию запросов и заготовки для приложений.

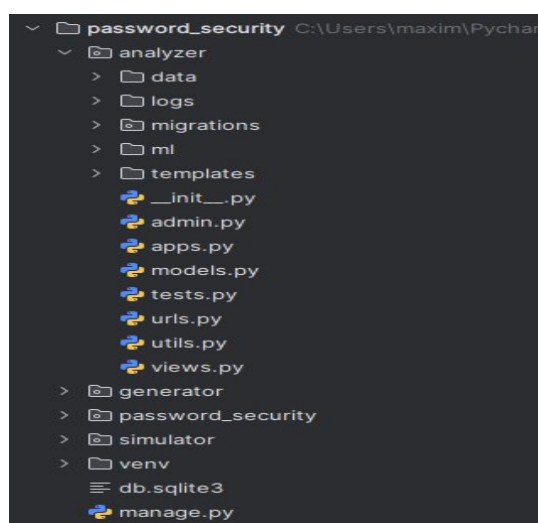


Рисунок 2.4 – Структура проекта

Дополнительно устанавливаются библиотеки, необходимые для реализации функциональности разрабатываемого приложения. Выбор используемых компонентов обусловлен архитектурой системы и задачами анализа стойкости паролей.

В процессе разработки были использованы следующие программные средства и библиотеки:

- ~ joblib, библиотека применяется для загрузки и использования предварительно обученной модели машинного обучения;

- ~ стандартные библиотеки Python (math, secrets, hashlib, re, json, logging), используемые при реализации алгоритмов анализа паролей, обработке данных и обеспечении безопасности.

Установка всех зависимостей и библиотек выполняется в рамках виртуального окружения. Это обеспечит независимость проекта от глобальных настроек системы и повысит стабильность его работы.

В результате описанных действий формируется настроенная среда разработки, которая позволяет приступить к реализации функциональных модулей веб-приложения.

2.4 Описание структуры проекта

Для более наглядного представления структуры приложения и взаимосвязей между его компонентами была построена UML-диаграмма компонентов представленная на рисунке 2.5. Она отражает структуру системы и взаимодействие ее основных частей.

Диаграмма компонентов веб-приложения анализа паролей

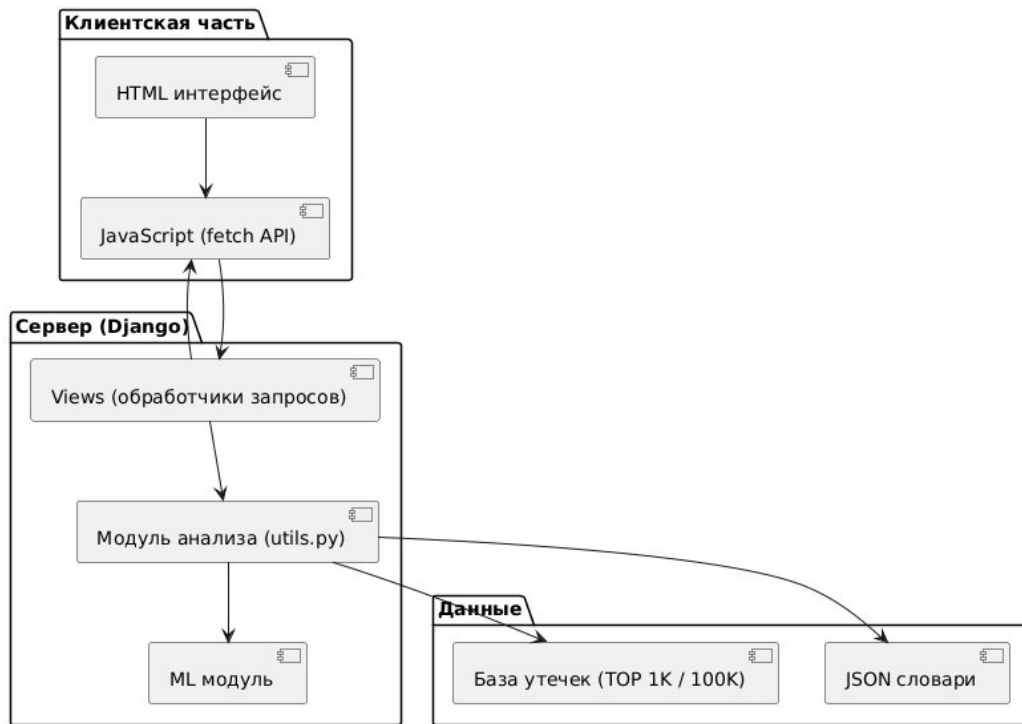


Рисунок 2.5 – UML-диаграмма компонентов

На ней показано, как между собой взаимодействуют ключевые компоненты приложения. Такая визуализация позволяет быстрее понять общую логику работы приложения и увидеть, какие модули за что отвечают. При этом основная вычислительная логика сосредоточена не в отдельных классах, а в наборе функций и вспомогательных модулей. Такой подход упрощает структуру проекта и делает её более гибкой при доработке.

Точкой входа в систему со стороны пользователя выступают обработчики HTTP-запросов. Именно они принимают данные, которые поступают с клиента, и передают их на дальнейшую обработку. После выполнения всех вычислений результат возвращается обратно. Обмен данными между клиентом и сервером осуществляется асинхронно. Благодаря этому страница не перезагружается при каждом запросе, а результаты анализа обновляются динамически. Такой подход делает интерфейс более отзывчивым и удобным для пользователя. Центральным элементом всей системы является модуль анализа паролей. Он реализован в

файле `utils.py`. В нём сосредоточены основная логика обработки введенных данных. Там выполняются следующие операции:

- ~ расчёт базовой энтропии пароля;
- ~ анализ длины и состава используемых символов;
- ~ выявление повторяющихся элементов и последовательностей;
- ~ обнаружение типовых шаблонов и масок;
- ~ проверка пароля по локальным базам утечек;
- ~ нормализацию итоговой оценки;
- ~ формирование рекомендаций для пользователя.

Обработка пароля происходит поэтапно. Сначала определяется базовая энтропия, после чего результат корректируется с учётом выявленных слабых мест. Например, при наличии повторяющихся символов или простых последовательностей итоговая оценка снижается. Если же пароль обнаруживается в базе скомпрометированных данных, его надёжность ограничивается. Дополнительно в системе предусмотрен модуль интеллектуального анализа. Он применяется для выявления паролей с формально высокой сложностью. Такой модуль подключается не всегда, а только при необходимости. Отдельным компонентом является генератор паролей. Он реализует создание случайных комбинаций с использованием криптографически стойкого генератора. При формировании пароля учитываются параметры, которые выбрал пользователь. Кроме того, из набора символов исключаются визуально похожие знаки, что снижает вероятность ошибок при вводе.

Помимо основных модулей, структура проекта включает вспомогательные элементы:

- ~ каталог `templates`, содержащий HTML-шаблоны интерфейса;
- ~ каталог `static`, включающий стили и клиентские скрипты;
- ~ директорию `ml`, в которой размещена обученная модель и функции извлечения признаков;
- ~ локальные базы данных паролей, используемые для проверки утечек.

Взаимодействие компонентов организовано следующим образом. В начале пользователь вводит пароль в интерфейсе, после чего данные передаются на сервер. Сервер выполняет анализ, при необходимости подключает модуль машинного обучения и возвращает результат пользователю. Полученные данные отображаются на странице.

Процесс взаимодействия компонентов системы при анализе пароля представлен на диаграмме последовательности, представленной на рисунке 2.6.

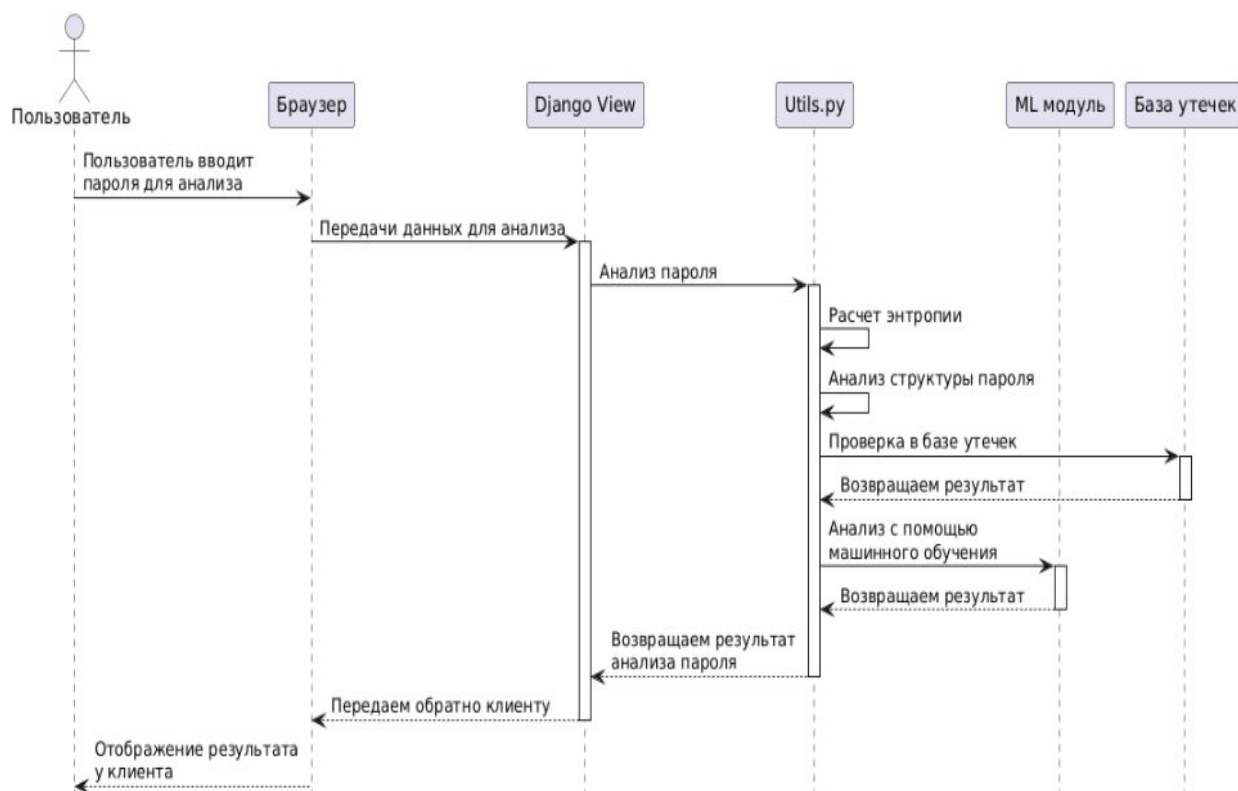


Рисунок 2.6 – Диаграмма последовательности анализа стойкости пароля

Последовательность выполнения основных этапов анализа пароля представлена на UML-диаграмме деятельности на рисунке 2.7.

Диаграмма деятельности анализа пароля

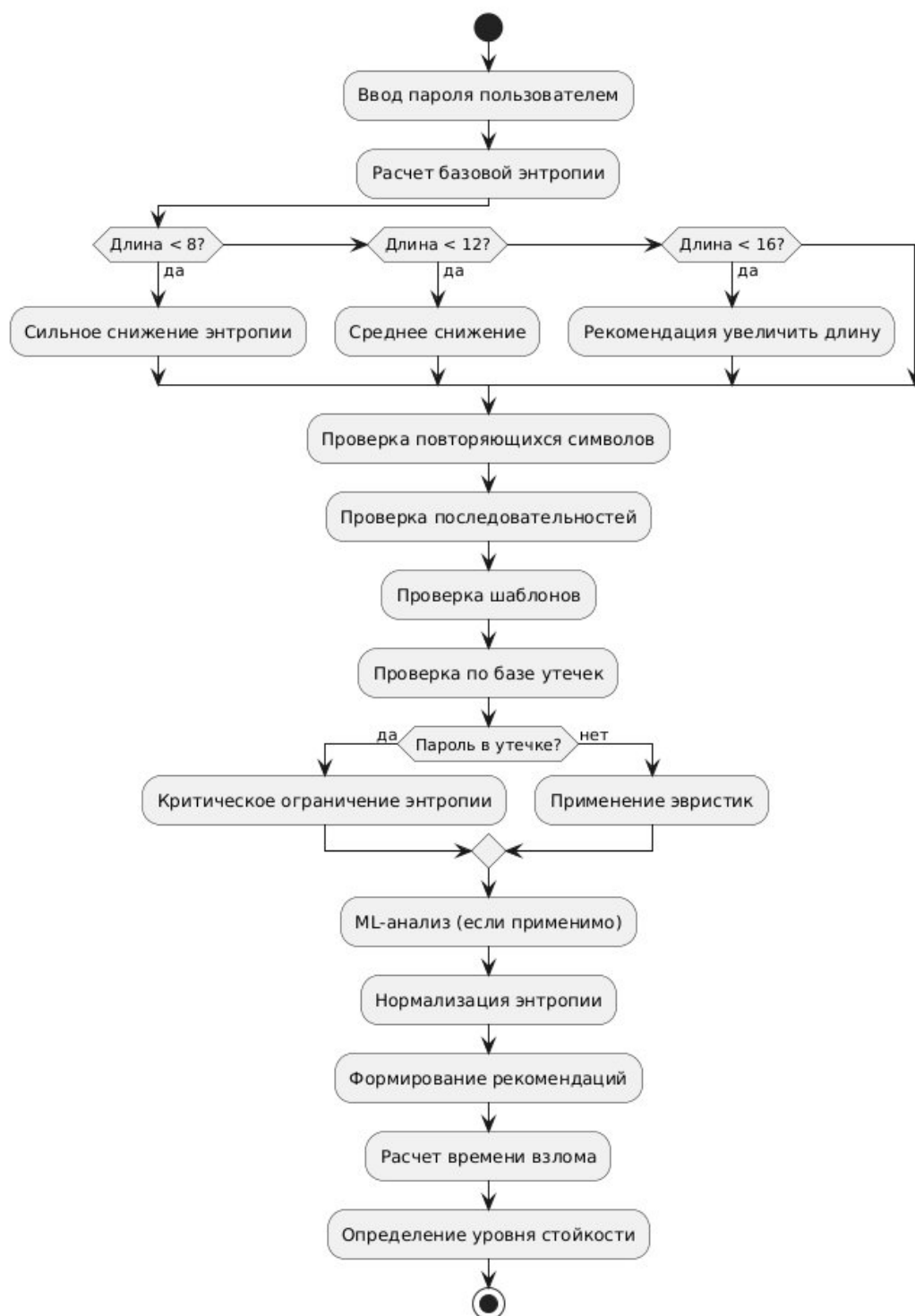


Рисунок 2.7 – Диаграмма деятельности веб-приложения для оценки стойкости пароля

Таким образом, структура проекта обеспечивает понятное разделение ответственности между компонентами, удобство масштабирования и высокую

отзывчивость системы. Реализованный подход позволяет эффективно обрабатывать пользовательские данные и формировать обоснованную оценку стойкости пароля.

2.5 Разработка программы

2.5.1 Проектирование структуры веб-приложения

Структура веб-приложения проектировалась на основе, описанной в разделе 2.4.

Проект включает следующие основные компоненты:

- ~ модуль представлений (views), отвечающий за обработку HTTP-запросов пользователя;
- ~ модуль вспомогательной логики (utils), реализующий алгоритмы анализа стойкости;
- ~ модуль машинного обучения (ml), включающий модель для дополнительной оценки структуры пароля;
- ~ шаблоны пользовательского интерфейса (templates);
- ~ статические ресурсы (static), обеспечивающие интерактивность интерфейса.

Обработка пользовательских запросов реализована в модуле views.py. Данный компонент принимает входные данные. Они поступают с клиентской части, инициирует процесс анализа и формирует ответ в формате JSON. При этом вычислительная логика вынесена в отдельные модули.

Основная логика анализа паролей сосредоточена в модуле utils.py. В нём реализованы функции, которые отвечают за вычисление энтропии, выявление предсказуемых структур, проверку по базам утечек и формирование итоговой оценки. Такой подход позволяет использовать данный модуль независимо от веб-интерфейса. Дополнительный уровень анализа обеспечивается модулем машинного обучения, который размещён в директории ml. Выделение данного компонента в отдельный модуль позволяет при необходимости обновлять модель без изменения основной логики приложения. Взаимодействие клиентской и

серверной частей организовано асинхронно с использованием JavaScript (fetch API). Запросы отправляются на сервер без перезагрузки страницы, а результаты динамически отображаются в интерфейсе. Это позволяет повысить удобство использования приложения и обеспечивает быструю реакцию системы на действия пользователя.

2.5.2 Реализация анализа стойкости пароля

Анализ стойкости пароля в разработанном приложении реализован как последовательный процесс. Он включает вычислительную оценку, проверку структуры и сопоставление с известными уязвимыми комбинациями. Основная логика анализа сосредоточена в функции `analyze_password()`. Она реализована в модуле `utils.py`. Такой подход позволяет учитывать формальные характеристики пароля и особенности его построения, которые зачастую снижают стойкость пароля пользователей.

Процесс обработки пользовательского ввода включает следующие шаги:

- ~ получение пароля от пользователя;
- ~ определение длины и состава используемых символов;
- ~ расчет базовой энтропии;
- ~ выявление повторяющихся элементов и предсказуемых последовательностей;
- ~ анализ типовых шаблонов;
- ~ проверка по локальным базам скомпрометированных паролей;
- ~ корректировка итоговой оценки;
- ~ расчёт ориентировочного времени подбора;
- ~ формирование итогового результата и рекомендаций.

