# АЛЬМАНАХ

## теоретических и прикладных исследований

ежегодный
научно-практический журнал
Физико-технического института

ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
ФТИ

ПРИДНЕСТРОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
им. Т. Г. Шевченко

# АЛЬМАНАХ
## ТЕОРЕТИЧЕСКИХ И ПРИКЛАДНЫХ МЕЖОТРАСЛЕВЫХ ИССЛЕДОВАНИЙ

**ежегодный научно-практический (методический) журнал
Физико-технического института ПГУ им. Т. Г. Шевченко**

**2024 г.**

Тирасполь

Издательство Приднестровского Университета

2024

УДК 001.891:62(051)
ББК А0я52+Ж/О-1я52
А57



**Альманах теоретических и прикладных межотраслевых исследований** : ежегодный научно-практический (методический) журнал Физико-технического института ПГУ им. Т. Г. Шевченко [Электронный ресурс] / ответственный редактор : Ю. А. Столяренко ; ГОУ «Приднестровский государственный университет им. Т. Г. Шевченко». – Тирасполь : Изд-во Приднестр. ун-та, 2024. – 57 с.

Минимальные системные требования: CPU (Intel/AMD) 1,5ГГц/ОЗУ 2ГГб/HDD 450Мб/1024*768/Windows 7 и старше/Internet Explorer 11/Adobe Acrobat Reader 6 и старше

*Данное издание является итогом научно-исследовательской работы коллектива Физико-технического института ГОУ «ПГУ им. Т. Г. Шевченко» за 2023/24 учебный год.*

*В сборнике представлены статьи бакалавров, магистров, профессорско-преподавательского состава института.*

УДК 001.891:62(051)
ББК А0я52+Ж/О-1я52

Рекомендовано Научно-координационным советом ПГУ им. Т. Г. Шевченко

## *СОДЕРЖАНИЕ*

УДК 627.8

# РАСПОЗНАВАНИЕ ОБЪЕКТОВ НА ИЗОБРАЖЕНИИ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИЙ C#

Е. В. Барановская, магистрант
С. В. Помян, доцент

***Аннотация.*** *Исследование методов распознавания текста на изображениях в среде C# охватывает обзор существующих алгоритмов и библиотек OCR для эффективного извлечения текстовой информации из изображений. Анализируются преимущества различных подходов, исследуется оптимизация процесса распознавания и возможности интеграции с другими инструментами в приложениях на C#.*

***Ключевые слова:*** *нейронные сети, генерация изображений, проектирование приложения, анализ требований, обработка изображений.*

*Искусственный интеллект* – это область информатики, посвященная разработке компьютерных систем, способных выполнять задачи, требующие обычно человеческого интеллекта. Она включает в себя создание алгоритмов, программ и моделей, которые позволяют компьютерам «думать» и принимать решения, анализируя данные, обучаясь на опыте и применяя логику.

Одной из задач искусственного интеллекта является распознавание информации с изображений. Распознавание текстовой информации с картинки, или *OCR* (*Optical Character Recognition*) – это процесс автоматического определения и извлечения текста из изображений. Системы *OCR* используют алгоритмы и методы компьютерного зрения для распознавания отдельных символов, слов и даже целых текстовых блоков на изображениях, делая текст доступным для дальнейшей обработки и анализа компьютером.

Извлечение текста с изображения позволяет автоматизировать различные задачи, такие как сканирование документов, обработка информации с фотографий, анализ содержимого изображений и т. д. Это сокращает время на обработку данных и упрощает рабочие процессы. Когда текст извлекается из изображений, его можно использовать для поиска ключевой информации, анализа контента изображений, создания метаданных или индексации данных.

В медицинской сфере *OCR* используется для распознавания текста с медицинских изображений, документов, результатов анализов и историй болезни, что помогает улучшить точность и скорость работы врачей и медицинского персонала.

В автомобильной отрасли *OCR* применяется для распознавания текста на номерных знаках, сигнальных табличках и документах, что улучшает процессы управления и безопасности на дорогах.

В сфере ритейла и торговли *OCR* используется для автоматического распознавания текста на этикетках товаров, сканирования документов, управления запасами и обработки заказов, что оптимизирует процессы и повышает эффективность бизнеса.

В образовательной сфере *OCR* помогает в распознавании текста на учебных материалах, тестах, работах студентов и документах, что улучшает доступность информации и облегчает учебный процесс.

В различных сферах административной работы *OCR* применяется для обработки документов, сканирования и распознавания текста на бумажных носителях, автоматизации рутинных задач и уменьшения временных затрат.

Эти области демонстрируют разнообразие применений *OCR* технологий и их значимость для улучшения процессов, и повышения эффективности в различных отраслях.

Технологии *C#* предлагают широкий спектр инструментов для реализации распознавания текста на изображениях. Этот процесс обычно включает в себя предварительную обработку изображения (например, улучшение контраста или выравнивание), а затем применение алгоритмов распознавания текста для извлечения нужной информации. Результатом распознавания является текстовая строка, которую можно использовать в дальнейшем для обработки данных или интеграции в другие системы. Распознавание текста с изображений имеет широкий спектр применений.

Для исследования были выбраны пять наиболее подходящих объектов исследования: *Tesseract OCR, Microsoft OCR (MODI), Google Cloud Vision AP, Tessnet2, IronOCR*.

*Tesseract OCR* (*Optical Character Recognition*) – это свободно распространяемый движок распознавания текста, разработанный *Google*. Он способен распознавать текст на изображениях и конвертировать его в машинно-читаемый формат. Возможности *Tesseract* включают поддержку различных языков, многоуровневую обработку изображений для улучшения качества распознавания и гибкую настройку параметров для оптимальных результатов. *Tesseract* широко используется в различных областях, включая образование, медицину, бизнес и научные исследования, благодаря своей открытой

природе и возможности интеграции с различными платформами и языками программирования.

*Microsoft OCR* (*Optical Character Recognition*), ранее известный как *Microsoft Office Document Imaging* (*MODI*), представляет собой технологию распознавания текста, разработанную компанией *Microsoft*. Эта технология позволяет извлекать текст из изображений и документов, таких как сканы или фотографии. *MODI* был включен в состав пакета *Microsoft Office* до версии 2007 и предоставлял возможности *OCR* непосредственно внутри приложений Office, таких как *Word* или *Excel*.

Однако после выпуска *Microsoft Office* 2007, *MODI* был удален из пакета *Office*, и *Microsoft* начала рекомендовать использование других инструментов и технологий для *OCR*, таких как *Microsoft Cognitive Services OCR API* или *Windows OCR* (поставляемый с *Windows* 10).

Технология *Microsoft OCR (MODI)* обладала возможностями распознавания текста с достаточно хорошей точностью и предоставляла некоторые дополнительные функции, такие как возможность создания и редактирования *TIFF*-файлов с текстом. Однако в последние годы Microsoft активно развивает более современные и гибкие решения для обработки изображений и текста, что делает использование более новых инструментов, таких как *Cognitive Services OCR API*, более предпочтительным в контексте современных приложений и сервисов.

*Google Cloud Vision API* – это сервис машинного зрения, предоставляемый *Google Cloud Platform*. Он предоставляет различные возможности для анализа и обработки изображений с использованием передовых технологий машинного обучения и компьютерного зрения.

Основные возможности *Google Cloud Vision API* включают: сервис может автоматически распознавать текст на изображениях и конвертировать его в машинно-читаемый формат, *API* позволяет анализировать содержимое изображений, такое как объекты, лица, метки, настроение и другие аспекты, сервис может распознавать логотипы и марки на изображениях, *API* позволяет фильтровать изображения по различным критериям, таким как безопасность контента или типы объектов, *Google Cloud Vision API* также поддерживает распознавание рукописного текста на изображениях.