На первом этапе вычисляется информационная энтропия. Она отражает сложность перебора пароля методом полного перебора. Используемая оценка основана на понятии информационной энтропии, введённом Клодом Шенноном в рамках теории информации. Для оценки применяется упрощенная модель, которая подразумевает равновероятное использование символов. В этом случае энтропия определяется как произведение длины пароля на логарифм мощности

используемого алфавита по основанию два. Значение N вычисляется с помощью функции `charset_size()`, которая определяет, какие группы символов присутствуют в пароле. Учитываются:

- ~ строчные латинские буквы (26 символов);
- ~ заглавные латинские буквы (26 символов);
- ~ цифры (10 символов);
- ~ специальные символы (32 символа).

Если пароль состоит из разных типов символов, мощность алфавита равна сумме значений для каждого типа. Например, если используются все категории, мощность алфавита равна 94. Энтропия зависит и от длины пароля, и от разнообразия используемых символов. Отметим, что эта модель предполагает, что символы распределены равномерно. Однако на практике это не всегда так. Пароли пользователей часто содержат повторяющиеся символы и предсказуемые элементы. Поэтому полученное значение энтропии нужно корректировать.

После вычисления основной оценки выполняется анализ структуры пароля. В рамках данного этапа выявляются признаки, которые характерны для слабых комбинаций. В частности, учитываются следующие факторы:

- ~ наличие повторяющихся символов;
- ~ последовательности символов (например, «1234» или «qwerty»);
- ~ повторяющиеся подстроки;
- ~ простые шаблоны (например, слово с добавлением цифр или года);
- ~ однородность используемых типов символов.

Для этого используются отдельные функции, такие как `detect_repetitions()`, `detect_sequences()` и `detect_simple_masks()`. Они помогают найти предсказуемые конструкции, которые упрощают подбор пароля. Например, если пароль состоит из повторяющихся блоков, таких как "abcabcabc" или "aA1!aA1!", он может быть длинным, но остаётся предсказуемым. В таких случаях мы снижаем оценку пароля, даже если он имеет высокую энтропию.

Далее идет проверка пароля по локальным базам утекших паролей. Она реализована функцией `check_local_leak()`. Поиск идет по локальной базе

распространённых паролей. Для этого используется предварительно сформированная база хешей, которая представлена в виде префиксной структуры. Проверка осуществляется по совпадению SHA-1 хеша. В случае обнаружения пароля в базе его оценка существенно снижается независимо от других характеристик, так как его использование уже представляет риск.

После применения всех правил выполняется нормализация результата с помощью функции `normalize_entropy()`. Это необходимо для предотвращения ситуаций, когда пароль получает завышенную оценку за счёт длины, несмотря на наличие очевидных шаблонов. В результате итоговая энтропия корректируется с учётом выявленных уязвимостей.

Для повышения наглядности результата выполняется оценка времени, необходимая для подбора. Расчёт выполняется функцией `crack_time_bruteforce()`, которая реализует модель полного перебора. Предполагается, что в среднем пароль будет найден после перебора половины всех возможных комбинаций. В системе предусмотрено несколько сценариев атак, задаваемых через словарь `ATTACK_SPEED`. Расчет выполняется для нескольких сценариев:

- ~ обычный персональный компьютер ($\sim 10^6$ – 10^8 попыток/с);
- ~ графические ускорители ($\sim 10^9$ – 10^{11});
- ~ распределённые системы ($\sim 10^{12}+$).

Полученное значение переводится в удобный для восприятия формат с помощью функции `format_time()`.

Итогом работы алгоритма является формирование комплексного результата, включающего:

- ~ категорию стойкости (низкая, средняя, высокая);
- ~ список выявленных уязвимостей;
- ~ рекомендации по улучшению пароля;
- ~ числовые показатели (энтропия, время подбора).

Таким образом, реализованный механизм анализа сочетает в себе формальные методы оценки и практические эвристики, что позволяет более точно отражать реальную стойкость пароля и выявлять его слабые стороны.

2.5.3 Реализация генератора паролей

Дополнительно был реализован модуль генерации паролей. Он предназначен для создания стойких комбинаций. Его использование помогает пользователю получить готовый безопасный пароль без необходимости самостоятельного создания. Основная логика генерации вынесена в отдельную функцию `generate_password()` модуля `utils.py`, что позволяет использовать данный механизм независимо от пользовательского интерфейса. Генерация выполняется на стороне сервера. Это обеспечивает контроль над алгоритмом и исключает влияние клиента на результат. Для формирования используется криптографический стойкий генератор случайных чисел и строк `secrets`, который предоставляется стандартной библиотекой языка Python. В отличие от других генераторов случайных чисел он устойчив к подбору и ориентирован на задачи информационной безопасности.

Пользователь может задать основные параметры будущего пароля:

- ~ длину;
- ~ использование заглавных букв;
- ~ использование цифр;
- ~ использование специальных символов.

На основе выбранных параметров внутри функции `generate_password()` формируется допустимый набор символов (`charset`). При этом организовано важное ограничение. Из алфавита исключаются визуально похожие символы. Такое ограничение введено для повышения удобства практического использования паролей и снижения вероятности ошибок при вводе.

Процесс генерации выполняется поэтапно. На первом этапе определяется конечный набор символов в зависимости от выбранных параметров. Далее формируется список используемых символов, по одному из каждой категории. Это делается для того, чтобы итоговый пароль действительно содержал все выбранные типы. После рассчитывается оставшееся длина пароля. Все недостающие символы заполняются случайным образом из общего набора параметров с помощью `secrets.choice()`. На завершающем этапе сформированная

последовательность символов перемешивается, с помощью `secrets.SystemRandom().shuffle()`, чтобы исключить предсказуемое расположение обязательных элементов в начале строки. Если пользователь не выбрал ни одной категории символов, то автоматически используется базовый набор (буквы и цифры), что предотвращает генерацию пустого или некорректного значения. Взаимодействие с генератором реализовано асинхронно. При нажатии кнопки на клиентской стороне формируется POST-запрос, который идет на сервер. Обработчик во `views.py` вызывает функцию `generate_password()`, после чего сформированный пароль возвращается клиенту в формате JSON.

Таким образом, реализованный генератор сочетает в себе криптографическую надежность, гибкость настройки и удобство использования. За счет исключения неоднозначных символов и гарантированного разнообразия состава формируемые пароли обладают высокой стойкостью и пригодны для реального использования пользователями в реальных системах.

2.5.4 Реализация модуля машинного обучения

В дополнение к классическим методам анализа в системе реализован дополнительный модуль машинного обучения. Он предназначен для повышения точности оценки. В отличие от основной логики анализа, которая основана на фиксированных правилах и эвристиках, данный модуль позволяет учитывать скрытые зависимости между характеристиками пароля. Это особенно важно в случаях, когда стандартные проверки не выявляют явных уязвимостей. Модуль реализован как отдельный компонент и подключается к приложению через заранее обученную модель. Она предварительно сохранена в файл с использованием библиотеки `joblib`. В случае возникновения ошибки загрузки. Например, при отсутствии файла модели используется резервный сценарий переменная `ml_model` принимает значение `None`, и дальнейший анализ продолжается без применения машинного обучения. В качестве алгоритма была выбрана логистическая регрессия, реализованная в библиотеке `scikit-learn`. Данный алгоритм хорошо подходит для задач бинарной классификации. Он

отличается высокой скоростью работы и позволяет получать вероятностную оценку результата, что удобно для последующей интерпретации. Обучение модели выполнялось отдельно от основного приложения. Для обучения модели была сформирована выборка, которая включала различные типы паролей. В нее вошли:

- ~ случайно сгенерированные сложные комбинации;
- ~ пароли, построенные на основе словарных слов;
- ~ модификации популярных шаблонов.

Перед обучением каждый пароль преобразовывался в набор числовых признаков. Извлечение признаков реализовано отдельной функцией `extract_features`. Функция формирует вектор характеристик, эти характеристики отражают, как количественные параметры, так и структуру пароля. В частности, учитываются:

- ~ длина пароля;
- ~ энтропия;
- ~ доля цифр, заглавных букв и специальных символов;
- ~ наличие повторов;
- ~ наличие последовательностей;
- ~ наличие типовых шаблонов.

Этот подход позволяет модели учитывать не только насколько сложный пароль, но и насколько предсказуемой конструкцией он обладает. В процессе работы программы она вызывает функцию `analyze_password()`, там происходит вызов модели. После извлечения признаков они передаются в модель. После анализа пароля, результаты передаются в модель, которая определяет надежен ли пароль. Если результат ниже определённого порога, пароль считается потенциально уязвимым, даже если он достаточно сложный. Нужно отметить, что эта модель не используется для всех паролей. В коде проверяются базовые условия, такие как минимальная длина и уровень сложности. Для коротких или явно слабых паролей модель не используется, анализ выполняется только обычными методами. Это помогает снизить нагрузку на компьютер и избежать

лишних операций. Результат работы модели не изменяет напрямую вычисленное значение энтропии. Но влияет на итоговую интерпретацию. В случае выявления подозрительной структуры пользователю дополнительно выводится предупреждение о возможной предсказуемости пароля.

Интеграция машинного обучения в общий процесс анализа позволяет:

- ~ выявлять сложные закономерности, которые трудно описать в виде правил;
- ~ повысить достоверность итоговой оценки;
- ~ снизить риск ложного ощущения безопасности;
- ~ сделать систему более гибкой к различным типам паролей.

Таким образом, использование логистической регрессии в составе приложения дополняет традиционные методы анализа и позволяет более точно оценивать реальную стойкость паролей с учётом их структуры, а не только формальных характеристик.

2.5.5 Реализация пользовательского интерфейса

Пользовательский интерфейс веб-приложения ориентирован на широкий круг пользователей. Его основной задачей является предоставление понятного и наглядного взаимодействия с системой анализа стойкости паролей и удобство представления результата.

Интерфейс реализован с использованием шаблонного механизма фреймворка Django. HTML-шаблоны размещены в каталоге templates и отвечают за структуру страниц. Обработка данных выполняется на стороне сервера. Такой подход упрощает сопровождение проекта и дает возможность изменять внешний вид приложения независимо от серверной части.

Основная страница приложения включает следующие элементы:

- ~ поле ввода пароля;
- ~ переключатель отображения пароля;
- ~ блок с индикаторами состава (наличие цифр, букв, символов);
- ~ область отображения итоговой оценки;
- ~ блок с отчётом и рекомендациями;

~ инструмент генерации пароля и выбора сценария атаки.

Пользователь вводит пароль в соответствующее поле, после чего запускается процесс анализа. Вводимые символы скрыты, но при нажатии на кнопку, можно увидеть, что вы ввели в поле для проверки. На клиентской стороне реализован обработчик события `input`. Он реагирует на изменение содержимого поля ввода. Для предотвращения избыточных обращений к серверу используется механизм задержки отправки запроса `debounce`. Он реализован с помощью функций `setTimeout` и `clearTimeout`. Запрос отправляется с небольшой задержкой. Около 120 мс, и при продолжении ввода предыдущий вызов отменяется. Это позволяет снизить нагрузку на сервер и избежать лишних вычислений. Дополнительно в системе реализован механизм отмены предыдущих запросов с использованием `AbortController`. При отправке нового запроса предыдущий принудительно прерывается, что исключает обработку устаревших данных и повышает актуальность отображаемого результата. Взаимодействие клиентской и серверной частей осуществляется с использованием асинхронных HTTP-запросов на основе JavaScript API `fetch`. Для передачи данных используется метод `POST`. Он позволяет избежать передачи пароля в адресной строке и повышает уровень безопасности. Запрос формируется в формате `JSON` и содержит введенный пароль, а также выбранный пользователем сценарий атаки. На стороне сервера обработка выполняется во `views.py`. Входные данные извлекаются из тела запроса и передаются в функцию. В рамках этой функции выполняется полный цикл анализа. Результат обработки возвращается клиентской части в формате `JSON`. Полученные данные используются для динамического обновления интерфейса. При этом изменяются только отдельные элементы страницы, которые отвечают за отображение результата анализа, что позволяет избежать полной перезагрузки страницы и делает работу приложения более быстрой и удобной.

Результаты анализа отображаются в удобной для пользователя форме. Пользователь получает:

~ текстовую оценку уровня стойкости;

					ПГУ 1.090301.13.001 ПЗ	Лист
						41
Изм.	Лист	№докум.	Подпись	Дата		

- ~ цветовую индикацию (шкалу надёжности);
- ~ числовые характеристики (длина, мощность алфавита, энтропия);
- ~ оценку времени подбора;
- ~ список выявленных уязвимостей;
- ~ рекомендации по улучшению пароля.

Дополнительно предусмотрено отображение результатов интеллектуального анализа. В случае, если модель машинного обучения выявляет признаки предсказуемости, пользователю выводится предупреждение о потенциальной уязвимости.

К преимуществам разработанного интерфейса можно отнести простоту использования, наглядность предоставления результатов, высокую отзывчивость и возможность дальнейшего расширения.

Таким образом, реализованный пользовательский интерфейс обеспечивает удобное и понятное взаимодействие с системой, а также способствует формированию более осознанного подхода к выбору паролей.

2.6 Описание доступных функций

Разрабатываемое приложение предназначено для оценки стойкости пользовательских паролей и может использоваться, как в повседневной практике, так и в учебных целях. Все основные функции доступны через браузер. Поэтому пользователю не требуется установка дополнительных программ. Функциональность системы построена таким образом, чтобы пользователь мог последовательно пройти весь процесс. От ввода пароля до получения подробного объяснения результата. При этом особое внимание уделено не только итоговой оценке, но и тому, чтобы пользователь понимал, за счёт каких факторов она формируется. Главная страница приложения показано на рисунке 2.8

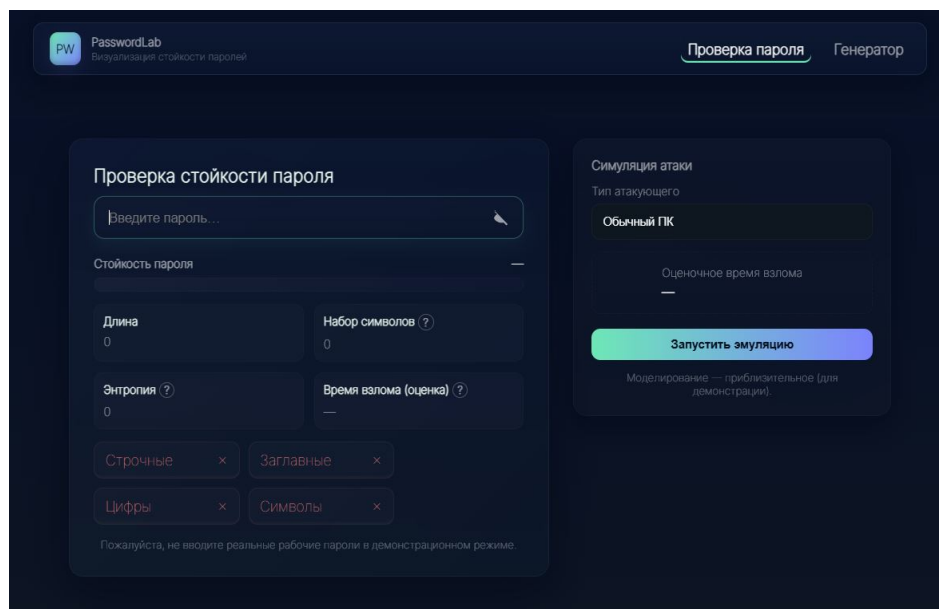


Рисунок 2.8 – Главная страница

2.6.1 Анализ стойкости пароля

Основной функцией приложения является анализ введенного пароля.

После ввода данных система запускает обработку, в ходе которой последовательно оцениваются различные характеристики. Результат формируется не сразу в виде одной оценки, а как совокупность показателей, на основе которых уже делается итоговый вывод. Это позволяет пользователю увидеть, какие именно аспекты делают пароль слабым или, наоборот, достаточно надёжным. Пример результатов анализа стойкости пароля представлен на рисунке 2.9.

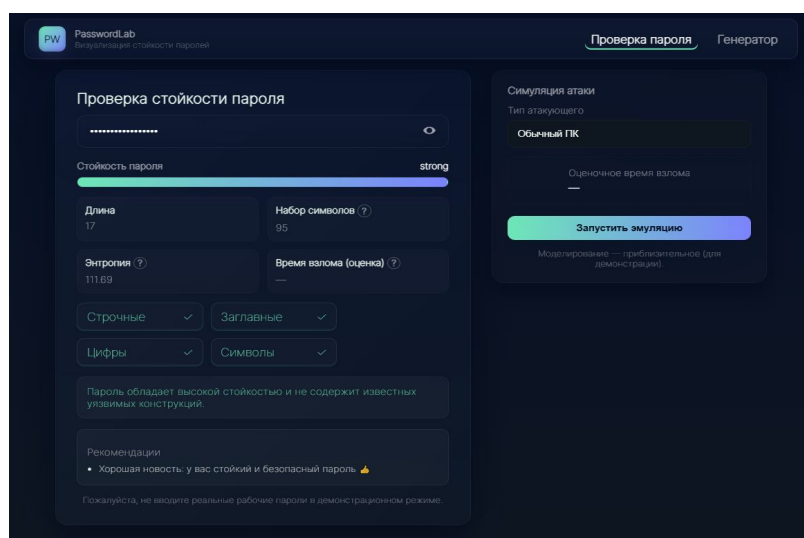


Рисунок 2.9 – Результаты анализа

2.6.2 Проверка пароля на утечки данных

Отдельное внимание уделено проверке пароля по известным базам утечек.

Такая проверка позволяет определить, использовался ли данный пароль ранее в скомпрометированных системах. При реализации учитывается вопрос безопасности. Пароль не передаётся в открытом виде. Если совпадение обнаружено, система сразу сообщает об этом пользователю. На практике это один из наиболее критичных факторов, поскольку даже сложный пароль не может считаться безопасным, если он уже известен злоумышленникам. Пример обнаружения пароля в утечках представлен на рисунке 2.10.

The screenshot shows the PasswordLab web application interface. The main section is titled "Проверка стойкости пароля" (Password Strength Check). The password "admin2004" is entered, and its strength is evaluated as "very_weak". A red progress bar indicates the low strength. The interface provides detailed statistics: Length (9), Symbol Set (36), Entropy (16), and Estimated Time to Crack (Practically impossible). It also shows which character types are included (Lowercase, Numbers) and excluded (Uppercase, Symbols). A warning message states: "Пароль обладает низкой стойкостью и может быть взломан при словарных или комбинированных атаках." (The password has low strength and can be cracked with dictionary or hybrid attacks). A "Дополнительный анализ:" (Additional analysis) section highlights that the password contains predictable patterns. Under "Причины снижения оценки:" (Reasons for score reduction), it lists: "Слишком короткий пароль" (Password is too short), "Пароль найден в утечках — использовать нельзя" (Password found in leaks — cannot be used), and "Критически распространённый пароль (TOP-1000)" (Critically common password (TOP-1000)). Recommendations include: "Пароль найден в самых распространённых — использовать нельзя" (Password found in the most common — cannot be used). A disclaimer at the bottom states: "Пожалуйста, не вводите реальные рабочие пароли в демонстрационном режиме." (Please do not enter real working passwords in demo mode).

Рисунок 2.10 – Результаты обнаружения пароля в базе утечек

2.6.3 Прогноз времени подбора пароля

Для того чтобы сделать результат более понятным, в приложении реализован расчёт предполагаемого времени подбора пароля. Оценка выполняется для нескольких сценариев. От обычных вычислительных возможностей до более мощных систем. Полученное значение переводится в удобный формат, например, секунды, часы или годы, чтобы пользователь мог наглядно представить, насколько устойчив его пароль. Такой подход позволяет уйти от абстрактных показателей и показать результат в форме, близкой к реальным условиям. Пример оценки времени подбора пароля представлен на рисунке 2.11.

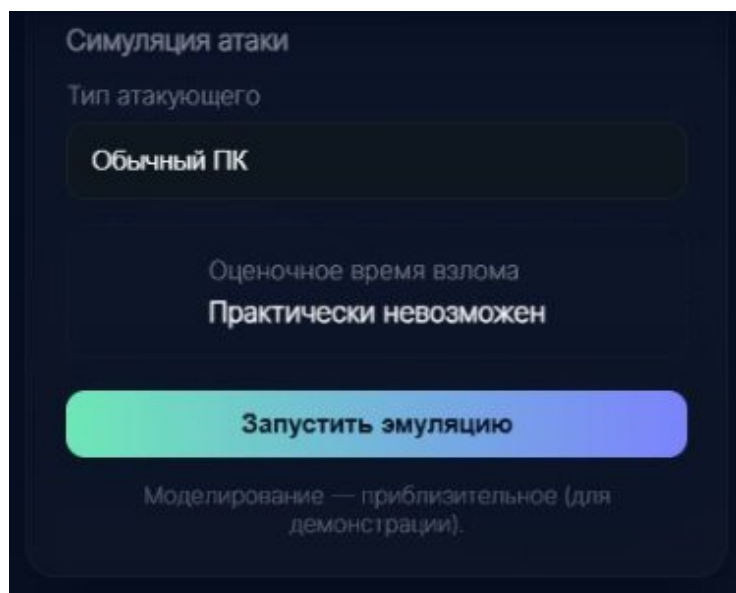


Рисунок 2.11 – Результат предполагаемого времени подбора пароля

2.6.4 Интеллектуальный анализ с использованием машинного обучения

Помимо базовых проверок в системе используется дополнительный уровень анализа, который основан на модели машинного обучения. Его задача выявлять ситуации, когда пароля выглядит сложным, но на самом деле является предсказуемым. К примеру, это могут быть комбинации, в которых используются слова с незначительными модификациями. Этот результат не заменяет основной анализ, а дополняет его, позволяя скорректировать итоговую

оценку. Таким образом, система учитывает не только формальные параметры, но и более скрытые закономерности. Результат интеллектуального анализа показан на рисунке 2.12.

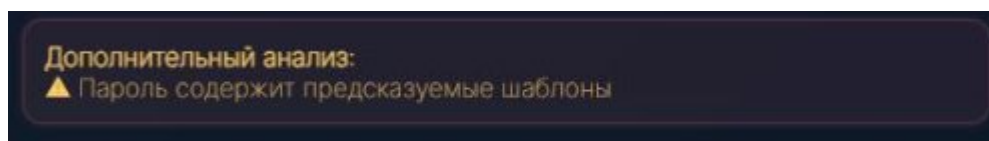


Рисунок 2.12 – Результат интеллектуального анализа

2.6.5 Генератор надёжных паролей

В приложении также предусмотрен инструмент генерации паролей. Он предназначен для пользователей, которым трудно самостоятельно придумать достаточно надежную комбинацию. Пользователь может самостоятельно задать основные параметры. После этого система формирует случайный пароль. Генерация основана на использовании устойчивого механизма случайного выбора, что обеспечивает непредсказуемость результата. Полученный пароль можно использовать напрямую или взять за основу. Интерфейс генератора паролей представлен на рисунке 2.13.

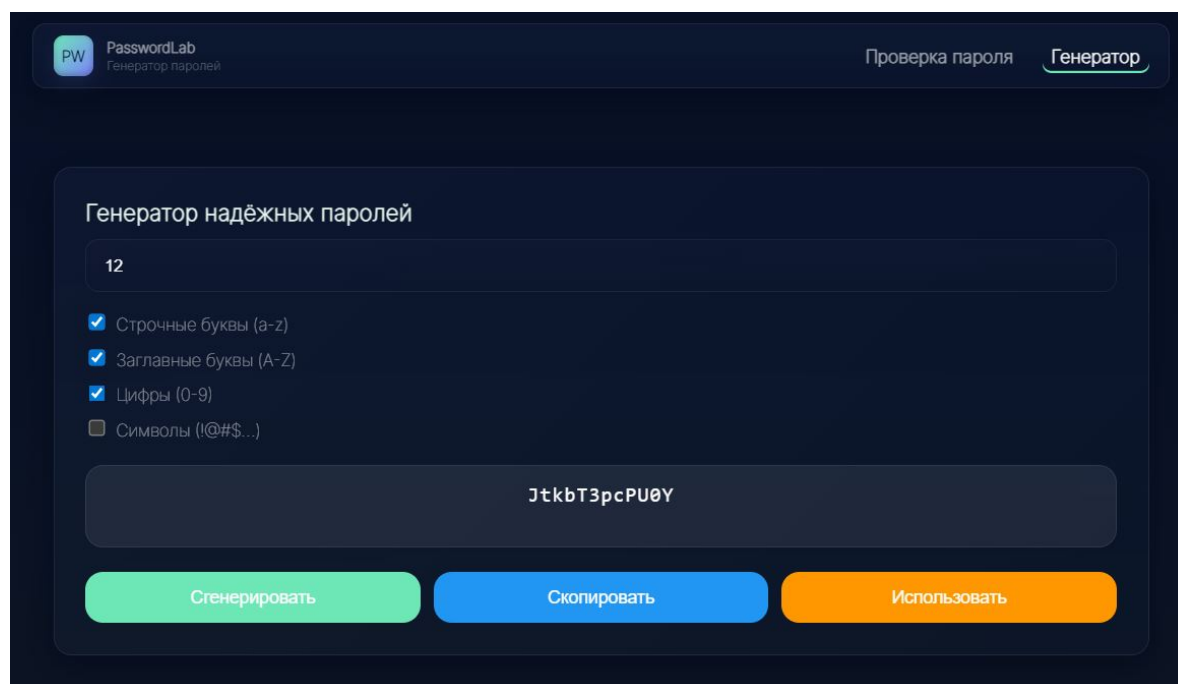


Рисунок 2.13 – Интерфейс генератора паролей

2.6.6 Формирование итогового отчёта

После завершения анализа формируется итоговый отчёт. В нём объединяются все полученные результаты уровень стойкости, числовые показатели, выявленные проблемы и рекомендации. Отчёт обновляется каждый раз при вводе нового пароля, поэтому пользователь может сразу увидеть, как изменения влияют на результат. Это делает процесс интерактивным и более понятным. Фактически отчёт выполняет две функции. С одной стороны, он даёт оценку текущего уровня безопасности. С другой помогает разобраться, как улучшить пароль и избежать типичных ошибок.

В совокупности реализованные функции позволяют оценивать пароль с разных сторон, а не только по формальным признакам. Пользователь получает не просто итоговый результат, а развернутое объяснение, что делает приложение полезным не только как инструмент проверки, но и как средство обучения.

3 Тестирование разработанного приложения

В процессе разработки была проведена серия тестов. Они были направлены на проверку работы алгоритма оценки стойкости пароля и пользовательского интерфейса. Оно выполнялось методом функциональной проверки, при котором анализировались входные данные и сравнивались ожидаемые результаты и фактические результаты работы системы.

Каждый тестовый сценарий включает:

- ~ входные данные (проверяемый пароль);
- ~ ожидаемый результат;
- ~ фактический результат работы системы;
- ~ иллюстрацию результата (скриншот).

В качестве первого теста был использован пароль password123. Он широко известен как один из самых распространенных и часто встречающихся в утечках.

Ожидаемый результат:

- ~ минимальный уровень стойкости;
- ~ сообщение о наличии в базе утечек;
- ~ игнорирование длины и состава символов при формировании итоговой оценки.

Система успешно обнаружила пароль в локальной базе утечек и принудительно ограничила итоговую оценку. Результат тестирования показан на рисунке 3.1. Пользователю было выведено соответствующее предупреждение. Полученный результат полностью совпал с ожидаемым.

Проверка стойкости пароля

Стойкость пароля

very_weak

Длина

11

Набор символов ?

36

Энтропия ?

16

Время взлома (оценка) ?

—

Строчные

✓

Заглавные

×

Цифры

✓

Символы

×

Пароль обладает низкой стойкостью и может быть взломан при словарных или комбинированных атаках.

Дополнительный анализ:

▲ Пароль содержит предсказуемые шаблоны

Причины снижения оценки:

▲ Слишком короткий пароль

▲ Пароль найден в утечках — использовать нельзя

▲ Критически распространённый пароль (TOP-1000)

Рекомендации

• Пароль найден в самых распространённых — использовать нельзя

Рисунок 3.1 – Результат тестирования пароля из утечек

Во втором тесте был рассмотрен пароль с повторяющейся структурой.

Ожидаемый результат:

- ~ оценка не должна быть максимальной;
- ~ наличие предупреждений о повторяющемся шаблоне;
- ~ ограничение итоговой оценки несмотря на длину.

Система корректно выявила повторяющийся шаблон и применила соответствующие штрафы. Результат тестирования показан на рисунке 3.2. Кроме того, итоговый уровень стойкости был ограничен, что исключило ситуацию ложного завышения оценки за счёт длины. Результат соответствует ожидаемому.

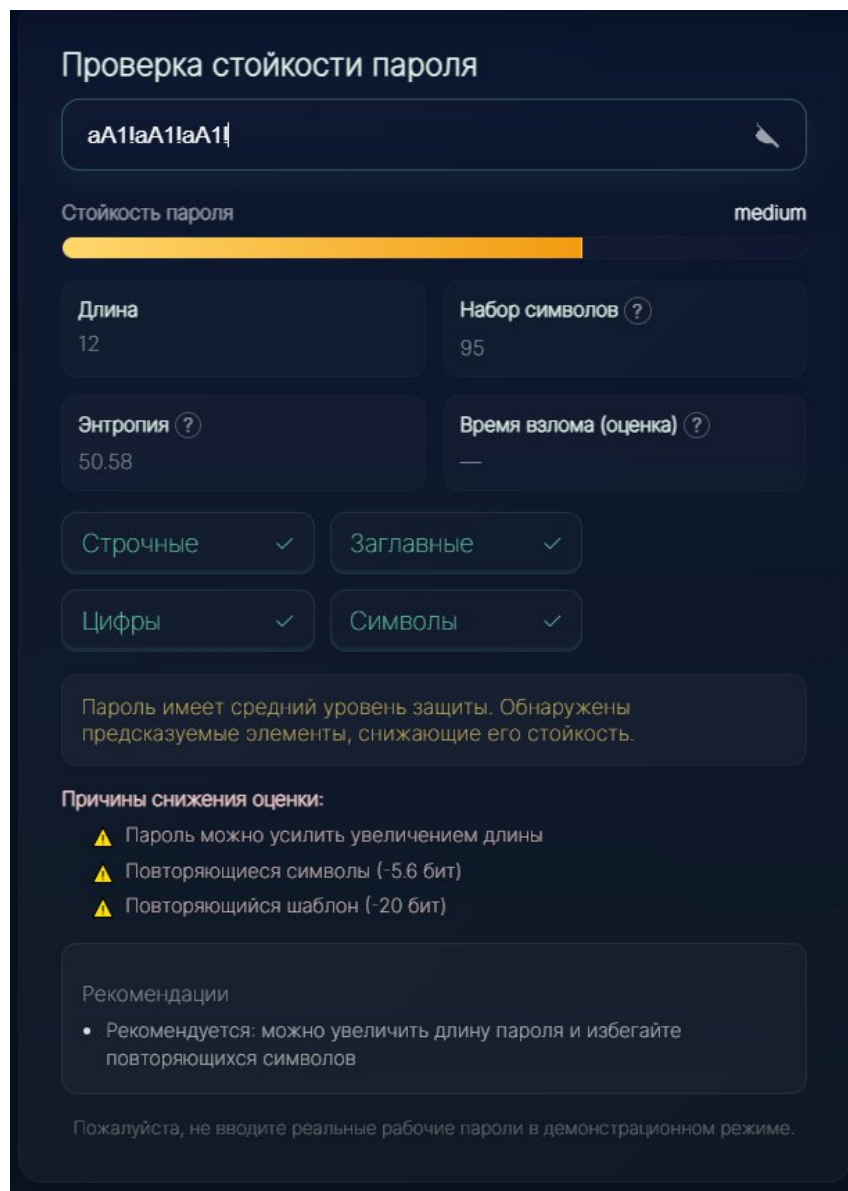


Рисунок 3.2 – Результат тестирования шаблонного пароля

В качестве следующего тестового сценария был выбран короткий пароль.

Ожидаемый результат:

- ~ низкая стойкость;
- ~ предупреждение о недостаточной длине;
- ~ рекомендации по увеличению длины и усложнению пароля.

Система корректно классифицировала пароль как небезопасный, отобразила низкий уровень стойкости и сформировала рекомендации по его улучшению. Результат полностью совпал с ожидаемым. Результат тестирования представлен на рисунке 3.3.

Проверка стойкости пароля

аВ1|

Стойкость пароля very_weak

Длина: 3

Набор символов: 62

Энтропия: 10.72

Время взлома (оценка): —

Строчные ✓ Заглавные ✓

Цифры ✓ Символы ✗

Пароль обладает низкой стойкостью и может быть взломан при словарных или комбинированных атаках.

Причины снижения оценки:

- ⚠ Слишком короткий пароль

Рекомендации

- Рекомендуется: добавить спецсимволы и увеличить длину пароля

Пожалуйста, не вводите реальные рабочие пароли в демонстрационном режиме.

Рисунок 3.3 – Результат тестирования короткого пароля

Заключительный тест был направлен на проверку корректной обработки сложного пароля, содержащего различные типы символов и не имеющего очевидных шаблонов. Результат тестирования показан на рисунке 3.4.

Ожидаемый результат:

- ~ высокий уровень стойкости;
- ~ высокая энтропия;
- ~ отсутствие критических предупреждений.

Система определила пароль как надёжный, не выявила существенных уязвимостей и не сформировала критических замечаний. Полученные результаты соответствуют ожидаемым.

Проверка стойкости пароля

G7#kLp9IQa2\$Xz

Стойкость пароля strong

Длина
14

Набор символов ?
95

Энтропия ?
91.98

Время взлома (оценка) ?
—

Строчные ✓

Заглавные ✓

Цифры ✓

Символы ✓

Пароль устойчив к перебору, однако содержит предсказуемые шаблоны, ограничивающие его стойкость.

Причины снижения оценки:

⚠ Пароль можно усилить увеличением длины

Рекомендации

- Хорошая новость: у вас стойкий и безопасный пароль 👍

Пожалуйста, не вводите реальные рабочие пароли в демонстрационном режиме.

Рисунок 3.4 – Результат тестирования надежного пароля

4 Анализ результатов

4.1 Результаты разработки приложения

В ходе выполнения выпускной квалификационной работы было разработано веб-приложение. Оно предназначено для оценки стойкости пользовательских паролей. При проектировании особое внимание уделялось структуре приложения. Оно организовано таким образом, чтобы отдельные компоненты можно было изменять и дополнять без переработки всей системы.

Работа приложения довольно проста. Пользователь вводит пароль через веб-интерфейс, после чего система запускает его обработку. Анализ выполняется поэтапно и включает несколько проверок, каждая из которых отвечает за отдельный аспект. Сначала определяется длина пароля и какие символы в нем используются, затем рассчитывается его энтропия. Далее применяются эвристические правила. Эти правила позволяют выявить повторяющиеся части, простые последовательности и характерные шаблоны. Параллельно выполняется проверка по базам утечек, которая реализована с учётом требований безопасности. На следующем этапе происходит анализ с помощью машинного обучения. Этот модуль помогает найти случаи, когда пароль кажется сложным, но на самом деле не является таковым. Последний этап оценивает время, которое понадобится для подбора пароля при различных сценариях. Результаты показываются в интерфейсе в удобной форме. Пользователь может увидеть не только итоговую оценку, но и дополнительные пояснения, какие проблемы обнаружены и что можно улучшить. Для наглядности используются цветовые индикаторы, числовые показатели и текстовые рекомендации.