Этот сервис широко используется в различных областях, таких как разработка приложений с машинным зрением, анализ изображений в медицине, автоматизация бизнес-процессов и в других сферах, где требуется обработка изображений и анализ их содержимого.

*Tessnet2* – это библиотека распознавания текста (*OCR*), которая является оберткой для *Tesseract OCR Engine*. Она предоставляет возможность использования мощного движка распознавания текста *Tesseract* в приложениях, разработанных на платформе *.NET*.

Библиотека позволяет распознавать текст на изображениях, включая сканы, фотографии и другие графические файлы.

*Tesseract OCR,* на котором основана *Tessnet2*, поддерживает множество языков и позволяет выбирать язык распознавания в зависимости от требований приложения.

Библиотека предоставляет различные параметры и настройки для оптимизации и улучшения процесса распознавания текста.

*Tessnet2* является оберткой для *Tesseract*, специально адаптированной для использования в приложениях, написанных на платформе *.NET*, что обеспечивает удобство и гибкость в интеграции.

*Tesseract OCR Engine*, используемый *Tessnet2*, известен своей высокой точностью распознавания текста, особенно при работе с качественными изображениями.

*Tessnet2* позволяет разработчикам создавать приложения с возможностью распознавания текста на изображениях, что находит применение в различных областях, включая медицину, финансы, образование, а также автоматизацию бизнес-процессов, где требуется обработка текста с графических файлов.

Для дальнейшего экспериментального анализа все выше перечисленные модели ИИ были проанализированы по следующим пунктам: поддержка мультиязычности, лицензированность (бесплатные версии), сообщество и поддержка.

Подробные результаты анализа по выбранным объектам исследования представлены на рисунке 1:

| № п/п | Наименование | Поддержка мультиязычности | Бесплатность/ Лицензирование | Сообщество и поддержка | Выбраны для исследования |
|---|---|---|---|---|---|
| 1. | Tesseract OCR | Да | Бесплатно (Apache License 2.0) | Активное сообщество на GitHub, Google Developers | Да |
| 2. | Microsoft OCR (MODI) | Да | Включено в Microsoft Office (лицензия MOSS) | Поддержка ограничена, так как Microsoft объявила о прекращении разработки MODI в 2009 году | Нет |
| 3. | Google Cloud Vision AP | Да | Платно | Полная поддержка от Google Cloud, активное обновление API | Нет |
| 4. | Tessnet2 | Да | Бесплатно (MIT License) | Ограниченная поддержка, но активное сообщество на GitHub | Нет |
| 5. | IronOCR | Да | Бесплатно | Профессиональная поддержка от разработчика, активное обновление | Да |

Рис. 1. Результаты сравнительного анализа

Таким образом для дальнейшего экспериментального анализа были выбраны: *Tesseract OCR, IronOCR.*

Пункты для экспериментального сравнительного анализа: сложность интеграции, проверка мультиязычности, распознавание рукописного текста, распознавание документов формата *PDF*, распознавание чисел и цифр.

Интеграция *Tesseract OCR* в *C#* может быть оценена как средняя сложность, требующая базового понимания работы с библиотеками и *API*, а также опыта работы с изображениями и текстовыми данными.

Интеграция *IronOCR* в *C#* оценивается как «низкая сложность», особенно для опытных разработчиков, благодаря простоте использования *API* и документации библиотеки.

*Tesseract OCR* поддерживает более 100 языков и алфавитов, позволяя выбирать язык распознавания и использовать соответствующие языковые модели и словари для улучшения качества распознавания. Это делает *Tesseract* мощным инструментом для работы с текстом на различных языках, включая языки с различными алфавитами и особенностями.

*IronOCR* обладает мощными возможностями мультиязычности, поддерживая более 120 языков и алфавитов. Это включает различные языки с разными алфавитами, системами письма и языковыми особенностями. Вы можете легко выбирать язык распознавания, что позволяет получать точные результаты для текстов на разных языках, и использовать различные языковые модели и словари для улучшения качества распознавания.

Итоги рукописного распознавания были неудовлетворительны для обеих моделей, тестовые данные были идентичны, а также обработаны для более качественного распознавания.

*IronOCR* максимально точно справилась с распознаванием текстовой информации с документов формата *PDF*. *Tesseract OCR* не имел инструментов для распознавания документов формата *PDF*, официальная документация предлагает пользователем сначала сконвертировать документ в изображения, а потом провести процесс распознавания.

С машинным текстом в виде цифр и чисел обе модели справились успешно, с рукописными данными (обработанного изображения: удаление лишнего шума, настройка контраста изображения) не справились обе модели.

Краткие итоги экспериментального анализа отображены на рисунке 2:

| **Пункты для сравнения** | *Tesseract OCR* | IronOCR |
|---|---|---|
| Сложность интеграции | средняя | низкая |
| Проверка мультизычности | > 100 языков | > 120 языков |
| Распознавание рукописного текста | неудовлетворительно | неудовлетворительно |
| Распознавание документов формата PDF | - | + |
| Распознавание чисел и цифр | Машинный: +<br>Рукописный: - | Машинный: +<br>Рукописный: - |

Рис. 2. Итоги экспериментального анализа

Обе модели, *Tesseract OCR* и *IronOCR*, являются мощными инструментами для распознавания текста на изображениях в *C#.*

*Tesseract OCR* имеет широкую поддержку языков (более 100), что делает его надежным выбором для мультиязычных приложений. Он предоставляет высокую точность распознавания и позволяет настраивать параметры для оптимальных результатов.

*IronOCR*, в свою очередь, предлагает еще больше языковой поддержки (более 120), что делает его более обширным в выборе языков распознавания текста. Он также обладает простотой интеграции и удобным *API* для работы с текстом на изображениях.

В целом, обе модели имеют свои преимущества в зависимости от конкретных требований проекта. *Tesseract OCR* подходит для широкого спектра языков, а *IronOCR* расширяет возможности мультиязычности и обладает удобным интерфейсом для работы с текстом на изображениях, поэтому предпочтение мы бы отдали именно ей.

В области распознавания текстовой информации с изображений с использованием технологий *C#* представлено разнообразие инструментов и библиотек, способных эффективно обрабатывать изображения и извлекать текстовую информацию. Интеграция и использование таких технологий становятся все более актуальными в свете растущей потребности в обработке и анализе визуальных данных.

Выбор подходящего инструмента зависит от конкретных требований проекта, таких как необходимость мультиязычной поддержки, точности распознавания, производительности и бюджета.

Исходя из преимуществ и недостатков каждого инструмента, разработчики могут принимать обоснованные решения, стремясь к оптимальному сочетанию функциональности, производительности и поддержки в своих проектах по распознаванию текста на изображениях.

**Цитированная литература**

1. *Singh, P., & Gautam, A. Object Detection and Recognition Techniques: A Review. International Journal of Advanced Science and Technology,* 30 (11). – 2021. – P. 3686–3698.

2. *Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* – 2016. – P. 779–788.

3. Попов, А. В. Обзор алгоритмов распознавания объектов на изображениях / А. В. Попов, А. А. Шептулин // Вестник Московского государственного технического университета имени Н. Э. Баумана. Серия: Информатика и системы управления, (1). – 2021. – С. 34–54.

4. Обнаружение объектов с помощью *ONNX* в *ML.NET.* – URL : https://learn.microsoft.com/ru-ru/dotnet/machine-learning/tutorials/object-detection-onnx.

5. Распознавание объектов на фото: принцип работы и способ реализации. – URL : https://codernet.ru/articles/drugoe/raspoznavanie_obektov_na_foto_princzip_rabotyi_i_sposob_realizaczii/#:~:text=Распознавание%20объектов%20на%20фото%20—%20это,базы%20знаний%2C%20программирование%20и%20мн.

УДК 004.03

# ВЗАИМОДЕЙСТВИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ «ОБНАРУЖЕНИЕ ПНЕВМОНИИ С ИСПОЛЬЗОВАНИЕМ РЕНТГЕНОВСКИХ ИЗОБРАЖЕНИЙ ГРУДНОЙ КЛЕТКИ» В ГОРОДСКОЙ БОЛЬНИЦЕ

Н. И. Вороненко, магистрант
А. М. Башкатов, доцент

***Аннотация.*** *Эта статья исследует взаимодействие программного обеспечения, разработанного на основе языка программирования Python, с рентгенологами в городской больнице для обнаружения пневмонии на рентгеновских изображениях. Мы рассматриваем значение автоматизированных решений в диагностике пневмонии, а также роль Python в создании алгоритмов глубокого обучения для быстрого и точного выявления признаков заболевания. Преимущества взаимодействия включают сокращение времени диагностики, повышение точности обнаружения, оптимизацию медицинских ресурсов и поддержку рентгенологов в принятии окончательных решений. Статья также обсуждает роль рентгенологов в этом взаимодействии и подчеркивает перспективы будущего в области диагностики пневмонии через интеграцию программного обеспечения и технологий искусственного интеллекта.*

***Ключевые слова:*** *программирование, рентген, машинное обучение, системы поддержки принятия решений.*

Рентгенология вместе с прогрессом технологий входит в новую эру с использованием программного обеспечения для обнаружения пневмонии на рентгеновских изображениях. Это революционное взаимодействие технологии и здравоохранения открывает новые горизонты для диагностики и лечения заболеваний легких. В этой статье мы рассмотрим, как программное обеспечение на основе *Python* активно взаимодействует с рентгенологами в городской больнице для эффективного обнаружения пневмонии.

## Значение программного обеспечения в диагностике пневмонии

Рентгеновские изображения являются важным инструментом для диагностики пневмонии, но их интерпретация требует опыта и времени. В городской больнице, где поток пациентов постоянно растет, автоматизированные решения становятся необходимостью. Программное обеспечение, основанное на алгоритмах машинного обучения, обеспечивает быстрое и точное обнаружение изменений на рентгеновских снимках, что существенно сокращает время диагностики и повышает эффективность лечения [1].

## Работа программного обеспечения на основе *Python*

*Python*, как мощный и универсальный язык программирования, становится основой для разработки программного обеспечения в здравоохранении. Алгоритмы глубокого обучения, реализованные на *Python*, позволяют автоматически обнаружить признаки пневмонии на рентгеновских снимках. Интеграция этих алгоритмов в повседневную практику рентгенологов позволяет им более точно и быстро определять наличие пневмонии, что, в свою очередь, способствует оперативному лечению пациентов.

Одним из наиболее перспективных подходов к созданию такого программного обеспечения является использование методов глубокого обучения, особенно сверточных нейронных сетей (*CNN*). *CNN* – это тип искусственных нейронных сетей, которые способны извлекать и обрабатывать признаки из изображений, используя операции свертки и пулинга. *CNN* имеют множество преимуществ перед традиционными методами машинного обучения, такими как:

– способность к самостоятельному обучению признаков из изображений, без необходимости ручной подготовки или выбора признаков;

– способность к обработке изображений высокого разрешения, сохраняя пространственную информацию и иерархическую структуру признаков;

– способность к обобщению и адаптации к новым данным, благодаря использованию больших объемов обучающих данных и регуляризации.

В настоящее время существует множество архитектур *CNN*, которые были разработаны и применены для различных задач компьютерного зрения, таких как классификация, сегментация, детектирование или распознавание объектов. Некоторые из наиболее известных и эффективных архитектур *CNN* включают в себя: *AlexNet, VGG, ResNet, DenseNet* [3].

Для решения задачи обнаружения пневмонии с использованием рентгеновских изображений грудной клетки на *Python* можно использовать любую из перечисленных архитектур *CNN*, либо их модификации или комбинации.

### Преимущества взаимодействия с программным обеспечением

1. Сокращение времени диагностики. Программное обеспечение значительно снижает время, необходимое для обработки рентгеновских снимков и выявления признаков пневмонии. Это особенно критично в условиях городской больницы, где число пациентов высоко [2].

2. Повышение точности. Алгоритмы машинного обучения, используемые в программном обеспечении, обучаются на больших наборах данных, что способствует высокой точности обнаружения. Это помогает избежать ошибок, связанных с человеческим фактором.

3. Оптимизация ресурсов. Своевременное обнаружение и лечение пневмонии сокращает стоимость лечения и использование медицинских ресурсов, так как предотвращает развитие болезни и обеспечивает более эффективное лечение.

### Роль рентгенологов во взаимодействии

Важно отметить, что программное обеспечение не заменяет роль рентгенологов, а скорее поддерживает их в их работе. Рентгенологи остаются ключевыми врачами, осуществляющими окончательную диагностику и разрабатывающими план лечения на основе данных, предоставленных программным обеспечением [4].

### Будущие взаимодействия

Взаимодействие программного обеспечения для обнаружения пневмонии на рентгеновских изображениях с рентгенологами представляет собой лишь начало. С постоянным развитием технологий и методов искусственного интеллекта эта связь будет укрепляться, предоставляя ещё более точные и быстрые методы диагностики.

В заключение, взаимодействие программного обеспечения с рентгенологами в городской больнице при обнаружении пневмонии привносит новые возможности в сферу здравоохранения. Это сочетание технологии и медицинского опыта обещает повышение эффективности диагностики и лечения, что в конечном итоге ведет к улучшению заботы о пациентах и сокращению затрат на здравоохранение.

### Цитированная литература

1. *Rajpurkar, P., Irvin, J., Zhu, K., Yang, B., Mehta, H., Duan, T., ... & Ng, A.* (2017). *CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning. arXiv preprint arXiv*:1711.05225.

2. *Oakden-Rayner, L.* (2017). *Exploring the ChestXray14 dataset: problems. arXiv preprint arXiv:*1712.06983.

3. *Anthimopoulos, M., Christodoulidis, S., Ebner, L., Christe, A., & Mougiakakou, S. (2016). Lung pattern classification for interstitial lung diseases using a deep convolutional neural network. IEEE Transactions on Medical Imaging,* 35(5), 1207-1216.

4. *Wang, X., Peng, Y., Lu, L., Lu, Z., Bagheri, M., & Summers, R. M. (2017). Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2097–2106).

УДК 004.8

# АНАЛИЗ ТРЕБОВАНИЙ И ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ ДЛЯ ГЕНЕРАЦИИ ИЗОБРАЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ НЕЙРОННЫХ СЕТЕЙ

А. И. Голубов, магистрант
Ю. А. Столяренко, доцент

***Аннотация.*** *Проведен анализ требований проектирования приложения, использующего нейронные сети для генерации изображений. Выполнен анализ и оценка алгоритма генерации изображений с использованием глубокого обучения. Рассмотрены различные аспекты процесса разработки, от теоретических основ до практических аспектов, включая подготовку данных, оптимизацию алгоритмов и тестирование.*

***Ключевые слова:*** *машинное обучение, нейронные сети, генерация изображений, проектирование приложения, анализ требований, обработка изображений.*

В последние годы область искусственного интеллекта достигла значительных успехов в разработке нейронных сетей, способных генерировать изображения. Эти сети, часто называемые генеративными моделями, могут создавать реалистичные и разнообразные изображения, которые трудно отличить от реальных фотографий.