Сделанное приложение производит анализ пароля с помощью комбинированного подхода. Это позволяет получить более точную и приближённую к реальности оценку стойкости.

Таким образом, поставленные задачи были выполнены, а разработанное приложение может использоваться как в учебных целях, так и в качестве инструмента для повышения уровня безопасности пользователей.

4.2 Оценка эффективности и ограничения системы

Эффективность работы приложения проводилась на основе тестирования, которое было описано в предыдущем разделе. Во время проверки были использованы различные варианты паролей, они отличались различными параметрами. Это позволило проверить правильность работы основных функций системы и оценить качество анализа стойкости паролей. Результаты тестирования показали, что приложение правильно определяет слабые пароли, при этом пользователю отображались предупреждения и рекомендации по улучшению пароля. Это позволило не только показать уровень безопасности, но и объяснить причины снижения оценки. Также была протестирована работа алгоритмов анализа структуры пароля. В процессе тестирования система успешно выявляла повторяющиеся шаблоны и усложнённые комбинации, например, пароли с повторением одинаковых символов не получали максимальную оценку даже при большой длине и наличии разных типов символов. Это показывает, что реализованные правила не завышают результатов анализа. Тестирование было проведено и для локальной базы утечек, при использовании украденных паролей система корректно определяла наличие совпадения и снижала оценку. Во время тестирования также проверили как работает модуль машинного обучения. Модель успешно находила пароли, которые имеют предсказуемую структуру. В подобных случаях система выдавала дополнительное предупреждение о возможной уязвимости пароля. Это позволило сделать итоговую оценку более приближённой к реальным условиям использования. Тестирование механизма оценки времени подбора пароля, соответствовала ожидаемым значениям. Несмотря на то, что данная модель является приближённой, она позволяет наглядно показать различие между слабыми и стойкими паролями. В процессе тестирования серьёзных ошибок, которые могли бы повлиять на работу системы, обнаружено не было. Основные функции приложения работали корректно. Результаты анализа отображались стабильно. При этом разработанная система имеет некоторые ограничения. В

					ПГУ 1.090301.13.001 ПЗ	Лист
						54
Изм.	Лист	№докум.	Подпись	Дата		

первую очередь это связано с тем, что модель машинного обучения зависит от обучающей выборки, поэтому она не может учитывать абсолютно все варианты построения пользовательских паролей. Расчёт времени подбора основан на модели полного перебора и не учитывает применение словарных атак, утечек данных и специализированных методов взлома. Стоит учитывать, что приложение ориентировано в основном на пользовательский уровень и не предназначено для анализа стойкости паролей в условиях специализированных целевых атак. Следовательно, в результате тестирования стало понятно, что разработанное приложение успешно справляется с поставленной задачей. Совокупность используемых методов позволила повысить точность оценки стойкости паролей и сделать результаты анализа более реалистичными и понятными для пользователя.

4.3 Рекомендации к дальнейшей разработке

Веб-приложение успешно выполняет поставленную задачу и может использоваться в качестве инструмента для оценки безопасности пользовательских комбинаций. В процессе разработки и тестирования были выявлены направления, в которых систему можно улучшить. Одним из основных направлений дальнейшего развития стало совершенствование модуля машинного обучения. В текущей версии используется модель логистической регрессии. Эта модель обучена на подготовленной выборке паролей различной сложности. Несмотря на то, что модель показывает хорошие результаты при выявлении предсказуемых шаблонов, качество анализа во многом зависит от данных на которых производилось обучение. В дальнейшем можно расширить набор используемых паролей. Добавить более разнообразные комбинации. Это позволит повысить точность классификации и сделать результаты анализа более точными. Дополнительно можно расширить набор анализируемых признаков.

Также перспективным направлением является развитие генератора паролей.

В дальнейшем можно добавить:

- ~ генерацию парольных фраз;
- ~ возможность сохранения пользовательских настроек;
- ~ настройку уровня сложности;

Ещё одним направлением для улучшения может стать механизма проверки утечек. В текущей реализации используется локальная база распространённых паролей. В дальнейшем можно реализовать автоматическое обновление базы или подключение нескольких источников данных. Это позволит повысить актуальность проверки и улучшить качество обнаружения скомпрометированных паролей.

Таким образом, разработанное приложение обладает хорошей основой для дальнейшего развития. Улучшение алгоритмов анализа, расширение функциональности и развитие пользовательского интерфейса позволят сделать систему более точной, гибкой и удобной для практического использования.

5 Безопасность жизнедеятельности

5.1 Правила безопасности при использовании ПК

Правила выполнения работ с ПК обязаны обеспечивать минимизацию основных рисков, перечисленных в ГОСТ 12.0.003-2015. В первую очередь к ним относятся:

- ~ высокий уровень напряженности электрического и магнитного полей в зоне установки компьютера;
- ~ повышенная температура отдельных элементов персонального компьютера;
- ~ монотонность трудового процесса;
- ~ высокий уровень зрительных нагрузок;
- ~ вероятность поражения статическим электричеством, генерируемым техникой;
- ~ недостаточная освещенность в зоне выполнения операций у компьютера;
- ~ другие факторы.

Полностью избавиться от их действия, конечно, невозможно, однако пользователь приложения обязан стремиться к тому, чтобы сократить воздействие этих факторов на себя, обеспечивая возможность восстановления нормального самочувствия, работоспособности и обучаемости, не подвергая себя всевозможным рискам.

5.2 Режим работы

Непрерывная работа за компьютером вызывает зрительное напряжение. При восьмичасовом рабочем дне необходимы перерывы. Рекомендуемая продолжительность перерыва составляет 10-15 минут через каждые два часа работы.

					ПГУ 1.090301.13.001 ПЗ	Лист
						57
Изм.	Лист	№докум.	Подпись	Дата		

Во время перерыва рекомендуется сменить положение тела, выполнить движения кистями и шеей. Допускается гимнастика для глаз.

5.3 Электробезопасность и противопожарные меры

Подключение оборудования должно выполняться через исправные розетки с заземлением. Использование кабелей с поврежденной изоляцией запрещается.

При обнаружении перегрева оборудования, запаха гари или нестабильной работы устройства эксплуатация должна быть прекращена. Оборудование отключается от сети.

Помещение должно быть оснащено средствами пожаротушения. Проходы к выходам должны оставаться свободными.

Системные блоки и вентиляционные отверстия должны очищаться от пыли.

					ПГУ 1.090301.13.001 ПЗ	Лист
						58
Изм.	Лист	№докум.	Подпись	Дата		

Заключение

В ходе выполнения выпускной работы была рассмотрена задача оценки стойкости пользовательских паролей. В процессе работы были рассмотрены основные методы анализа паролей, подходы к определению их надёжности, а также современные технологии, которые применяются при разработке веб-приложений. На основе проведённого анализа было разработано веб-приложение для проверки стойкости паролей. Приложение реализовано на языке Python с использованием фреймворка Django и работает по клиент-серверному принципу. Особое внимание в ходе разработки было уделено тому, чтобы система не только выполняла анализ пароля, но и предоставляла пользователю понятный результат с пояснениями и рекомендациями. В приложении были реализованы функции вычисления энтропии пароля, анализа структуры, проверки повторяющихся последовательностей и поиска паролей по локальной базе утечек. Был разработан генератор паролей, который позволяет создавать случайные комбинации с заданными параметрами. В систему был встроен модуль машинного обучения. Его использование позволило выявлять пароли, которые формально выглядят достаточно сложными, но при этом содержат предсказуемые шаблоны. Благодаря этому итоговая оценка стала более приближённой к реальным условиям использования паролей. В ходе тестирования было установлено, что приложение корректно обрабатывает различные типы паролей. Также было проверено взаимодействие между основными компонентами системы и работа пользовательского интерфейса.

Таким образом, поставленная цель работы была достигнута и успешно выполнены основные задачи. Разработанное приложение может использоваться как в учебных целях, так и в качестве вспомогательного инструмента для

повышения осведомлённости пользователей в вопросах информационной безопасности.

Список использованных источников

1. Официальная документация Python [Электронный ресурс]. – URL: <https://docs.python.org/3/> (дата обращения: 07.04.2026).
2. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы и исходные тексты на языке С. – М.: Вильямс, 2019. – 784 с.
3. DjangoSoftwareFoundation. Django Documentation [Электронный ресурс]. – URL: <https://docs.djangoproject.com/> (дата обращения: 08.04.2026).
4. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow. – 3rd ed. – Sebastopol: O'Reilly Media, 2022.
5. HuntT. HaveIBeenPwned: PwnedPasswordsAPI [Электронныйресурс]. – URL: <https://haveibeenpwned.com/API/v3#PwnedPasswords> (датаобращения: 16.04.2026).
6. Joblib Documentation [Электронный ресурс]. –URL: <https://joblib.readthedocs.io/> (датаобращения: 14.04.2026).
7. MozillaDeveloperNetwork. FetchAPI [Электронныйресурс]. – URL: https://developer.mozilla.org/enUS/docs/Web/API/Fetch_API (дата обращения: 15.04.2026).
8. NISTDigitalIdentityGuidelines [Электронныйресурс]. – URL: <https://pages.nist.gov/800-63-3/> (датаобращения: 05.04.2026).
9. OpenAI. ChatGPT: largelanguagemodel [Электронныйресурс]. – URL: <https://chat.openai.com> (датаобращения: 05.04.2026).
10. OWASPFoundation. AuthenticationCheatSheet [Электронныйресурс]. – URL: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html (датаобращения: 05.04.2026).

11. Scikit-learn Developers. Scikit-learn Documentation [Электронный ресурс]. – URL: <https://scikit-learn.org/stable/> (дата обращения: 11.04.2026).
12. Stallings W. Cryptography and Network Security: Principles and Practice. – 7th ed. – Pearson, 2018.

Приложение А

(обязательное)

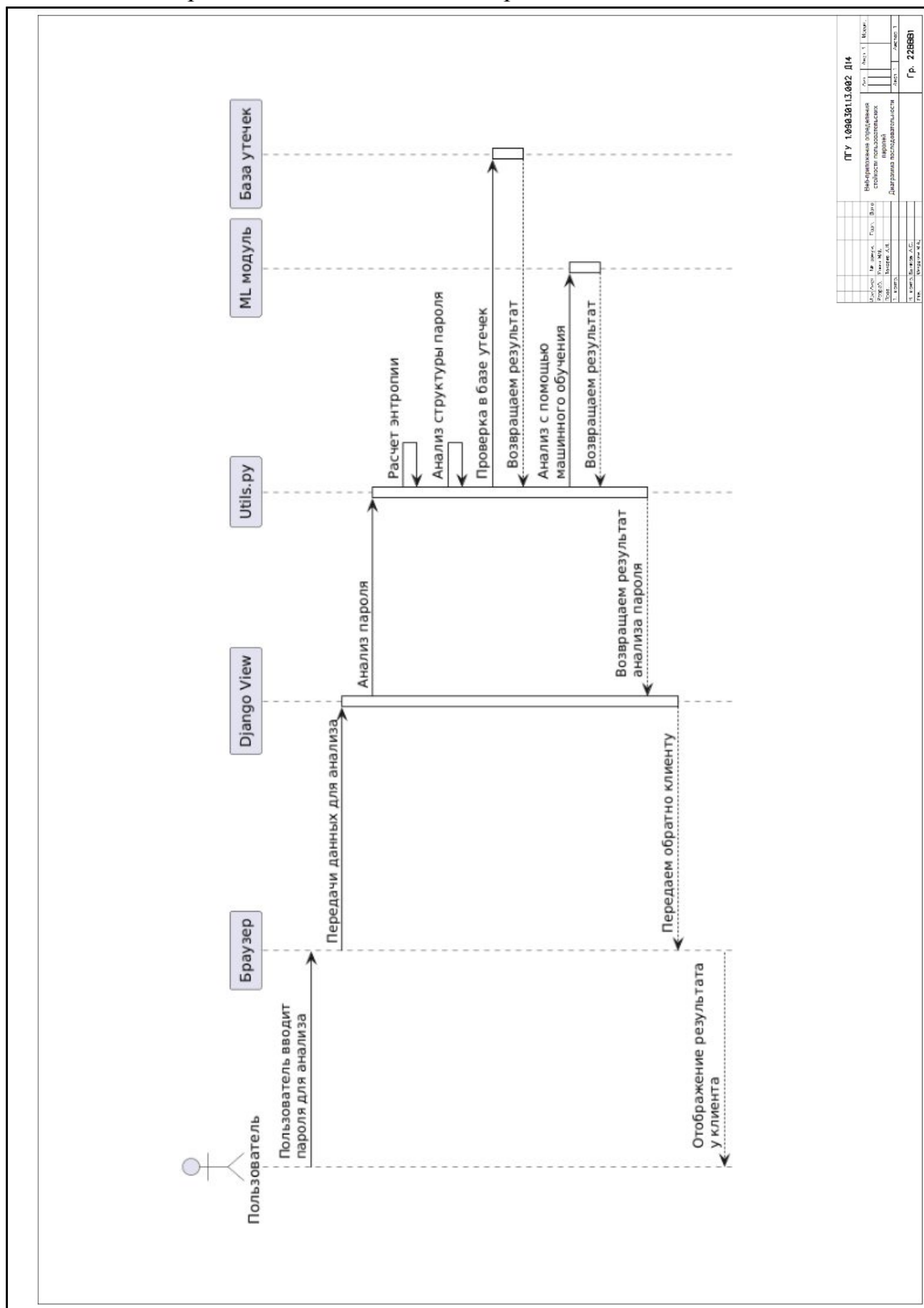
Веб-приложение определения стойкости пользовательских паролей

UML диаграммы

					ПГУ 1.090301.13.001 ПЗ	Лист
						61
Изм.	Лист	№докум.	Подпись	Дата		

[illegible][illegible]

Приложение А.2 – UML-диаграмма последовательности



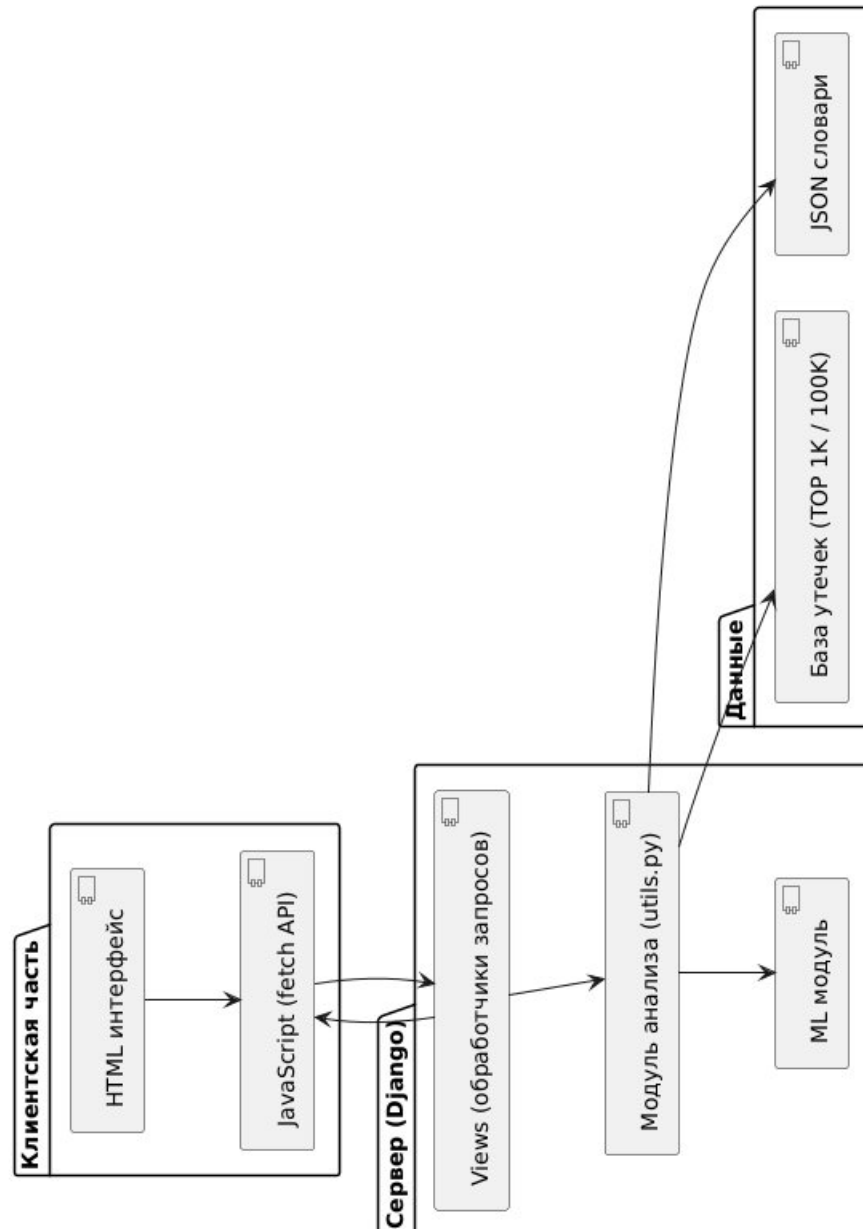
ПГУ 1.090301.13.002 Д14									
Инициальная определяющая статусы последовательности									
Диаграмма последовательности									
Имя	№ докум.	Вариант	Дата	Исполн.	Провер.	Утверд.	Визир.	Визир.	Визир.
Имя	№ докум.	Вариант	Дата	Исполн.	Провер.	Утверд.	Визир.	Визир.	Визир.
Имя	№ докум.	Вариант	Дата	Исполн.	Провер.	Утверд.	Визир.	Визир.	Визир.
Гр. 228881									