Целью данной работы является разработка приложения, которое использует нейронные сети для генерации изображений. В рамках работы будет исследован и реализован алгоритм генерации изображений с использованием глубокого обучения.

Программный продукт будет разработан на языке программирования *Python*, который широко используется в области машинного обучения и глубокого обучения. Будут использованы современные библиотеки и инструменты, такие как *TensorFlow* и *PyTorch*, которые предоставляют функциональность для создания и обучения нейронных сетей.

В процессе разработки будет уделено внимание выбору и подготовке обучающих данных. Обучение нейронной сети для генерации изображений требует большого объема данных, поэтому будет необходимо подобрать или создать подходящий набор данных для обучения модели.

Также будет проведена оптимизация алгоритмов для достижения высокого качества генерируемых изображений. Это может включать в себя изменение архитектуры нейронной сети, выбор подходящих функций потерь или применение методов, таких как генеративные состязательные сети, чтобы улучшить качество результатов.

В процессе разработки будет проведено тестирование программного продукта для проверки его работоспособности и качества генерации изображений. Будут использованы метрики оценки качества изображений, такие как структурное сходство или перплексия, чтобы оценить результаты работы приложения.

Одним из популярных подходов к созданию изображений с помощью нейронных сетей является использование генеративных-состязательных сетей.

*Генеративно-состязательные сети (GAN)* – это тип архитектуры нейронных сетей, предназначенных для обучения без контроля. Они были впервые представлены Яном Гудфеллоу и его коллегами в 2014 году и с тех пор стали популярным подходом для генерации новых данных, похожих на заданный набор данных.

*GAN* состоят из двух нейронных сетей: *сети-генератора,* которая создает изображения, и *сети-дискриминатора,* которая пытается отличить созданные изображения от реальных. Эти две сети обучаются вместе в процессе, в котором генератор пытается обмануть дискриминатор и заставить его думать, что созданные им изображения реальны, а дискриминатор пытается правильно определить, какие изображения созданы, а какие реальны.

Разработка приложения для генерации изображений с помощью *GAN* включает в себя несколько этапов. Во-первых, необходимо получить и предварительно обработать подходящий набор данных изображений. Этот набор данных будет использоваться для обучения *GAN* и должен содержать большое количество разнообразных изображений, представляющих тип изображений, которые должно генерировать приложение. Далее необходимо разработать архитектуру сетей генератора и дискриминатора. Это включает в себя выбор количества и размера слоев, а также типа функций активации и других гиперпараметров. Архитектура должна быть выбрана таким образом, чтобы сбалансировать способность генератора создавать разнообразные и реалистичные изображения с доступными вычислительными ресурсами.

После разработки архитектуры, *GAN* может быть обучена на наборе данных. В процессе обучения веса генератора и дискриминатора обновляются итеративно, используя обратное распространение ошибки и градиентный спуск. Процесс обучения может занять значительное количество времени и

вычислительных ресурсов, в зависимости от размера и сложности набора данных и архитектуры сети.

По окончании обучения, генеративная сеть может быть применена для генерации новых изображений. Для этого можно подать случайный шум на вход генератора и использовать его для создания изображения. Затем сгенерированное изображение можно подвергнуть постобработке и вывести на экран или сохранить для последующего использования.

**Постобработка.** После создания новых изображений с использованием обученной *GAN* часто используются методы постобработки для улучшения визуального качества сгенерированных изображений. Такие методы могут включать в себя коррекцию цвета, улучшение резкости и снижение уровня шума.

Цветокоррекция может использоваться для корректировки цветов генерируемых изображений, чтобы сделать их более визуально привлекательными или привести в соответствие с распределением цветов в обучающем наборе данных. Для этого могут применяться методы, такие как согласование гистограмм или передача цвета.

Использование методов повышения резкости может улучшить детализацию сгенерированных изображений и придать им большую четкость. Для этого могут быть применены техники, такие как нерезкое маскирование или фильтрация высоких частот.

Применение методов шумоподавления позволяет удалить артефакты и шум, присутствующие на сгенерированных изображениях. Для этой цели могут быть использованы такие методы, как медианная фильтрация или денойзинг с использованием нелокальных средств.

Помимо этих методов, для улучшения визуального качества созданных изображений могут применяться и другие методы постобработки, такие как усиление контраста, гамма-коррекция и тональное отображение.

Важно отметить, что постобработку следует применять разумно и осторожно, поскольку чрезмерная постобработка может внести артефакты или искажения в созданные изображения. Целью постобработки должно быть улучшение визуального качества сгенерированных изображений при сохранении их реалистичности и разнообразия.

**Другие генеративные модели**. Помимо генеративно-состязательных сетей, существует несколько других типов генеративных моделей, которые могут быть использованы для создания новых данных, сходных с заданным набором данных. Некоторые распространенные типы генеративных моделей включают вариационные автоэнкодеры, модели авторегрессии и модели нормализующего потока.

Вариационные автоэнкодеры представляют собой тип генеративной модели, где используется кодирующая сеть для преобразования входных данных в скрытое пространство и декодирующая сеть для восстановления точек из скрытого пространства обратно в пространство данных. Сети кодировщика и декодировщика обучаются совместно, с целью максимизировать вероятность данных с учетом скрытых переменных. После обучения сеть декодировщика может быть использована для генерации новых данных путем выборки точек из латентного пространства и преобразования их обратно в пространство данных.

Авторегрессионные модели представляют собой класс генеративных моделей, которые создают новые данные по одному элементу за раз, учитывая предыдущие сгенерированные элементы. Эти модели применяются для генерации последовательностей данных, таких как текст или временные ряды. Рекуррентные нейронные сети и трансформеры являются распространенными типами авторегрессионных моделей.

Модели нормализующего потока – это тип генеративной модели, которая использует серию инвертируемых преобразований для преобразования данных из простого распределения (например, гауссова распределения) в более сложное распределение, соответствующее распределению данных. Эти модели могут быть обучены путем максимизации правдоподобия данных под преобразованным распределением. После обучения новые данные могут быть получены путем выборки из простого распределения и применения обратных преобразований.

**Алгоритмы оптимизации.** В процессе обучения генеративной и дискриминаторной сетей важную роль играют алгоритмы оптимизации, которые отвечают за обновление и корректировку весов этих сетей. Выбор оптимального алгоритма оптимизации имеет значительное значение для обеспечения стабильности и сходимости процесса обучения.

Алгоритмы оптимизации предназначены для нахождения оптимальных значений весов моделей, которые минимизируют заданную функцию потерь или максимизируют целевую функцию. Они основываются на различных математических и статистических методах, и каждый алгоритм имеет свои особенности и преимущества. Некоторые распространенные алгоритмы оптимизации, используемые для обучения *GAN*, включают стохастический градиентный спуск, *Adam* и *RMSprop*. Эти алгоритмы используют различные подходы для оценки градиента функции потерь относительно весов сети и соответствующего обновления весов.

*Стохастический градиентный спуск (SGD)* – это простой алгоритм оптимизации, который обновляет веса сетей в направлении отрицательного градиента функции потерь. *SGD* может быть чувствителен к

выбору скорости обучения и может потребовать тщательной настройки для достижения хорошей производительности.

*Adam* представляет собой алгоритм адаптивной оптимизации, который использует оценки первого и второго моментов градиентов для обновления весов модели. Этот алгоритм показал хорошие результаты при обучении генеративно-состязательных сетей и широко используется в качестве алгоритма оптимизации по умолчанию.

*RMSprop* является еще одним адаптивным алгоритмом оптимизации, который использует оценку второго момента градиентов для обновления весов модели. Этот алгоритм также продемонстрировал хорошую эффективность в некоторых случаях обучения генеративно-состязательных сетей.

Кроме упомянутых алгоритмов оптимизации, для обучения генеративно-состязательных сетей могут быть использованы и другие методы, включая *Adagrad*, *Adadelta* и *Nesterov Accelerated Gradient* (*NAG*). Выбор конкретного алгоритма оптимизации зависит от различных факторов, таких как архитектура сетей, размер набора данных и особенности самих данных.

**Анализ требований к приложению.** Для создания приложения для генерации изображений с помощью нейронных сетей необходимо тщательно определить требования, которым оно должно соответствовать. К таким *требованиям* относятся функциональные требования, например, возможность создавать качественные и разнообразные изображения, а также нефункциональные требования, например, скорость работы, простота использования и возможность масштабирования.

Одним из основных требований к приложению для генерации изображений является способность создавать высококачественные, реалистичные и визуально привлекательные изображения. Для достижения этой цели может быть использованы передовые архитектуры нейронных сетей и методы обучения, которые обеспечивают высокий уровень качества генерируемых изображений.

Также важным требованием является способность генерировать разнообразные изображения, представляющие широкий спектр стилей и тем. Для этого могут использоваться большие и разнообразные наборы обучающих данных, а также методы контроля разнообразия генерируемых изображений.

В дополнение к этим функциональным требованиям существует также несколько нефункциональных требований, которые необходимо учитывать при разработке приложения для генерации изображений. Они могут включать требования к производительности, например, способность быстро и эффективно генерировать изображения, а также требования к удобству использования, например, необходимость интуитивно понятного и дружественного интерфейса.

Другим важным аспектом является масштабируемость приложения, поскольку оно должно быть способно обрабатывать все большие объемы данных и растущее количество пользователей. Для обеспечения эффективного масштабирования могут использоваться облачные вычислительные ресурсы и другие методы, которые позволяют расширять возможности приложения в соответствии с его популярностью и потребностями.

Помимо функциональных и нефункциональных требований, рассмотренных выше, существует еще несколько факторов, которые необходимо учитывать при разработке приложения для генерации изображений с помощью нейронных сетей. К ним относятся потребности пользователя, технические ограничения и другие соображения, которые могут повлиять на дизайн приложения.

Понимание потребностей пользователей является ключевым аспектом при создании любого приложения, и это особенно актуально для приложений, связанных с генерацией изображений. Важно глубоко осознать потребности и ожидания целевой аудитории, чтобы создать приложение, которое не только отвечает их требованиям, но и удобно в обращении. Для этого можно провести исследования среди потенциальных пользователей и собирать от них отзывы что поможет сделать процесс разработки более обоснованным.

Технические ограничения – еще один важный фактор, который необходимо учитывать при разработке приложения. Они могут включать ограничения на доступные вычислительные ресурсы, такие как вычислительная мощность и память, а также ограничения на размер и качество обучающих данных. Эти ограничения должны учитываться при разработке архитектуры нейронной сети и выборе соответствующих методов обучения.

Другие факторы, которые могут повлиять на разработку приложения, включают нормативные требования, такие как правила конфиденциальности и безопасности данных, а также этические соображения, например, необходимость избегать создания изображений, которые могут быть вредными или оскорбительными. Эти факторы должны быть тщательно рассмотрены и учтены, чтобы обеспечить ответственный и этичный подход к разработке приложения.

**Архитектура и дизайн приложения.** Предлагаемая архитектура и дизайн приложения для генерации изображений с помощью нейронных сетей будут зависеть от множества факторов, включая требования и ограничения, рассмотренные в предыдущем разделе. Однако ниже описан один из возможных подходов к проектированию такого приложения с использованием *Python* в качестве языка разработки.

Приложение может быть разработано как модульная система с отдельными компонентами для предварительной обработки данных, обучения нейронной сети и генерации изображений. Данные компоненты могут быть реализованы с использованием популярных библиотек *Python*, таких как *TensorFlow* или *PyTorch* для построения и обучения нейронных сетей и *OpenCV* или *Pillow* для обработки изображений.

Компонент предварительной обработки данных отвечает за загрузку и подготовку обучающих данных для использования нейронной сетью. Он включает в себя различные задачи, такие как изменение размера изображений, преобразование их в градации серого или другое цветовое пространство, а также нормализацию значений пикселей.

Компонент обучения нейронной сети играет ключевую роль в создании и обучении генеративной модели с использованием предварительно обработанных обучающих данных. В его задачи входит определение архитектуры сетей генератора и дискриминатора, выбор соответствующих функций потерь и алгоритмов оптимизации, а также итеративное обновление весов сети с использованием обратного распространения.

Компонент генерации изображений отвечает за использование обученной генеративной модели для создания новых изображений. Он включает в себя подачу случайных векторов шума в генеративную сеть и использование ее выходных данных для генерации новых изображений. Затем сгенерированные изображения могут быть подвергнуты дополнительной обработке, такой как цветокоррекция или повышение резкости, для улучшения их визуального качества.

Предложенная архитектура обеспечивает модульный и гибкий подход к разработке приложения для генерации изображений с использованием нейронных сетей и *Python*. Благодаря разделению различных компонентов системы, можно разрабатывать и тестировать каждый компонент независимо, что облегчает итерации при разработке и улучшает производительность приложения.

**Обзор процесса разработки и технологий, используемых для реализации приложения.** Процесс разработки приложения будет зависеть от множества факторов, включая выбор дизайна и требования, рассмотренные в предыдущих разделах. Однако ниже описан один из возможных подходов к реализации такого приложения с использованием *Python* в качестве языка разработки.

Первый шаг в процессе разработки приложения включает в себя конфигурацию адекватной среды разработки, инкорпорирующей все необходимые утилиты и библиотеки для конструирования и обучения нейронных сетей. Это предполагает установку языка программирования *Python* и таких распространенных библиотек для машинного обучения, как *TensorFlow* или *PyTorch*. Кроме того, следует настроить систему контроля версий и другие необходимые инструменты для эффективной разработки.

После создания среды разработки следующим шагом будет реализация различных компонентов приложения, включая компоненты предварительной обработки данных, обучения нейронной сети и генерации изображений. Эти компоненты можно разрабатывать и тестировать независимо друг от друга, используя модульные тесты и другие методы тестирования для обеспечения их правильного функционирования.

После реализации и тестирования отдельных компонентов, их можно интегрировать в комплексное приложение. В этот процесс входит разработка пользовательского интерфейса для приложения, а также реализация любых дополнительных функций или возможностей, необходимых для проекта.

Данный процесс разработки обеспечивает структурированный подход к реализации приложения. Следуя этому процессу, можно разработать высококачественное приложение, которое будет отвечать потребностям пользователей и генерировать реалистичные и разнообразные изображения.

После разработки приложения важно тщательно протестировать и оценить его, чтобы убедиться, что оно соответствует всем требованиям и работает так, как ожидалось. Для этого могут использоваться различные методы тестирования и оценки, включая функциональное тестирование, тестирование производительности и пользовательское тестирование.

Функциональное тестирование направлено на проверку того, выполняет ли приложение все функции, указанные в требованиях. В данном случае функциональное тестирование включает проверку способности приложения генерировать высококачественные и разнообразные изображения. Также важно проверить правильность функционирования всех элементов пользовательского интерфейса и других функциональных возможностей приложения.