Изм.	Лист	№ докум.	Подпись	Дата

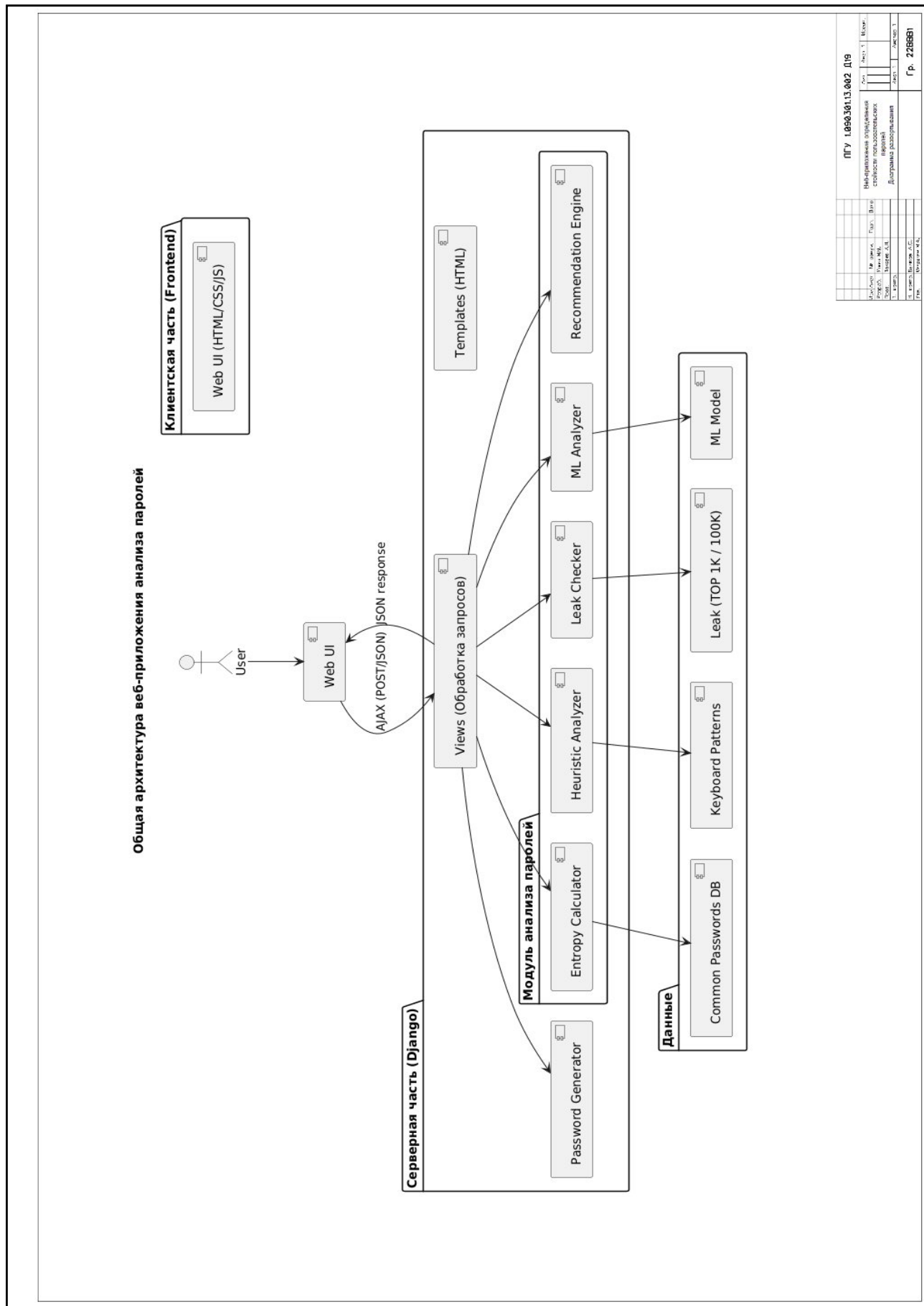
ПГУ 1.090301.13.001 ПЗ

Лист

63



ПТУ 09030113.002 Д18		Найр-павианова отрядинава столости столостолостолости павианов		Сол	Дави	Маси
		Диагностика отрядинава		Дави	Дави	Дави
				г.р. 2286881		



Приложение Б

(обязательное)

Веб-приложение определения стойкости пользовательских паролей

Листингпрограммы

					ПГУ 1.090301.13.001 ПЗ	Лист
						66
Изм.	Лист	№докум.	Подпись	Дата		

Файл Manage.py

```
"""Django's command-line utility for administrative tasks."""
import os
import sys
def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'password_security.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and
"
            "available on your PYTHONPATH environment variable? Did
you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)
if __name__ == '__main__':
    main()
```

Файл«Utils.py»

```
import math
import secrets
import re
import json
import string
import logging
import hashlib
import requests
import os
import joblib
from .ml.features import extract_features
from pathlib import Path
# Настройка логирования
LOG_DIR = Path(__file__).parent / "logs"
LOG_DIR.mkdir(exist_ok=True)
logging.basicConfig(
    filename=LOG_DIR / "password_analyzer.log",
    level=logging.INFO,
    format="%(asctime)s | %(levelname)s | %(message)s",
    encoding="utf-8"
)
# Пути к JSON-данным
BASE_DIR = Path(__file__).resolve().parent
DATA_DIR = BASE_DIR / "data"
COMMON_PASSWORDS_FILE = DATA_DIR / "common_passwords.json"
KEYBOARD_SEQUENCES_FILE = DATA_DIR / "keyboard_sequences.json"
TOP_1K_FILE = DATA_DIR / "top_1k_shal.txt"
TOP_100K_FILE = DATA_DIR / "top_100k_shal.txt"
```

					ПГУ 1.090301.13.001 ПЗ	Лист
						67
Изм.	Лист	№докум.	Подпись	Дата		

```

TOP_1K_PENALTY = 40
TOP_100K_PENALTY = 25
def load_prefix_db(path):
    db = {}
    try:
        with open(path, "r", encoding="utf-8") as f:
            for line in f:
                line = line.strip()
                if not line or ":" not in line:
                    continue
                prefix, suffix = line.split(":", 1)
                if prefix not in db:
                    db[prefix] = set()
                db[prefix].add(suffix)
    except Exception as e:
        logging.error(f"Ошибка загрузки базы {path}: {e}")
    return db
LEAKED_TOP_1K = load_prefix_db(TOP_1K_FILE)
LEAKED_TOP_100K = load_prefix_db(TOP_100K_FILE)
# Загрузка словарей
def load_common_passwords():
    try:
        with open(COMMON_PASSWORDS_FILE, "r", encoding="utf-8") as f:
            data = json.load(f)
            return set(p.lower() for p in data.get("passwords", []))
    except Exception:
        return set()
def load_keyboard_sequences():
    try:
        with open(KEYBOARD_SEQUENCES_FILE, "r", encoding="utf-8") as f:
            data = json.load(f)
            return data.get("sequences", [])
    except Exception:
        return []
COMMON_PASSWORDS = load_common_passwords()
KEYBOARD_SEQUENCES = load_keyboard_sequences()
def check_local_leak(password):
    if not password:
        return None
    sha1 = hashlib.sha1(password.encode("utf-8")).hexdigest().upper()
    prefix = sha1[:5]
    suffix = sha1[5:]
    # TOP 1K (самый критичный)
    if prefix in LEAKED_TOP_1K:
        if suffix in LEAKED_TOP_1K[prefix]:
            return "top1k"
    # TOP 100K
    if prefix in LEAKED_TOP_100K:
        if suffix in LEAKED_TOP_100K[prefix]:
            return "top100k"

```

```

        return None
    AMBIGUOUS = "00I1l"
    def remove_ambiguous(s):
        return "".join(c for c in s if c not in AMBIGUOUS)
    # Генератор паролей
    def generate_secure_password(length=12, use_lower=True,
    use_upper=True, use_digits=True, use_symbols=True):
        lower = remove_ambiguous(string.ascii_lowercase)
        upper = remove_ambiguous(string.ascii_uppercase)
        digits = remove_ambiguous(string.digits)
        symbols = "!@#$%^&*()-_+=[]{}<>?/"
        charset = ""
        required_chars = []
        if use_lower:
            charset += lower
            required_chars.append(secrets.choice(lower))
        if use_upper:
            charset += upper
            required_chars.append(secrets.choice(upper))
        if use_digits:
            charset += digits
            required_chars.append(secrets.choice(digits))
        if use_symbols:
            charset += symbols
            required_chars.append(secrets.choice(symbols))
        if not charset:
            charset = lower + upper + digits
            required_chars.extend([
                secrets.choice(lower),
                secrets.choice(upper),
                secrets.choice(digits)
            ])
        remaining_length = max(0, length - len(required_chars))
        password = required_chars + [secrets.choice(charset) for _ in
range(remaining_length)]
        secrets.SystemRandom().shuffle(password)
        result = "".join(password)
        logging.info(
            f"Password generated | length={length} |
charset_size={len(charset)}"
        )
        return result
    # Charset
    def charset_size(password):
        has_lower = any(c.islower() for c in password)
        has_upper = any(c.isupper() for c in password)
        has_digit = any(c.isdigit() for c in password)
        has_special = any(not c.isalnum() for c in password)
        size = 0
        if has_lower: size += 26
        if has_upper: size += 26
        if has_digit: size += 10
        if has_special: size += 33

```

```

        return size
# Штрафы
def detect_sequences(password):
    lower_pw = password.lower()
    # === 1. Клавиатурные последовательности ===
    for item in KEYBOARD_SEQUENCES:
        pattern = item.get("pattern", "") if isinstance(item, dict)
    else item
        penalty = item.get("penalty", 15) if isinstance(item, dict)
    else 15
        if pattern and pattern in lower_pw:
            return penalty # сильный штраф
    # === 2. Числовые последовательности ===
    if re.search(r"0123|1234|2345|3456|4567|5678|6789", password):
        return 12
    if re.search(r"9876|8765|7654|6543|5432|4321", password):
        return 12
    # === 4. Годы (очень частый паттерн!) ===
    if re.search(r"(19\d{2}|20\d{2})", password):
        return 5
    return 0
def detect_repetitions(password):
    repeats = len(password) - len(set(password))
    return repeats * 0.7
def detect_simple_masks(password):
    p = password.lower()
    # только цифры
    if password.isdigit():
        return 25
    # только буквы
    if password.isalpha():
        return 15
    # слово + цифры
    if re.fullmatch(r"[a-z]+\d{2,}", p):
        return 18
    # слово + год (1900-2029)
    if re.search(r"(19\d{2}|20[0-2]\d)", p):
        return 20
    # слово + цифры + 1 символ
    if re.fullmatch(r"[a-z]+\d+[@#$$%^&*]$", p):
        return 18
    # повторяемый паттерн
    if re.fullmatch(r"(abc|123|qwe)+", p):
        return 20
    return 0
# НОРМАЛИЗАЦИЯ ЭНТРОПИИ
def normalize_entropy(entropy, penalties):
    text = " ".join(penalties).lower()
    cap = entropy
    if "утечк" in text or "популярный пароль" in text:
        cap = min(cap, 40)
    if "простые маски" in text:
        cap = min(cap, 60)

```

```

        if "повторяющийся шаблон" in text:
            entropy *= 0.6
    if "простые последовательности" in text:
        cap = min(cap, 65)
        entropy = cap
        if "повторяющиеся" in text:
            entropy *= 0.95
        return entropy
def detect_pattern_repetition(password):
    """
    Ищет повторяющиеся блоки типа abcabcabс или aA1!aA1!
    """
    length = len(password)
    for size in range(1, length // 2 + 1):
        pattern = password[:size]
        if pattern * (length // size) == password:
            return 20 # сильный штраф
    return 0
# Анализ пароля
def analyze_password(password):
    penalties = []
    count = 0 # для совместимости
    # === БАЗОВАЯ ЭНТРОПИЯ ===
    charset = max(1, charset_size(password))
    entropy = len(password) * math.log2(charset)
    # ЛОГИКА ДЛИНЫ
    length = len(password)
    if length < 8:
        penalties.append("Слишкомкороткийпароль")
    entropy *= 0.6
    elif length < 12:
        penalties.append("Слишкомкороткийпароль")
    entropy *= 0.8
    elif length < 16:
        penalties.append("Пароль можно усилить увеличением длины")
    # === ЭВРИСТИКИ (считаем, но пока не применяем) ===
    rep = detect_repetitions(password)
    seq = detect_sequences(password)
    mask = detect_simple_masks(password)
    pattern = detect_pattern_repetition(password)
    # === ПРОВЕРКА УТЕЧЕК ===
    leak_level = check_local_leak(password)
    is_common = password.lower() in COMMON_PASSWORDS
    # ПРИОРИТЕТ: УТЕЧКИ
    if leak_level:
        penalties.append("Пароль найден в утечках — использовать нельзя")
    # ограничиваемэнтропию
    entropy = min(entropy, 25)
    if leak_level == "top1k":
        penalties.append("Критически распространённый пароль (TOP-1000)")
    entropy = min(entropy, 20)
    logging.warning(f"Leaked password detected")

```

```

        else:
            # ЭВРИСТИКИ (ТОЛЬКО ЕСЛИ НЕТ УТЕЧКИ)
    ifrep> 0:
        penalties.append(f"Повторяющиеся символы (-{rep:.1f} бит)")
    ifseq> 0:
        penalties.append(f"Простые последовательности (-{seq} бит)")
    if mask > 0:
        penalties.append(f"Простые маски (-{mask} бит)")
        if pattern > 0:
            penalties.append(f"Повторяющийся шаблон (-{pattern}
бит)")
        if is_common:
            penalties.append("Популярныйпароль (-25 бит)")
    entropy -= 25
        # применяем штрафы
        entropy -= (rep + seq + mask + pattern)
    # === защита от отрицательных значений ===
    entropy = max(0.0, entropy)
    # === НОРМАЛИЗАЦИЯ ===
    normalized_entropy = normalize_entropy(entropy, penalties)
        # === ЛОГ ===
        logging.info(
            f"Password analyzed | len={len(password)} | "
            f"entropy={normalized_entropy:.2f} | penalties={penalties}"
        )
        return normalized_entropy, penalties, count, leak_level
# Уровень стойкости
def strength_level(entropy):
    if entropy < 28: return "very_weak"
    elif entropy < 36: return "weak"
    elif entropy < 60: return "medium"
    else: return "strong"
def expert_comment(entropy, penalties):
    level = strength_level(entropy)
    has_penalties = len(penalties) > 0
    if level in ("very_weak", "weak"):
        return (
            "Пароль обладает низкой стойкостью и может быть взломан
"
            "при словарных или комбинированных атаках."
        )
    if level == "medium":
        if has_penalties:
            return (
                "Пароль имеет средний уровень защиты. "
                "Обнаружены предсказуемые элементы, снижающие его
стойкость."
            )
    return (
        "Пароль обеспечивает базовый уровень защиты, "
        "однако рекомендуется увеличить его сложность."
    )
ifhas_penalties:

```

```

return (
    "Пароль устойчив к перебору, однако содержит "
    "предсказуемые шаблоны, ограничивающие его стойкость."
)
return (
    "Пароль обладает высокой стойкостью и не содержит "
    "известных уязвимых конструкций."
)
# Время взлома
def crack_time_bruteforce(entropy_bits, guesses_per_second):
    try:
        total = 2 ** entropy_bits
    except OverflowError:
        return float("inf")
    average = total / 2
    if guesses_per_second <= 0:
        return float("inf")
    return average / guesses_per_second
ATTACK_SPEED = {
    "pc": 1_000_000,
    "gpu": 100_000_000,
    "botnet": 10_000_000_000,
    "supercomputer": 100_000_000_000
}

def format_time(seconds):
    if seconds == float("inf"):
        return "Практически невозможен"
    if seconds < 1:
        return "Менее секунды"
    if seconds < 60:
        return f"{int(seconds)} секунд"
    minutes = seconds / 60
    if minutes < 60:
        return f"{int(minutes)} минут"
    hours = minutes / 60
    if hours < 24:
        return f"{int(hours)} часов"
    days = hours / 24
    if days < 365:
        return f"{int(days)} дней"
    years = days / 365
    if years < 10:
        return f"{int(years)} лет"
    if years < 100:
        return "Десятки лет"
    if years < 1_000:
        return "Века"
    if years < 1_000_000:
        return "Тысячелетия"
    return "Практически невозможен"
BASE_DIR = os.path.dirname(__file__)
ML_MODEL_PATH = os.path.join(BASE_DIR, "ml", "model.joblib")
try:

```

```

ml_model = joblib.load(ML_MODEL_PATH)
except Exception:
    ml_model = None
def ml_password_assessment(password: str, entropy: float):
    if not ml_model or not password:
        return {"is_deceptive": False, "confidence": 0.0}
    # ML применяем только к формально "не слабым"
    if len(password) < 8 or entropy < 30:
        return {"is_deceptive": False, "confidence": 0.0}
    features = extract_features(password, entropy)
    X = [[
        features["length"],
        features["entropy"],
        features["digit_ratio"],
        features["upper_ratio"],
        features["special_ratio"],
        features["has_sequence"],
        features["has_repeat"],
        features["has_common_pattern"],
    ]]
    try:
        prob = ml_model.predict_proba(X)[0][1]
    except Exception:
        return {"is_deceptive": False, "confidence": 0.0}
    is_deceptive = prob >= 0.75
    return {
        "is_deceptive": bool(is_deceptive),
        "confidence": round(prob, 2)
    }
# Безопасная оценка времени взлома
def estimate_crack_time(entropy, attack_type="pc", leak_level=None):
    """
    Корректирует время взлома для паролей из утечек
    и популярных комбинаций.
    """
    # Пароли из TOP-1000
    if leak_level == "top1k":
        return "Менее секунды"
    # Пароли из TOP-100K
    if leak_level == "top100k":
        return "Несколько секунд"
    # Обычный расчёт
    guesses = ATTACK_SPEED.get(
        attack_type,
        ATTACK_SPEED["pc"]
    )
    seconds = crack_time_bruteforce(entropy, guesses)
    return format_time(seconds)
def build_recommendations(password, entropy, penalties, leak_level):
    parts = []
    # === УТЕЧКИ (приоритет) ===
    if leak_level == "top1k":

```

```

        return ["Пароль найден в самых распространённых –
использовать нельзя"]
elif leak_level == "top100k":
    return ["Пароль найден в утечках – рекомендуется заменить"]
# === БАЗОВЫЕ ПРОВЕРКИ ===
    length = len(password)
    if length < 12:
        parts.append("увеличить длину пароля")
elif length < 16 and entropy < 60:
    parts.append("можно увеличить длину пароля")
# недостающие типы символов
missing = []
if not any(c.islower() for c in password):
    missing.append("строчные буквы")
if not any(c.isupper() for c in password):
    missing.append("заглавные буквы")
if not any(c.isdigit() for c in password):
    missing.append("цифры")
if not any(not c.isalnum() for c in password):
    missing.append("специальные символы")
if missing:
    parts.insert(0, "добавить " + ", ".join(missing))
# === ЭВРИСТИКИ ===
if any("последовательности" in p.lower() for p in penalties):
    parts.append("избегайте последовательностей")
if any("маски" in p.lower() for p in penalties):
    parts.append("избегайте предсказуемых шаблонов")
if any("повторяющиеся" in p.lower() for p in penalties):
    parts.append("избегайте повторяющихся символов")
# === ФИНАЛ ===
    if not parts:
        if entropy >= 60:
            return ["Хорошая новость: у вас стойкий и безопасный
пароль 🍀"]
    return []
    # объединяем всё в одну строку
    text = "Рекомендуется: " + " и ".join(parts)

return [text]

```

Файл «views.py»

```

# analyzer/views.py
import string
import secrets
import math
from django.views.decorators.http import require_POST
from .utils import estimate_crack_time
import json
from django.http import JsonResponse
from django.shortcuts import render
from .utils import build_recommendations
from django.views.decorators.csrf import ensure_csrf_cookie

```

```

from .utils import (
    analyze_password,
    strength_level,
    charset_size,
    crack_time_bruteforce,
    ATTACK_SPEED,
    format_time,
    expert_comment,
    ml_password_assessment
)
#Страница проверки
def index(request):
    return render(request, "analyzer/index.html")
@ensure_csrf_cookie
def index(request):
    return render(request, "analyzer/index.html")
@require_POST
def check_password(request):
    try:
        data = json.loads(request.body)
    except Exception:
        return JsonResponse({"error": "Invalid JSON"}, status=400)
    pwd = data.get("password", "")
    attack = data.get("attack", "pc")
    #АНАЛИЗ
    entropy, penalties, _, leak_level = analyze_password(pwd)
    #БАЗОВЫЕ ПАРАМЕТРЫ
    cs = charset_size(pwd)
    raw_entropy = len(pwd) * math.log2(max(1, cs)) if pwd else 0
    level = strength_level(entropy)
    comment = expert_comment(entropy, penalties)
    #ВРЕМЯ ВЗЛОМА
    if pwd:
        speed = ATTACK_SPEED.get(attack, ATTACK_SPEED["pc"])
        seconds = crack_time_bruteforce(entropy, speed)
        crack_time = format_time(seconds)
    else:
        seconds = 0
        crack_time = "-"
    #ML
    ml_result = ml_password_assessment(pwd, raw_entropy)
    if ml_result["is_deceptive"]:
        entropy *= 0.8
        level = strength_level(entropy)
    # РЕКОМЕНДАЦИИ
    recommendations = build_recommendations(
        pwd,
        entropy,
        penalties,
        leak_level
    )
    #ПРОВЕРКА УТЕЧКИ
    is_leaked = leak_level is not None

```

```

return JsonResponse({
    "entropy": round(entropy, 2),
    "level": level,
    "length": len(pwd),
    "charset": cs,
    "crack_time": crack_time,
    "crack_seconds": seconds,
    "penalties": penalties,
    "expert_comment": comment,
    "ml": ml_result,
    "is_leaked": is_leaked,
    "recommendations": recommendations
})

#Генератор
def password_generator_page(request):
    return render(request, "analyzer/generator.html")
@require_POST
def generate_password(request):
    try:
        data = json.loads(request.body)
    except Exception:
        return JsonResponse({"error": "Invalid JSON"}, status=400)
    #ПАРАМЕТРЫ
    try:
        length = int(data.get("length", 12))
    except Exception:
        length = 12
    use_digits = data.get("digits") == 1
    use_upper = data.get("upper") == 1
    use_lower = data.get("lower") == 1
    use_symbols = data.get("symbols") == 1
    #ВАЛИДАЦИЯ ДЛИНЫ
    length = max(4, min(length, 64))
    lower = string.ascii_lowercase
    upper = string.ascii_uppercase
    digits = string.digits
    symbols = "!@#$%^&*()-_+[]{}<>?/"
    charset = ""
    required_chars = []
    if use_lower:
        charset += lower
        required_chars.append(secrets.choice(lower))
    if use_upper:
        charset += upper
        required_chars.append(secrets.choice(upper))
    if use_digits:
        charset += digits
        required_chars.append(secrets.choice(digits))
    if use_symbols:
        charset += symbols
        required_chars.append(secrets.choice(symbols))
    # если пользователь ничего не выбрал
    if not charset:

```

```

charset = lower + upper + digits
    required_chars.extend([
        secrets.choice(lower),
        secrets.choice(upper),
        secrets.choice(digits)
    ])
    remaining_length = max(0, length - len(required_chars))
    password = required_chars + [
        secrets.choice(charset) for _ in range(remaining_length)
    ]
    secrets.SystemRandom().shuffle(password)
    return JsonResponse({"password": "".join(password)})
#Эмуляция атаки
@require_POST
def attack_simulation(request):
    try:
        data = json.loads(request.body)
    except Exception:
        return JsonResponse({"error": "Invalid JSON"}, status=400)
    pwd = data.get("password", "")
    attack_type = data.get("attack", "pc")
    ifpwd:
        # Используем ОСНОВНОЙ анализ системы
        entropy, penalties, _, leak_level = analyze_password(pwd)
        cs = charset_size(pwd)
        # Корректная оценка времени
        readable = estimate_crack_time(
            entropy,
            attack_type,
            leak_level
        )
        # Для совместимости
        seconds = 0
    else:
        entropy = 0
        cs = 0
        seconds = 0
        readable = "-"
    return JsonResponse({
        "entropy": round(entropy, 2),
        "charset": cs,
        "time_seconds": seconds,
        "time_readable": readable,
        "attack_type": attack_type,
    })

```

Файл Urls.py

```

from django.urls import path
from . import views
urlpatterns = [
    path("", views.index, name="analyzer_index"),
    path("check/", views.check_password, name="check_password"),
    # Новая страница

```

```

    path("generator/", views.password_generator_page,
name="password_generator_page"),
    # генерации
    path("generate/", views.generate_password,
name="generate_password"),
    path("attack/", views.attack_simulation,
name="attack_simulation"),
]

```

Файл«Features.py»

```

# analyzer/ml/features.py
from collections import Counter
COMMON_PATTERNS = [
    "password", "qwerty", "admin", "welcome",
    "abc", "user", "login", "letmein"
]
def max_sequence_length(pw: str) -> int:
    pw = pw.lower()
    max_len = 1
    for i in range(len(pw) - 1):
        cur = 1
        while (
            i + cur < len(pw)
            and ord(pw[i + cur]) - ord(pw[i + cur - 1]) in (1, -1)
        ):
            cur += 1
        max_len = max(max_len, cur)
    return max_len
def extract_features(password: str, entropy: float) -> dict:
    length = len(password)
    if length == 0:
        return {
            "length": 0,
            "entropy": 0,
            "digit_ratio": 0,
            "upper_ratio": 0,
            "special_ratio": 0,
            "has_sequence": 0,
            "has_repeat": 0,
            "has_common_pattern": 0,
        }
    digits = sum(c.isdigit() for c in password)
    uppers = sum(c.isupper() for c in password)
    specials = sum(not c.isalnum() for c in password)
    # последовательности
    has_sequence = int(max_sequence_length(password) >= 3)
    # повторы
    counts = Counter(password)
    repeat_ratio = sum(v - 1 for v in counts.values() if v > 1) /
length
    has_repeat = int(repeat_ratio >= 0.2)
    # известные шаблоны

```

```

pw_lower = password.lower()
has_common_pattern = int(any(p in pw_lower for p in
COMMON_PATTERNS))
return {
    "length": length,
    "entropy": entropy,
    "digit_ratio": digits / length,
    "upper_ratio": uppers / length,
    "special_ratio": specials / length,
    "has_sequence": has_sequence,
    "has_repeat": has_repeat,
    "has_common_pattern": has_common_pattern,
}

```

Файл«Train.py»

```

# analyzer/ml/train.py
import random
import joblib
import math
import string
from pathlib import Path
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score
from analyzer.ml.features import extract_features
MODEL_PATH = Path(__file__).parent / "model.joblib"
NUM_GOOD = 2500
NUM_DECEPTIVE = 2500
ALL_CHARS = string.ascii_letters + string.digits + !@#$%^&*()_+
=[]{ }|;:,.<>?
def calc_entropy(password: str) -> float:
    charset = 0
    if any(c.islower() for c in password): charset += 26
    if any(c.isupper() for c in password): charset += 26
    if any(c.isdigit() for c in password): charset += 10
    if any(not c.isalnum() for c in password): charset += 33
    return len(password) * math.log2(max(1, charset))
def generate_dataset():
    data = []
    # --- хорошие (реально случайные)
    for _ in range(NUM_GOOD):
        length = random.randint(12, 18)
        pwd = "".join(random.choices(ALL_CHARS, k=length))
        data.append((pwd, 0))
    # --- обманчивые
    bases = [
        "password", "qwerty", "admin", "welcome",
        "abc", "user", "login"
    ]
    for _ in range(NUM_DECEPTIVE):
        base = random.choice(bases)
        pwd = (

```

```

        base.capitalize() +
        str(random.randint(10, 9999)) +
        random.choice("!@#")
    )
    data.append((pwd, 1))
    random.shuffle(data)
    return data
dataset = generate_dataset()
X, y = [], []
for pwd, label in dataset:
    entropy = calc_entropy(pwd)
    f = extract_features(pwd, entropy)
    X.append([
        f["length"],
        f["entropy"],
        f["digit_ratio"],
        f["upper_ratio"],
        f["special_ratio"],
        f["has_sequence"],
        f["has_repeat"],
        f["has_common_pattern"],
    ])
    y.append(label)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42, stratify=y
)
model = LogisticRegression(
    max_iter=3000,
    class_weight="balanced"
)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
joblib.dump(model, MODEL_PATH)
print(f"Модель сохранена: {MODEL_PATH}")

```

Файл Generate_leaks.py

```

import hashlib
from pathlib import Path
BASE_DIR = Path(__file__).resolve().parent.parent # analyzer/
def generate(input_file, output_file, limit=None):
    hashes = set()
    with open(input_file, "r", encoding="latin-1", errors="ignore")
as f:
    for i, line in enumerate(f):
        if limit and i >= limit:
            break
        pwd = line.strip()
        if not pwd:
            continue
        sha1 = hashlib.sha1(pwd.encode("utf-
8")).hexdigest().upper()

```

					ПГУ 1.090301.13.001 ПЗ	Лист
						81
Изм.	Лист	№докум.	Подпись	Дата		

```

        prefix = sha1[:5]
        suffix = sha1[5:]
        hashes.add(f"{prefix}:{suffix}")
    with open(output_file, "w", encoding="utf-8") as out:
        for h in hashes:
            out.write(h + "\n")
    print(f"{output_file}: {len(hashes)} entries")
generate(
    BASE_DIR / "data" / "top_1k.txt",
    BASE_DIR / "data" / "top_1k_sha1.txt"
)
generate(
    BASE_DIR / "data" / "top_100k.txt",
    BASE_DIR / "data" / "top_100k_sha1.txt",
    limit=100000
)

```

Файл«Index.html»

```

<!DOCTYPE html>
<html lang="ru">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width,initial-scale=1"
/>
<title>Проверка стойкости пароля – GlassUI</title>
<!-- ===== CSS: Glassmorphism + cyber accent ===== -->
<style>
:root{
    --bg: #0f1720;
    --panel: rgba(255,255,255,0.06);
    --glass-border: rgba(255,255,255,0.08);
    --glass-blur: 10px;
    --card-grad: linear-gradient(135deg, rgba(255,255,255,0.03),
    rgba(255,255,255,0.01));
    --accent: #6EE7B7;
    --accent-2: #7C83FD;
    --danger: #FF7A7A;
    --muted: rgba(255,255,255,0.6);
    --radius: 16px;
    --shadow: 0 10px 30px rgba(2,6,23,0.6);
    --glass-inner: rgba(255,255,255,0.02);
    --glass-border-2: rgba(255,255,255,0.04);
    font-family: Inter, ui-sans-serif, system-ui, -apple-system,
    "Segoe UI", Roboto, "Helvetica Neue", Arial;
    color-scheme: dark;
}
html,body{height:100%;margin:0;background:linear-
gradient(180deg,#071026 0%, #0f1720 100%);-webkit-font-
smoothing:antialiased;}
body::before{
    content:"";
    position:fixed;inset:0;pointer-events:none;

```

					ПГУ 1.090301.13.001 ПЗ	Лист
Изм.	Лист	№докум.	Подпись	Дата		82

```

    background:radial-gradient(800px 400px at 10% 10%,
    rgba(124,131,253,0.06), transparent 8%),
    radial-gradient(600px 300px at 90% 90%,
    rgba(110,231,183,0.03), transparent 8%);
    mix-blend-mode:overlay;
}
.nav {
    position:fixed;left:0;right:0;top:14px;display:flex;justify-
content:center;z-index:40;
    pointer-events:auto;
}
.nav-inner{
    width:100%;max-width:980px;display:flex;align-
items:center;justify-content:space-between;
    padding:10px 18px;border-radius:12px;
    background:linear-gradient(180deg, rgba(255,255,255,0.03),
    rgba(255,255,255,0.02));
    border:1px solid var(--glass-border);
    backdrop-filter: blur(var(--glass-blur)) saturate(120%);
    box-shadow: var(--shadow);
    gap:16px;
}
.brand { display:flex;align-items:center;gap:12px; }
.logo {
    width:36px;height:36px;border-radius:9px;
    background:linear-gradient(135deg,var(--accent),var(--accent-2));
    display:inline-grid;place-items:center;color:#061221;font-
weight:700;font-size:14px;
    box-shadow: 0 6px 18px rgba(124,131,253,0.12), inset 0 -2px 6px
    rgba(0,0,0,0.25);
}
.nav-links{display:flex;gap:18px;align-items:center;}
.nav-links a{
    color:var(--muted); text-decoration:none; font-weight:600;
padding:6px 8px; border-radius:8px;
}
.nav-links a.active, .nav-links a:hover{ color: white; box-shadow:
inset 0 -2px 0 var(--accent); }
.wrap{width:100%;max-width:980px;margin:120px auto
40px;padding:22px;box-sizing:border-box;display:flex;gap:28px;align-
items:flex-start;}
.card {
    flex:1; min-width:320px; background:var(--card-grad);
    border-radius:var(--radius);
    padding:28px;
    border:1px solid var(--glass-border-2);
    backdrop-filter: blur(var(--glass-blur)) saturate(120%);
    box-shadow: 0 18px 50px rgba(2,6,23,0.6);
}
h2{margin:0 0 12px 0;color:#eafaf1;font-size:20px;letter-
spacing:0.2px}
.input-group{position:relative;margin:10px 0 18px 0;}
.input {

```

```

width:100%;padding:14px 46px 14px 16px;border-
radius:12px;border:1px solid rgba(255,255,255,0.06);
background: linear-gradient(180deg, rgba(255,255,255,0.01),
rgba(255,255,255,0.00));

color:#f8fafc;font-size:15px; outline:none; box-sizing:border-box;
transition: box-shadow .18s, border-color .18s, transform .08s;
}
.input:focus { box-shadow: 0 6px 18px rgba(110,231,183,0.06);
border-color: rgba(110,231,183,0.32); transform:translateY(-1px); }
.icon-btn {
position:absolute; right:8px; top:50%; transform:translateY(-50%);
border:0; background:transparent; cursor:pointer;padding:8px;border-
radius:8px;color:var(--muted);
}
.icon-btn:hover{ color:white; background:rgba(255,255,255,0.03); }
.strength-wrap{margin-bottom:14px}
.strength-label{display:flex;justify-content:space-between;align-
items:center;margin-bottom:8px;color:var(--muted);font-
weight:600;font-size:13px}
.bar {
height:14px;border-radius:12px;background:linear-gradient(90deg,
rgba(255,255,255,0.03), rgba(255,255,255,0.01));
border:1px solid rgba(255,255,255,0.02); overflow:hidden;
position:relative;
}
.bar-fill{
height:100%; width:0%; background: linear-gradient(90deg,
rgba(255,122,122,0.9), rgba(255,206,110,0.95),
rgba(124,131,253,0.95));
transition: width .6s cubic-bezier(.2,.9,.2,1), background .3s;
box-shadow: 0 8px 18px rgba(124,131,253,0.12);
}
.bar-fill{
height:100%; width:0%;
transition: width .8s cubic-bezier(.22,1,.36,1), background .4s
ease;
background-size: 200% 100%;
animation: gradientFlow 6s linear infinite;
}
.info-grid { display:grid; grid-template-columns: 1fr 1fr; gap:12px;
margin-top:12px; color:var(--muted); font-size:14px;}
.info-item { background: rgba(255,255,255,0.02);
padding:10px;border-radius:10px;border:1px solid
rgba(255,255,255,0.02); }
.info-item b{display:block;color:#eafaf1;font-size:13px;margin-
bottom:6px}
.checks{margin-top:14px; display:flex; gap:10px; flex-wrap:wrap}
.check { padding:10px 12px;border-
radius:10px;background:rgba(255,255,255,0.02); border:1px solid
rgba(255,255,255,0.02); color:var(--muted); display:flex; gap:10px;
align-items:center; min-width:140px; }