Тестирование производительности включает измерение производительности приложения в различных условиях с целью проверить, соответствует ли оно установленным требованиям к производительности. Данное тестирование может включать оценку скорости и эффективности процесса генерации изображений, а также измерение использования ресурсов приложения.

Пользовательское тестирование предполагает сбор отзывов от потенциальных пользователей приложения с целью оценки его удобства и эффективности. Для этого могут быть проведены опросы или интервью с пользователями, чтобы выяснить их мнение о приложении. Также может быть осу-

ществлено наблюдение за пользователями во время их взаимодействия с приложением для выявления любых проблем с удобством использования или областей, требующих улучшения.

Тщательное тестирование и оценка играют важную роль в процессе разработки приложения для генерации изображений с использованием нейронных сетей. Путем проведения всесторонней оценки приложения можно убедиться, что оно полностью соответствует требованиям и работает так, как ожидается. Такой подход гарантирует предоставление пользователям высококачественного и эффективного инструмента для генерации изображений.

Таким образом, в ходе анализа требований и проектирования данного приложения была определена цель и выделены задачи для её решения. Также были определены требования к функциональным возможностям приложения и описан основной алгоритм, применяемый в приложении

На основе результатов, полученных в ходе проделанной работы, возможна реализация приложения для генерации изображений с использованием нейронных сетей.

**Цитированная литература**

1. *O'Connor R. Build Your Own Imagen Text-to-Image Model* [Электронный ресурс]. – 2022. – URL: https://www.assemblyai.com/blog/minimagen-build-your-own-imagen-text-to-image-model/ (дата обращения: 21.03.2024).

2. *Radford A., Kim J. W., Hallacy C. Learning Transferable Visual Models From Natural Language Supervision* [Электронный ресурс]. – 2020. – URL: https://arxiv.org/pdf/2103.00020.pdf (дата обращения: 20.03.2024).

3. *Radich Q., Jenks A. Image Classification with PyTorch* [Электронный ресурс]. – 2021. – URL: https://learn.microsoft.com/en-us/windows/ai/windows-ml/tutorials/pytorch-intro (дата обращения: 22.03.2024).

4. *Brownlee J. A Gentle Introduction to Generative Adversarial Networks (GANs)* [Электронный ресурс]. – 2019. – URL: https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/ (дата обращения: 21.03.2024).

УДК 681.5:004

# УСТРОЙСТВА КОНТРОЛЯ РАЗМЕРОВ ДЕТАЛЕЙ С ПРИМЕНЕНИЕМ ПЛАТЫ ARDUINO

И. В. Николаев, студент
Д. А. Филин, студент
А. А. Зуев, ст. преподаватель

***Аннотация.*** *В статье представлены результаты разработки устройства для контроля размеров с применением платы Arduino. Данная аппаратно-программная платформа используется в образовательных целях и для разработки мехатронных систем и устройств. Ускорение проверки размеров позволяет повысить производительность труда рабочего.*

***Ключевые слова:*** *контроль размеров, Arduino, аппаратно-программная платформа, мехатронные системы и устройства, промышленные роботы, производительность труда.*

Успех приходит к тому, чья работа дает наилучшие результаты, а не к тому, кто просто много работает. На ее результат влияют умение, знание дела и бережливость. Машинное оборудование, избавляющее от тяжелого ручного труда, произвело в производстве настоящую революцию [1].

Машиной называют любое приспособление, служащее для преобразования энергии и совершения полезной работы. Машина состоит из деталей, узлов и агрегатов (механизмов) [2].

Технические средства автоматизации позволяют создать на своей базе системы, выполняющие в автоматическом режиме как технологические производственные операции, так и работы, проводимые в экстремальных условиях и опасные для человека. К последним относятся, например, работы на атомных станциях в условиях радиоактивности, тушение пожаров, работы на высотных конструкциях, на больших глубинах и разминирование.

Технические средства автоматизации производств включают в себя станки и системы ЧПУ, промышленные роботы, системы автоматического контроля, автоматические транспортные системы и автоматизированные складские системы.

Современное производство стремительно развивается в направлении автоматизации с широким использованием компьютерных технологий и робототехнических систем, позволяющих быстро перестраивать технологические процессы на изготовление новых изделий.

Начиная с 1980-х годов одним из направлений повышения эффективности производства стало широкое применение информационных технологий.

Принципиальной особенностью гибкой производственной системы являлось наличие новой компоненты – компьютерной системы управления, обеспечивающей возможность увязки отдельных процессов, функций и задач в единую систему.

Дальнейшее развитие работ в данном направлении в конце 80-х годов привело к формированию компьютеризированного интегрированного производства (КИП). Концепция КИП подразумевает новый подход к организации и управлению производством, особенность которого заключалась не только в применении компьютерных технологий для автоматизации технологических процессов и операций, но в создании интегрированной информационной системы предприятия. Информационная интеграция процессов производства достигалась путем использования общих баз данных, позволяющих более эффективно решать вопросы разработки и проектирования изделий, подготовки производства, планирования и управления производством, материально-технического обеспечения, охватывая все процессы предприятия [3].

*Arduino* – это платформа, состоящая из аппаратно-программных средств для построения систем автоматики и робототехники. Аппаратной основой платформы является плата с размещенным на ней микроконтроллером и разведенными (т. е. распаянными) по фиксированной электрической схеме контактами. Программ пая часть представлена средой разработки *Arduino IDE.*

Платформа обладает несколькими особенностями, делающими ее популярной во всем мире.

Аппаратная часть платформы имеет открытую архитектуру. Архитектура аппаратной части устройства называется открытой, если опубликована ее спецификация, т. е. подробное описание, составляющие, схема и применение каждого компонента. Спецификация позволяет любому производителю создать копию продуктов для платформы, тем самым делая их более доступными. Кроме того, появляется возможность создавать улучшенные, более эффективные версии плат и модулей или новые совместимые устройства. Все это влияет на цены, удерживая их на уровне, доступном широкому кругу пользователей. И наконец, открытая спецификация – это возможность самостоятельного ремонта при должных навыках и умениях [4].

*Arduino,* как компьютерная система управления, обладает широкими возможностями. Она мо-

жет использоваться для управления различными устройствами, такими как моторы, светодиоды, датчики и даже сложные роботы. *Arduino* может считывать данные с датчиков, обрабатывать эти данные и, на основе полученной информации, управлять другими устройствами.

*Arduino* имеет множество вариантов плат, каждая из которых имеет свои уникальные характеристики. Например, *Arduino Uno* – это отличный выбор для начинающих, в то время как *Arduino Mega* предлагает больше возможностей для более сложных проектов.

Важно отметить, что *Arduino* – это не только оборудование, но и сообщество. Существует множество ресурсов, включая онлайн-форумы, где вы можете найти помощь, идеи для проектов и даже код, который вы можете использовать в своих проектах.

*Arduino* – это мощная и гибкая система управления, которая может быть использована в широком диапазоне проектов, от простых до сложных. Благодаря своей открытости и доступности, *Arduino* стала одной из самых популярных платформ для создания прототипов в мире.

Актуальность разработки устройства обусловлена необходимостью ускорению и повышению точности измерений в процессах контроля размеров. Цель проекта – разработать устройство для контроля размеров с использованием платы *Arduino*, которое будет обеспечивать высокую точность измерений и информативность. Задача разработки включала в себя выбор подходящих измерительных инструментов, программирование платы *Arduino* для обработки данных с датчиков и создание пользовательского интерфейса для отображения результатов измерений.