```

```

.check.ok { border-color: rgba(110,231,183,0.12); color:var(--
accent); box-shadow: inset 0 -2px 0 rgba(110,231,183,0.06); }
.check.bad { border-color: rgba(255,122,122,0.08); color:var(--
danger); box-shadow: inset 0 -2px 0 rgba(255,122,122,0.04); }
.penalties { margin-top:14px; color:#ffddd7; font-size:13px; }
.penalties ul{ margin:8px 0 0 18px; padding:0; }
.side {
  width:320px; min-width:260px;
  border-radius:var(--radius);
  padding:20px;
  background:linear-gradient(180deg, rgba(255,255,255,0.02),
rgba(255,255,255,0.01));
  border:1px solid rgba(255,255,255,0.03);
  backdrop-filter: blur(calc(var(--glass-blur) - 3px));
  box-shadow: 0 12px 30px rgba(2,6,23,0.6);
}
.side h3{margin:0 0 12px 0;color:var(--muted);font-size:14px}
.select {
  width:100%;
  padding:12px;
  border-radius:10px;
  border:1px solid rgba(255,255,255,0.04);
  background-color: rgba(15,23,32,0.95);
  color:#f6faf8;
  appearance:none;
}
.select option {
  background-color: #0f1720;
  color: #f6faf8;
}
.crack { margin-top:14px;padding:12px;border-radius:10px;border:1px
dashed rgba(255,255,255,0.03); color:var(--muted); font-size:14px;
min-height:44px; display:flex;align-items:center;justify-
content:center; }
.small { font-size:12px;color:rgba(255,255,255,0.45); margin-
top:12px; text-align:center }
@media (max-width:920px){
  .wrap{flex-direction:column;padding:18px}
  .side{width:100%;order:2}
  .card{order:1}
  .nav-inner{padding:10px}
}
/* ===== Tooltip ===== */
.tooltip {
  position: relative;
  display: inline-flex;
  align-items: center;
  gap: 6px;
  cursor: help;
}
.tooltip-icon {
  font-size: 12px;
  color: var(--muted);

```

```

border: 1px solid rgba(255,255,255,0.25);
border-radius: 50%;
width: 16px;
height: 16px;
display: inline-grid;
place-items: center;
line-height: 1;
}
.tooltip-text {
position: absolute;
bottom: calc(100% + 8px);
left: 0;
min-width: 220px;
padding: 10px 12px;
border-radius: 10px;
background: rgba(15,23,32,0.96);
border: 1px solid rgba(255,255,255,0.08);
box-shadow: 0 12px 30px rgba(2,6,23,0.6);
font-size: 12px;
color: #eafaf1;
opacity: 0;
transform: translateY(4px);
pointer-events: none;
transition: opacity .2s ease, transform .2s ease;
z-index: 50;
}
.tooltip:hover .tooltip-text {
opacity: 1;
transform: translateY(0);
}
@keyframes gradientFlow{
0%{background-position:0% 50%}
100%{background-position:200% 50%}
}
#ml-warning{
border-left:4px solid #ff7a7a;
animation:pulse 2s infinite;
}
@keyframes pulse{
0%{box-shadow:0 0 0 rgba(255,122,122,.2)}
50%{box-shadow:0 0 12px rgba(255,122,122,.35)}
100%{box-shadow:0 0 0 rgba(255,122,122,.2)}
}
.penalties ul li{
list-style:none;
margin-bottom:6px;
display:flex;
gap:6px;
align-items:flex-start;
}
.penalties ul li::before{
content:"⚠";
}

```

```

#recommendations-box {
  font-size: 13px;
  opacity: 0.85;
  margin-top: 10px;
}
#recommendations-box h3 {
  font-size: 14px;
  margin-bottom: 6px;
  opacity: 0.7;
  font-weight: 500;
}
#recommendations-list {
  padding-left: 16px;
  margin: 0;
}
#recommendations-list li {
  margin-bottom: 4px;
  line-height: 1.4;
}

#recommendations-box {
  background: rgba(255,255,255,0.04);
  border: 1px solid rgba(255,255,255,0.08);
  border-radius: 10px;
  padding: 10px 12px;
}
</style>
</head>
<body>
<div class="nav" role="navigation" aria-label="Главное меню">
<div class="nav-inner" role="toolbar">
<div class="brand">
<div class="logo">PW</div>
<div>
<div style="font-size:13px;color:var(--muted);font-
weight:700">PasswordLab</div>
<div style="font-size:11px;color:rgba(255,255,255,0.35);margin-
top:2px">Визуализация стойкости паролей</div>
</div>
</div>
<div class="nav-links" role="menubar" aria-hidden>
<a href="/" class="active" aria-current="page">Проверка пароля</a>
<a href="/generator">Генератор</a>
</div>
</div>
</div>
<div class="wrap" role="main">
<section class="card" aria-labelledby="page-title">
<h2 id="page-title">Проверка стойкости пароля</h2>

<div class="input-group">
<input id="password-input" class="input" type="password"
  placeholder="Введите пароль..." aria-label="Поле ввода пароля" />

```

```

<button id="toggle-password" class="icon-btn" title="Показать/скрыть
пароль" aria-pressed="false" aria-label="Показать пароль">
<svg id="eye-icon" viewBox="0 0 24 24" width="18" height="18"
fill="currentColor" aria-hidden>
<path d="M12 5C7 5 2.73 8.11 1 12c1.73 3.89 6 7 11 7s9.27-3.11 11-
7c-1.73-3.89-6-7-11-7zm0 12c-2.76 0-5-2.24-5-5s2.24-5 5-5 5 2.24 5
5-2.24 5-5 5z"/>
</svg>
</button>
</div>
<div class="strength-wrap" aria-hidden>
<div class="strength-label">
<div style="font-size:13px;color:var(--muted)">Стойкость
пароля</div>
<div id="level" style="font-weight:700;color:#fff">—</div>
</div>
<div class="bar" role="progressbar" aria-valuemin="0" aria-
valuemax="100" aria-valuenow="0">
<div id="strength-fill" class="bar-fill" style="width:0%"></div>
</div>
<div class="info-grid" aria-live="polite">
<div class="info-item">
<b>Длина</b>
<div id="len">0</div>
</div>
<div class="info-item">
<b class="tooltip">
Набор символов
<span class="tooltip-icon">?</span>
<span class="tooltip-text">
Количество уникальных символов, которые реально
используются
в пароле (строчные, заглавные, цифры, спецсимволы).
Чем больше набор — тем выше потенциальная стойкость.
</span>
</b>
<div id="charset">0</div>
</div>
<div class="info-item">
<b class="tooltip">
Энтропия
<span class="tooltip-icon">?</span>
<span class="tooltip-text">
Энтропия — это количественная оценка сложности пароля в
битах.
Учитывает длину, размер набора символов и снижает оценку
за предсказуемые шаблоны.
</span>
</b>
<div id="entropy">0</div>
</div>
<div class="info-item">

```

```

<b class="tooltip">
    Время взлома (оценка)
<span class="tooltip-icon">?</span>
<spanclass="tooltip-text">
    Приблизительное время полного перебора пароля
    при выбранной модели атакующего.
    Значение носит демонстрационный характер.
</span>
</b>
<div id="crack_time">-</div>
</div>
</div>
<div class="checks" aria-hidden="false">
<div id="chk-lower" class="check bad" title="Строчные
буквы">Строчные <span style="margin-left:auto">X</span></div>
<div id="chk-upper" class="check bad" title="Заглавные
буквы">Заглавные <span style="margin-left:auto">X</span></div>
<div id="chk-digit" class="check bad" title="Цифры">Цифры <span
style="margin-left:auto">X</span></div>
<div id="chk-spec" class="check bad" title="Спецсимволы">Символы
<span style="margin-left:auto">X</span></div>
</div>
<div id="expert-comment"
    style="margin-top:14px;
        padding:12px;
        border-radius:10px;
        background:rgba(255,255,255,0.03);
        border:1px solid rgba(255,255,255,0.04);
        color:#eafaf1;
        font-size:14px;
        display:none;"
    aria-live="polite">
</div>
<div id="ml-warning"
    style="margin-top:12px;
        padding:10px;
        border-radius:10px;
        background:rgba(255,122,122,0.08);
        border:1px solid rgba(255,122,122,0.2);
        color:#ffbaba;
        font-size:13px;
        display:none;">
</div>
<div class="penalties" id="penalty-section" style="display:none"
    aria-live="polite">
<div style="font-weight:700;color:#ffd6d6">Причины снижения
оценки:</div>
<ul id="penalty-list"></ul>
</div>
<div id="recommendations-box" style="display:none; margin-
top:15px;">
<h3>Рекомендации</h3>

```

```

<ul id="recommendations-list"></ul>
</div>
<div class="small" style="margin-top:14px">
Пожалуйста, не вводите реальные рабочие пароли в демонстрационном
режиме.
</div>
</section>
<aside class="side" aria-label="Параметры атаки">
<h3>Симуляция атаки</h3>
<label for="attack_type" style="display:block;font-
size:13px;color:var(--muted);margin-bottom:6px">Тип
атакующего</label>
<select id="attack_type" class="select" aria-label="Тип атаки">
<option value="pc">Обычный ПК</option>
<option value="gpu">Игровая GPU</option>
<option value="botnet">Ботнет</option>
<option value="supercomputer">Суперкомпьютер</option>
</select>
<div class="crack" id="crack-box" aria-live="polite" style="margin-
top:12px">
<div>
<div style="font-size:13px;color:var(--muted);margin-
bottom:6px">Оценочное время взлома</div>
<div id="crack_time_side" style="font-weight:700;color:#fff">—</div>
</div>
</div>
<div style="margin-top:16px">

<button id="simulate-btn" style="width:100%;padding:10px;border-
radius:10px;border:0;background:linear-gradient(90deg,var(--
accent),var(--accent-2));color:#051221;font-
weight:700;cursor:pointer">Запустить эмуляцию</button>
</div>
<div class="small">Моделирование — приблизительное (для
демонстрации).</div>
</aside>
</div>
<script>
function setText(id, text){
    const el=document.getElementById(id);
    if(el) el.textContent=text;
}
function updateChecks(pwd){
    const lower=/[a-z]/.test(pwd);
    const upper=/[A-Z]/.test(pwd);
    const digit=/[0-9]/.test(pwd);
    const spec=/[^A-Za-z0-9]/.test(pwd);
    const set=(id,ok)=>{
        const el=document.getElementById(id);
        if(!el) return;
        el.classList.toggle('ok',ok);
        el.classList.toggle('bad',!ok);
        el.querySelector('span').textContent=ok?'✓':'✗';
    }
}

```

```

    };
    set('chk-lower', lower);
    set('chk-upper', upper);
    set('chk-digit', digit);
    set('chk-spec', spec);
}
let lastFetchController=null;
function updatePasswordInfo() {
    const pwd=document.getElementById('password-input').value||'';
    const attack=document.getElementById('attack_type').value||'pc';
    updateChecks(pwd);
    // ===== Защита от пустого ввода =====
    if(pwd.length===0) {
        setText('len', 0);
        setText('charset', 0);
        setText('entropy', 0);
        setText('level', '-');
        document.getElementById('strength-fill').style.width='0%';
        document.getElementById('penalty-section').style.display='none';
        document.getElementById('expert-comment').style.display='none';
        document.getElementById('ml-warning').style.display='none';
        const recBox = document.getElementById('recommendations-box');
        const recList = document.getElementById('recommendations-list');
        if(recBox) recBox.style.display = 'none';
        if(recList) recList.innerHTML = '';
        return;
    }
    if(lastFetchController){ lastFetchController.abort(); }
    lastFetchController=new AbortController();
    const signal=lastFetchController.signal;
    function getCookie(name) {
        let cookieValue = null;
        if (document.cookie && document.cookie !== "") {
            const cookies = document.cookie.split(";");
            for (let cookie of cookies) {
                cookie = cookie.trim();
                if (cookie.startsWith(name + "=")) {
                    cookieValue =
decodeURIComponent(cookie.substring(name.length + 1));
                    break;
                }
            }
        }
        return cookieValue;
    }
    fetch("/check/", {
        method: "POST",
        headers: {
            "Content-Type": "application/json",
            "X-CSRFToken": getCookie("csrftoken")
        },
        body: JSON.stringify({
            password: pwd,

```

```

        attack: attack
    }},
    signal
  })
  .then(r=>{if(!r.ok)throw new Error();return r.json();})
  .then(data=>{
    setText('len',data.length??0);
    setText('charset',data.charset??0);
    setText('entropy',data.entropy??0);
    const expert=document.getElementById('expert-comment');
    if(data.expert_comment){
      expert.textContent=data.expert_comment;
      expert.style.display='block';
      if(data.level==='very_weak')expert.style.color='#ff7a7a';
      else if(data.level==='weak')expert.style.color='#ffb36b';
      else if(data.level==='medium')expert.style.color='#ffd86d';
      else if(data.level==='strong')expert.style.color='#6EE7B7';
      else expert.style.color='#eafaf1';
    } else expert.style.display='none';
    const mlBox=document.getElementById('ml-warning');
    if(data.ml && data.ml.is_deceptive){
      mlBox.style.display='block';
      if(data.level === 'strong'){
        mlBox.innerHTML =
          `<b>Дополнительный анализ:</b><br>
          △ Пароль формально стойкий, но содержит предсказуемые шаблоны`;
      } else {
        mlBox.innerHTML =
          `<b>Дополнительный анализ:</b><br>
          △ Пароль содержит предсказуемые шаблоны`;
      }
      mlBox.style.color = '#ffcc66'; // мягкийwarningцвет
    } else {
      mlBox.style.display='none';
    }
    const fill=document.getElementById('strength-fill');
    const lvl=data.level||'-';
    setText('level',lvl);
    if(lvl==='very_weak'){
      fill.style.width='18%';
      fill.style.background='linear-
gradient(90deg,#ff7a7a,#ffbaba)';
    }else if(lvl==='weak'){
      fill.style.width='38%';
      fill.style.background='linear-
gradient(90deg,#ffb36b,#ffdd99)';
    }else if(lvl==='medium'){
      fill.style.width='70%';
      fill.style.background='linear-
gradient(90deg,#ffd86d,#f39c12)';
    }else if(lvl==='strong'){
      fill.style.width='100%';
    }
  })

```

```

        fill.style.background='linear-
gradient(90deg,#6EE7B7,#7C83FD)';
    }else fill.style.width='0%';
    const sec=document.getElementById('penalty-section');
    const list=document.getElementById('penalty-list');
    // ===== РЕКОМЕНДАЦИИ =====
    const recBox = document.getElementById('recommendations-box');
    const recList = document.getElementById('recommendations-list');
    if(data.recommendations && data.recommendations.length){
        recBox.style.display = 'block';
        recList.innerHTML = '';
        data.recommendations.forEach(r => {
            const li = document.createElement('li');
            li.textContent = r;
            recList.appendChild(li);
        });
    } else {
        recBox.style.display = 'none';
    }
    list.innerHTML='';
    if(data.penalties && data.penalties.length){
        sec.style.display='block';
        data.penalties.forEach(p=>{
            const li=document.createElement('li');
            li.textContent=p;
            list.appendChild(li);
        });
    } else sec.style.display='none';
    })
    .catch(e=>{
        if(e.name!=='AbortError') console.error(e);
    });
}
document.getElementById('toggle-
password').addEventListener('click',function(){
    const input=document.getElementById('password-input');
    const icon=document.getElementById('eye-icon');
    const pressed=this.getAttribute('aria-pressed')==='true';
    if(!pressed){
        input.type='text';
        this.setAttribute('aria-pressed','true');
        icon.innerHTML='<path d=\"M2 2120 20-2 2-4.2-4.2C14.5 20 13.3 20
12 20c-5 0-9.27-3.11-11-7 1-2.3 2.7-4.3 4.8-5.7L2 412-2z\"/>';
    }else{
        input.type='password';
        this.setAttribute('aria-pressed','false');
        icon.innerHTML='<path d=\"M12 5C7 5 2.73 8.11 1 12c1.73 3.89 6 7
11 7s9.27-3.11 11-7c-1.73-3.89-6-7-11-7zm0 12c-2.76 0-5-2.24-5-
5s2.24-5 5-5 2.24 5 5-2.24 5-5 5z\"/>';
    }
});
document.getElementById('password-
input').addEventListener('input',()=>{

```

```

clearTimeout(window._pw_update_timeout);
window._pw_update_timeout=setTimeout(updatePasswordInfo,120);
});
document.getElementById('attack_type').addEventListener('change',updatePasswordInfo);
document.getElementById('simulate-
btn').addEventListener('click',()=>{
    const btn=document.getElementById('simulate-btn');
    btn.disabled=true;
    const orig=btn.textContent;
    btn.textContent='Идёт эмуляция...';
    const pwd=document.getElementById('password-input').value||'';
    const attack=document.getElementById('attack_type').value||'pc';
    function getCookie(name) {
        let cookieValue = null;
        if (document.cookie && document.cookie !== "") {
            const cookies = document.cookie.split(";");
            for (let cookie of cookies) {
                cookie = cookie.trim();
                if (cookie.startsWith(name + "=")) {
                    cookieValue =
decodeURIComponent(cookie.substring(name.length + 1));
                    break;
                }
            }
        }
        return cookieValue;
    }
    fetch("/attack/", {
        method: "POST",
        headers: {
            "Content-Type": "application/json",
            "X-CSRFToken": getCookie("csrftoken")
        },
        body: JSON.stringify({
            password: pwd,
            attack: attack
        })
    })
    .then(r=>r.json())
    .then(json=>{
        setText('crack_time',json.time_readable||'-');
        setText('crack_time_side',json.time_readable||'-');
        document.getElementById('crack-box').animate(
            [{boxShadow:'0 0 0 rgba(124,131,253,0)'},{boxShadow:'0 10px
30px rgba(124,131,253,0.12)'}]],
            {duration:450,easing:'ease-out'}
        );
    })
    .finally(()=>{
        setTimeout(()=>{
            btn.disabled=false;
            btn.textContent=orig;

```

```

        }, 600);
    });
});
(function initFromGenerator(){
    const pwd = sessionStorage.getItem("generated_password");
    if(pwd){
        const input = document.getElementById('password-input');
        input.value = pwd;
        // очищаем после использования
        sessionStorage.removeItem("generated_password");
        // триггер анализа
        input.dispatchEvent(new Event('input'));
    }
})();
</script>

</body>
</html>

```

Файл«Generator.html»

```

<!DOCTYPE html>
<html lang="ru">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width,initial-scale=1"
/>
<title>Генератор паролей – Glass UI</title>
<style>
:root{
    --bg: #0f1720;
    --panel: rgba(255,255,255,0.06);
    --glass-border: rgba(255,255,255,0.08);
    --glass-blur: 10px;
    --card-grad: linear-gradient(135deg, rgba(255,255,255,0.03),
    rgba(255,255,255,0.01));
    --accent: #6EE7B7;
    --accent-2: #7C83FD;
    --accent-copy: #2196F3;
    --accent-use: #FF9800;
    --radius: 16px;
    --shadow: 0 10px 30px rgba(2,6,23,0.6);
    font-family: Inter, ui-sans-serif, system-ui, -apple-system,
    "Segoe UI", Roboto, "Helvetica Neue", Arial;
    color-scheme: dark;
}
html,body{height:100%;margin:0;background:linear-
gradient(180deg,#071026 0%, #0f1720 100%);-webkit-font-
smoothing:antialiased;}
body::before{
    content:"";
    position:fixed;inset:0;pointer-events:none;

```

					ПГУ 1.090301.13.001 ПЗ	Лист
Изм.	Лист	№докум.	Подпись	Дата		95

```

    background:radial-gradient(800px 400px at 10% 10%,
    rgba(124,131,253,0.06), transparent 8%),
        radial-gradient(600px 300px at 90% 90%,
    rgba(110,231,183,0.03), transparent 8%);
    mix-blend-mode:overlay;
}
/* NAV */
.nav {
    position:fixed;left:0;right:0;top:14px;display:flex;justify-
content:center;z-index:40;
}
.nav-inner{
    width:100%;max-width:980px;display:flex;align-
items:center;justify-content:space-between;
    padding:10px 18px;border-radius:12px;
    background:linear-gradient(180deg, rgba(255,255,255,0.03),
    rgba(255,255,255,0.02));
    border:1px solid var(--glass-border);
    backdrop-filter: blur(var(--glass-blur)) saturate(120%);
    box-shadow: var(--shadow);
    gap:16px;
}
.brand { display:flex;align-items:center;gap:12px; }
.logo { width:36px;height:36px;border-radius:9px;background:linear-
gradient(135deg,var(--accent),var(--accent-2));display:inline-
grid;place-items:center;color:#061221;font-weight:700;font-
size:14px;box-shadow:0 6px 18px rgba(124,131,253,0.12), inset 0 -2px
6px rgba(0,0,0,0.25);}
.brand h1{margin:0;font-size:15px;color:rgba(255,255,255,0.6);font-
weight:600;}
.nav-links{display:flex;gap:18px;align-items:center;}
.nav-links a{ color:rgba(255,255,255,0.6); text-decoration:none;
font-weight:600; padding:6px 8px; border-radius:8px;}
.nav-links a.active, .nav-links a:hover{ color: white; box-shadow:
inset 0 -2px 0 var(--accent); }

/* WRAP */
.wrap{width:100%;max-width:980px;margin:120px auto
40px;padding:22px;display:flex;gap:28px;align-items:flex-start;}
/* CARD */
.card {
    flex:1; min-width:320px; background:var(--card-grad);
    border-radius:var(--radius); padding:28px;
    border:1px solid var(--glass-border);
    backdrop-filter: blur(var(--glass-blur)) saturate(120%);
    box-shadow: var(--shadow);
}
h2{margin:0 0 12px 0;color:#eafaf1;font-size:20px;letter-
spacing:0.2px}
/* INPUTS */
.input-group{position:relative;margin:10px 0 18px 0;}
.input {

```

```

width:100%;padding:14px 16px;border-radius:12px;border:1px solid
rgba(255,255,255,0.06);
background: linear-gradient(180deg, rgba(255,255,255,0.01),
rgba(255,255,255,0.00));
color:#f8fafc;font-size:15px; outline:none; box-sizing:border-box;
transition: box-shadow .18s, border-color .18s, transform .08s;
}
.input:focus { box-shadow: 0 6px 18px rgba(110,231,183,0.06);
border-color: rgba(110,231,183,0.32); transform:translateY(-1px); }
.icon-btn { position:absolute; right:8px; top:50%;
transform:translateY(-50%); border:0; background:transparent;
cursor:pointer;padding:8px;border-radius:8px;color:var(--muted);}

.icon-btn:hover{ color:white; background:rgba(255,255,255,0.03); }
/* CHECKBOX GROUP */
.checkbox-group { display:flex; flex-direction:column; gap:12px;
margin-bottom:22px;}
.checkbox-group label{ cursor:pointer; font-size:15px;
color:rgba(255,255,255,0.6);}
.checkbox-group input{ transform:scale(1.2); margin-right:6px; }
/* RESULT */
.result-
box{padding:16px;background:rgba(255,255,255,0.08);border:1px solid
var(--glass-border);border-radius:var(--radius);font-size:18px;text-
align:center;min-height:40px;user-select:all;margin-
bottom:22px;transition:background .3s, border .3s;}
.highlight{background: rgba(110,231,183,0.15);}
.result-box{
font-family: "JetBrains Mono", monospace;
letter-spacing: 1px;
}
.result-box.highlight{
box-shadow:0 0 20px rgba(110,231,183,.4);
}
/* BUTTONS */
.buttons-group{display:flex;gap:12px;}
button{flex:1;padding:13px;font-size:16px;border:none;border-
radius:var(--
radius);cursor:pointer;color:#fff;transition:background .25s,
transform .15s;}
button:hover{transform:translateY(-2px);}
#generate-btn{background:var(--accent);} #generate-
btn:hover{background:var(--accent-2);}
#copy-btn{background:var(--accent-copy);} #copy-
btn:hover{opacity:0.85;}
#use-btn{background:var(--accent-use);} #use-
btn:hover{opacity:0.85;}
/* RESPONSIVE */
@media (max-width:920px){ .wrap{flex-
direction:column;} .card{order:1;} }
</style>
</head>
<body>

```

```

<!-- NAV -->
<div class="nav">
<div class="nav-inner">
<div class="brand">
<div class="logo">PW</div>
<div>
<div style="font-size:13px;color:rgba(255,255,255,0.6);font-
weight:700">PasswordLab</div>
<div style="font-size:11px;color:rgba(255,255,255,0.35);margin-
top:2px">Генератор паролей</div>
</div>
</div>
<div class="nav-links">
<a href="/">Проверка пароля</a>
<a href="/generator" class="active">Генератор</a>
</div>
</div>
</div>

<!-- MAIN WRAP -->
<div class="wrap">
<section class="card">
<h2>Генератор надёжных паролей</h2>

<div class="input-group">
<input id="length" class="input" type="number" value="12" min="4"
max="64" aria-label="Длина пароля"/>
</div>
<div class="checkbox-group">
<label><input type="checkbox" id="lower" checked> Строчные буквы (a-
z)</label>
<label><input type="checkbox" id="upper" checked> Заглавные буквы
(A-Z)</label>
<label><input type="checkbox" id="digits" checked> Цифры (0-
9)</label>
<label><input type="checkbox" id="symbols"> Символы (!@#$...)</label>
</div>
<div class="result-box" id="result">Сгенерированный пароль появится
здесь</div>
<div class="buttons-group">
<button id="generate-btn"
onclick="generate()">Сгенерировать</button>
<button id="copy-btn" onclick="copyPassword()">Скопировать</button>
<button id="use-btn" onclick="useInCheck()">Использовать</button>
</div>
</section>

</div>
<script>
window.onload = generate;
function getCookie(name) {
let cookieValue = null;
if (document.cookie && document.cookie !== "") {

```

```

    const cookies = document.cookie.split(";");
    for (let cookie of cookies) {
        cookie = cookie.trim();
        if (cookie.startsWith(name + "=")) {
            cookieValue =
decodeURIComponent(cookie.substring(name.length + 1));
            break;
        }
    }
}
return cookieValue;
}
function generate(){
    const length = parseInt(document.getElementById("length").value);
    const lower = document.getElementById("lower").checked?1:0;
    const upper = document.getElementById("upper").checked?1:0;
    const digits = document.getElementById("digits").checked?1:0;
    const symbols = document.getElementById("symbols").checked?1:0;
    if(!lower && !upper && !digits && !symbols){
        alert("Выберите хотя бы один набор символов");
return;
    }
    fetch("/generate/", {
        method: "POST",
        headers: {
            "Content-Type": "application/json",
            "X-CSRFToken": getCookie("csrftoken")
        },
        body: JSON.stringify({
            length: length,
            lower: lower,
            upper: upper,
            digits: digits,
            symbols: symbols
        })
    })
    .then(r=>r.json())
    .then(data=>{
        const box=document.getElementById("result");
        const pwd=data.password;
        box.textContent="";
        box.classList.add("highlight");
        let i=0;
        const interval=setInterval(()=>{
            if(i<pwd.length){ box.textContent+=pwd[i++]; }
            else{ clearInterval(interval);
setTimeout(()=>box.classList.remove("highlight"),600); }
            },35);
    });
}
function copyPassword(){
    const pwd=document.getElementById("result").textContent;
    const btn=document.getElementById("copy-btn");

```

```

    setTimeout(()=>btn.textContent="Скопировать",1200);
    btn.textContent="✓ Скопировано";
    if(!pwd||pwd.includes("появится")) return;
    navigator.clipboard.writeText(pwd).then(()=>btn);
}
function useInCheck(){
    const pwd = document.getElementById("result").textContent;
    if(!pwd || pwd.includes("появится")) return;
    // сохраняем безопасно
    sessionStorage.setItem("generated_password", pwd);
    // переходим без пароля в URL
    window.location.href = "/";
}
</script>
</body>
</html>

```

Приложение В

(обязательное)

Веб-приложение определения стойкости пользовательских паролей

Презентация

					ПГУ 1.090301.13.001 ПЗ	Лист
						100
Изм.	Лист	№докум.	Подпись	Дата		

ВЕБ-ПРИЛОЖЕНИЕ ДЛЯ ОПРЕДЕЛЕНИЯ СТОЙКОСТИ ПОЛЬЗОВАТЕЛЬСКИХ ПАРОЛЕЙ

АВТОР РАБОТЫ:

СТУДЕНТ ГР. 22ВВВ1 УТКИН МИХАИЛ МАКСИМОВИЧ

НАУЧНЫЙ РУКОВОДИТЕЛЬ:

К.Т.Н. КАФ. ВТ ТОКАРЕВ АНДРЕЙ НИКОЛАЕВИЧ

АКТУАЛЬНОСТЬ ТЕМЫ

- Слабые пароли остаются одной из основных причин утечек данных
- Пользователи часто используют простые и предсказуемые комбинации
- Современные методы подбора паролей постоянно совершенствуются
- Возникает необходимость в инструментах комплексной оценки стойкости паролей

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Цель:

Разработка веб-приложения для комплексного анализа стойкости пользовательских паролей

Основные задачи:

- Исследование методов оценки паролей;
- Реализация анализа структуры;
- Вычисление энтропии;
- Проверка по базе утечек;
- Разработка ML-модуля;
- Создание веб-интерфейса;
- Тестирование приложения.

Изм.	Лист	№докум.	Подпись	Дата

ПГУ 1.090301.13.001 ПЗ

Лист

102

ПРЕИМУЩЕСТВА РАЗРАБОТАННОГО ПРИЛОЖЕНИЯ

- Комплексный анализ паролей
- Проверка структуры и шаблонов
- Проверка по базе утечек
- Использование машинного обучения
- Генерация надежных паролей
- Веб-интерфейс
- Динамическое обновление результатов

ПЕРЕД НАЧАЛОМ РАЗРАБОТКИ БЫЛО НЕОБХОДИМО

- Изучить существующие методы оценки паролей
- Проанализировать угрозы информационной безопасности
- Исследовать существующие сервисы
- Выбрать технологии разработки
- Определить архитектуры системы

АНАЛИЗ АНАЛОГИЧНЫХ РЕШЕНИЙ

Сервис	Энтропия	Утечки	ML	Генератор
2IP	✓	✗	✗	✓
Have I Been Pwned	✗	✓	✗	✗
Разработанное приложение	✓	✓	✓	✓

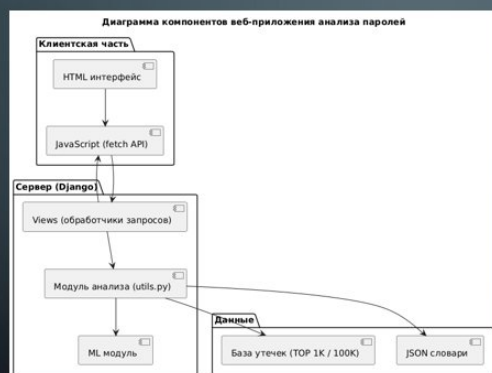
РАЗРАБАТЫВАЕМАЯ СИСТЕМА ДОЛЖНА ОБЕСПЕЧИВАТЬ

- Анализ стойкости пароля
- Вычисление энтропии
- Выявление повторов и последовательностей
- Проверку по локальной базе утечек
- Оценку времени подбора
- Формирование рекомендации пользователю
- Генерацию безопасных паролей

ВЫБОР ТЕХНОЛОГИЙ

Технология	Назначение
Python	Основной язык разработки
Django	Серверная часть
HTML/CSS	Интерфейс
JavaScript	Динамическое обновление
scikit-learn	Машинное обучение
joblib	Загрузка ML-модели

АРХИТЕКТУРА ПРИЛОЖЕНИЯ



Основные компоненты:

- клиентская часть
- Django-приложение;
- Модуль анализа;
- ML-модуль;
- база утечек.

МОДУЛЬ МАШИННОГО ОБУЧЕНИЯ

- Используется алгоритм Logistic Regression
- Библиотека scikit-learn
- Обучение выполнено на выборке из 5000 паролей
- Модель выявляет предсказуемые структуры паролей

ПРИМЕР АНАЛИЗА СЛАБОГО ПАРОЛЯ

The screenshot shows the 'Проверка стойкости пароля' (Password Strength Check) interface. The password 'password123' is entered, and its strength is rated as 'very weak' (very слабый). The interface displays various metrics: Length (11), Number of symbols (26), Entropy (16), and Time to guess (10 seconds). It also shows a 'Симуляция атаки' (Attack Simulation) section with a 'Суперкомпьютер' (Supercomputer) icon and a 'Запустить атаку' (Start Attack) button. A 'Дополнительный анализ' (Additional Analysis) section highlights that the password is common and contains predictable patterns. A 'Рекомендации' (Recommendations) section suggests using a longer password, avoiding common words, and using a mix of symbols and numbers.

ПРИМЕР АНАЛИЗА НАДЕЖНОГО ПАРОЛЯ

The screenshot shows the 'Проверка стойкости пароля' (Password Strength Check) interface for a strong password 'G7mK_rYQzKx5vlg'. The password is rated as 'strong' (сильный). The interface displays various metrics: Length (16), Number of symbols (95), Entropy (102.12), and Time to guess (102.12 seconds). The 'Симуляция атаки' (Attack Simulation) section shows a 'Обычный ПК' (Ordinary PC) icon and a 'Запустить атаку' (Start Attack) button. The 'Дополнительный анализ' (Additional Analysis) section states that the password has high strength and does not contain known weak patterns. The 'Рекомендации' (Recommendations) section congratulates the user on a strong and secure password.

ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ

Тип пароля	Результат
Короткий	Низкая стойкость
Из утечек	Обнаружен
Шаблонный	Оценка снижена
Сложный	Высокая стойкость

ДАЛЬНЕЙШЕЕ РАЗВИТИЕ И ОГРАНИЧЕНИЯ

Возможные направления развития:

- Расширение базы утечек;
- Улучшение ML-модели;
- API для интеграции;
- Анализ словарных слов;
- Улучшение интерфейса;
- Адаптация под корпоративные системы.

Текущие ограничения системы:

- Оценка времени подбора является приближенной;
- Модель машинного обучения зависит от обучающей выборки;
- Система ориентирована на массовые сценарии использования;
- Отсутствует интеграция с внешними сервисами утечек.

ЗАКЛЮЧЕНИЕ

- Поставленная цель работы достигнута
- Разработано веб-приложение для анализа стойкости паролей
- Реализован комплексный подход к оценке паролей;
- Система успешно прошла тестирование
- Приложение может использоваться на практике