Разработка устройства контроля размеров деталей проходила с использованием метода виртуального моделирования и макетирования. Применение данного устройства позволит оптимизировать процесс контроля. Создания виртуальных прототипов устройства, позволяет провести серию тестов и оптимизаций до его физического производства.

Применение метода моделирования также облегчает процесс идентификации и устранения потенциальных проблем, которые могут возникнуть в процессе эксплуатации устройства.

Использование современных технологий, таких как *Arduino* для управления устройством и обработки данных, может значительно улучшить эффективность устройства контроля размеров деталей. Это может включать в себя автоматизацию процесса измерения и предоставление более точных и повторяемых результатов.

Рассмотрим электрическую структурную устройства контроля размеров деталей с применением платы *Arduino* (рис. 1).

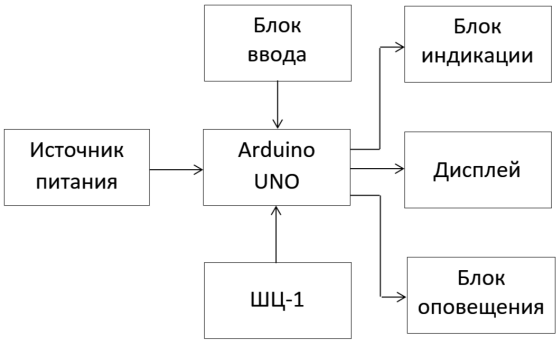


Рис. 1. Электрическая структурная устройства контроля размеров (схема)

Источник питания – это основной источник энергии 5В для всей системы. Он обеспечивает необходимое напряжение и ток для работы всех компонентов.

Основа схемы *Arduino UNO* – это микроконтроллер, который управляет всей системой. Он принимает входные данные от блока ввода, обрабатывает их и передает соответствующие команды другим компонентам системы.

Блок ввода – это блок, включающий в себя две кнопки, которые служат для настройки измерения.

ШЦ-1 – это устройство измерения.

Блок индикации – это блок, отвечающий за отображение информации пользователю и включать в себя светодиод.

Блок оповещения – это блок, отвечающий за звуковое информирования пользователя и включать в себя пьезодинамик.

Дисплей – это устройство вывода, которое отображает информацию в понятном для пользователя формате и отображать текст, графику или другие виды данных.

Все эти блоки вместе работают для обеспечения функционирования устройства. Они взаимодействуют друг с другом для обработки входных данных, выполнения заданных задач и предоставления обратной связи пользователю.

Схема электрической принципиальной устройства раскрывает основные элементы устройства, их электрические связи и функциональные взаимодействия (рис. 2).

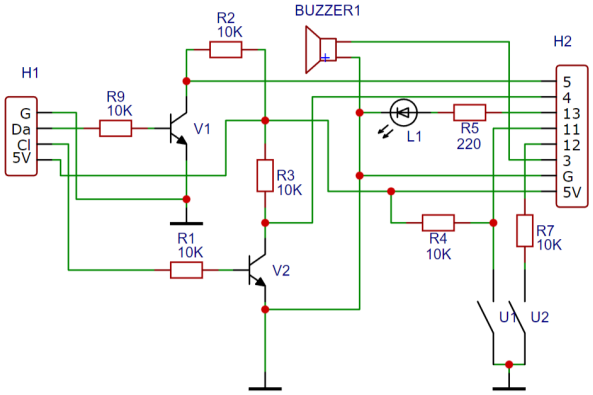


Рис. 2. Электрическая принципиальная устройства контроля размеров (схема)

Эта схема включает в себя резисторы (R1, R2, R3, R4, R5, R7), конденсаторы (C1), индукторы (L1), транзисторы (V1, V2) – для усиления сигнала от *SCL* и *SDA* от штангенциркуля, звуковой сигнал (*BUZZER1*), разъемы (H1) – контакты штангенциркуля и (H2) – контакты *Arduino UNO,* кнопки для настройки (U1, U2) – для настройки максимального и минимального контролируемого размера. *LSD* дисплей подключается к линии *SCL* и *SDA* по *I2C.*

Помимо виртуального тестирования схема собиралась на макетной плате (рис. 3) для тестирования и отладки электронных устройств перед их окончательной сборкой и для демонстрации функциональности или проведения экспериментов.

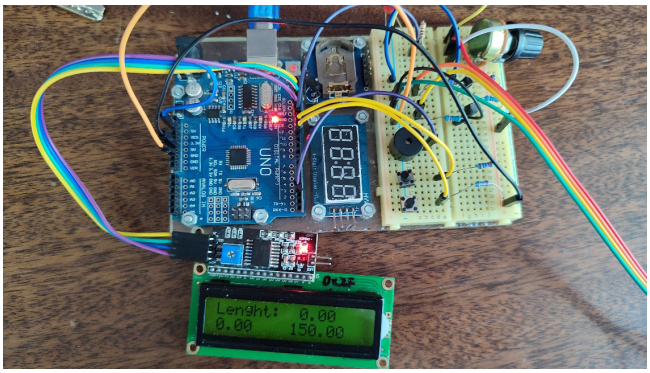


Рис. 3. Схема устройства контроля размеров на макетной плате

Была разработана печатная плата (рис. 4) для этого устройства, которая чертилась в электронный вид с использованием специализированного программного обеспечения для проектирования печатных плат. Затем происходит расстановка компонентов и трассировка путей на плате с учетом оптимального расположения компонентов и минимизации помех.

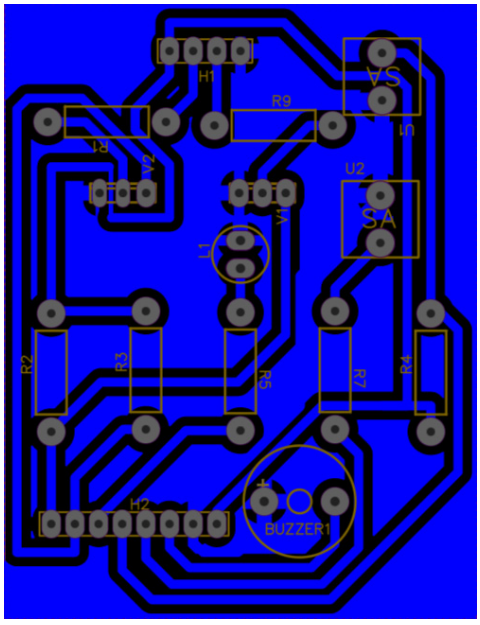


Рис. 4. Печатная плата устройства контроля размеров

Разработка устройства для контроля размеров с использованием платы *Arduino* оказалась успешной и подтвердила заложенные в него требования. *Arduino* позволило создать гибкую и масштабируемую систему, способную обрабатывать данные с датчиков в реальном времени. Данная система имеет потенциал для дальнейшего совершенствования.

Это устройство упрощает контроль размеров детали измерений, что, в свою очередь, приводит к повышению производительности и снижению вероятности ошибок. В целом, использование *Arduino* в данном проекте оказалось эффективным решением.

**Цитированная литература**

1. Ролф Кобли. Малая механизация. Машины, механизмы и оборудование для домашнего хозяйства / Ролф Кобли. – Харьков : Клуб Семейного Досуга, 2010. – 320 с.

2. Асмоловский, М. К. Механизация лесного хозяйства : учебное пособие / М. К. Асмоловский, С. Е. Арико, С. А. Голякевич. – Минск : РИПО, 2020. – 355 с.

3. Рачков, М. Ю. Автоматизация производства : учебник для среднего профессионального образования / М. Ю. Рачков. – 2-е изд., испр. и доп. – Москва : Юрайт, 2024. – 182 с.

4. *Arduino.* Полный учебный курс. От игры к инженерному проекту / А. А. Салахова, О. А. Феоктистова, Н. А. Александрова, М. В. Храмова. – 3-е изд., электрон. – Москва : Лаборатория знаний, 2024. – 178 с.

УДК 004.056

# ИССЛЕДОВАНИЕ МЕТОДОВ ЗАЩИТЫ ОТ ХИЩЕНИЯ ДАННЫХ ПОЛЬЗОВАТЕЛЕЙ С METASPLOIT

В. М. Русский, магистрант
Ю. А. Столяренко, доцент

***Аннотация.*** *Metasploit Project – проект, посвящённый информационной безопасности. Создан для предоставления информации об уязвимостях, помощи в создании сигнатур для IDS, создания и тестирования эксплойтов. Кроме того, проект включает в себя базу опкодов, архив шеллкодов и информацию по исследованиям информационной безопасности.*

***Ключевые слова:*** *защита информации, сети, тестирование, эксплойт, информационная безопасность.*

С ростом современных технологий и повсеместного использования интернета, защита данных пользователей становится все более важной задачей. Многие злоумышленники активно используют различные методы, включая программное обеспечение *Metasploit*, для хищения личной информации пользователей.

В 2003 году хакеру, известному как ***«HD Moore»***, пришла в голову идея разработать инструмент для быстрого написания и использования эксплойтов. Так на свет появился известный во всех кругах проект ***Metasploit project***.

Первая версия фреймфорка была написана на языке *Perl*, содержавшая псевдографический интерфейс на базе библиотеки *curses*. На тот момент это был просто набор разрозненных эксплойтов и скриптов, общие сведения о которых хранились в единой базе данных.

Информация о необходимом окружении для запуска скриптов, как правило, отсутствовала. Также они несли в себе кучу устаревшего кода, требовали модификации жестко прописанных путей для каждого конкретного случая, что весьма затрудняло рабочий процесс и усложняло разработку новых инструментов.

При работе над второй (2.x) версией к *HD Moore* присоединился ***Мэтт Миллер*** и еще несколько добровольцев. Третья версия была полностью переписана на *Ruby*, её разрабатывала компания *Metasploit LLC* (основанная все теми же разработчиками в 2006 году). В 2008 году лицензия ***Metasploit Framework*** была сменена с проприетарной на *BSD*. А еще позднее, в 2009 году, фирма *Rapid7*, занимающаяся управлением уязвимостями, объявила о приобретении ***Metasploit*** – программного пакета двойного назначения для проведения теста на проникновение.

Так же сообщалось, что некоммерческая версия утилиты по-прежнему будет доступна для всех желающих [1].

## Кто использует *Metasploit*

Благодаря широкому спектру применений и доступному открытому исходному коду *Metasploit* используется самыми разными людьми, от профессионалов кибербезопасности до хакеров. *Metasploit* полезен для всех, кому нужен простой в установке и надежный инструмент, выполняющий свою работу независимо от платформы или языка. Это программное обеспечение пользуется популярностью у хакеров и широко доступно, что мотивирует специалистов по безопасности изучать платформу *Metasploit*, даже если сами они ей не пользуются.

Современная версия *Metasploit* содержит свыше 1677 эксплойтов для более 25 платформ, включая *Android*, *PHP*, *Python*, *Java*, *Cisco* и другие [2].

## Эксплойты

*Эксплойт* – это фрагмент кода, который использует уязвимость в системе. Эти эксплойты выполняют определенные действия в зависимости от того, насколько серьезна уязвимость.

Эксплойты могут использовать уязвимости программного обеспечения, аппаратного обеспечения, уязвимости нулевого дня и так далее. Некоторые из распространенных эксплойтов включают переполнение буфера, *SQL*-инъекции и так далее.

*Metasploit* предлагает ряд эксплойтов, которые вы можете использовать в зависимости от существующих уязвимостей в целевой системе. Эти эксплойты можно разделить на *два типа:*

1. *Активные эксплойты.* Запускаются в целевой системе, используют систему, предоставляют вам доступ или выполняют определенную задачу, а затем завершают работу.

2. *Пассивные эксплойты.* Пассивные эксплойты будут ждать, пока целевая система подключится к эксплойту. Этот подход часто используется хакерами в Интернете, которые просят вас загрузить файлы или программное обеспечение. Как только вы это сделаете, вы подключитесь к пассивному эксплойту, запущенному на компьютере хакера.

## Полезные нагрузки

*Полезная нагрузка* – это фрагмент кода, который выполняется с помощью эксплойта. Эксплой-

ты используются для проникновения в систему, а полезные нагрузки – для выполнения определенных действий.

Например, вы можете использовать кейлоггер в качестве полезной нагрузки вместе с эксплойтом. Как только эксплойт будет успешным, он установит кейлоггер в систему цели.

*Metasploit* предлагает хорошую коллекцию полезных нагрузок, таких как *reverse shells*, *bind shells*, *Meterpreter* и так далее.

Есть несколько полезных нагрузок, которые будут работать с большинством эксплойтов, но требуется некоторое исследование, чтобы найти правильную полезную нагрузку, которая будет работать с эксплойтом.

В *Metasploit* есть несколько типов полезных нагрузок. В конечном итоге вы будете использовать чаще всего следующие *три типа:*

*Синглы* – полезные нагрузки, работающие сами по себе, например, кейлоггеры.

*Stagers* – полезные нагрузки, которые работают с другими, например, две полезные нагрузки: одна для установления соединения с целью, другая для выполнения инструкции.

*Meterpreter* – расширенная полезная нагрузка, которая хранится в памяти цели, ее трудно отследить, и она может загружать/выгружать плагины по желанию.

**Meterpreter**

*Meterpreter* – это продвинутая полезная нагрузка в *Metasploit*. В отличие от других полезных нагрузок, выполняющих определенную функцию, *Meterpreter* динамичен и может быть написан по сценарию «на лету».

Если вы можете использовать систему и вводить *Meterpreter* в качестве полезной нагрузки, вот некоторые из вещей, которые вы можете сделать:

1. Установите зашифрованную связь между вашей системой и целью.
2. Сброс хэшей паролей из целевой системы.
3. Поиск файлов в файловой системе цели.
4. Загрузка файлов.
5. Делайте снимки с веб-камеры.

*Meterpreter* также невероятно скрытен. Поскольку *Meterpreter* хранится в памяти цели, его чрезвычайно трудно обнаружить. Также сложно отследить *Meterpreter* с помощью инструментов судебной экспертизы.

Так можно писать скрипты *Meterpreter* «на лету», используя *Ruby* для выполнения пользовательских функций. В *Meterpreter* также есть модуль *Python*, который предоставляет вам дополнительные команды для выполнения скриптов *Python* на целевой машине.

**Армитаж**

*Armitage* – это графический пользовательский интерфейс для *Metasploit*, написанный на *Java*. *Armitage* считается отличным дополнением для начинающих тестировщиков, знакомых с интерфейсом командной строки.

Основная *функция Armitage* – визуализировать цели и рекомендовать эксплойты. *Armitage* также поддерживает сценарии, что означает, что вы можете автоматизировать избыточные задачи, такие как обнаружение хоста.

*Armitage* чрезвычайно полезен при работе с большим количеством систем в сети. Вы можете использовать графический интерфейс *Armitage* для повышения привилегий, просмотра файлов, сброса хэшей паролей и так далее.

**Msfdb**

*Metasploit* предлагает инструмент управления базами данных под названием *msfdb*. *msfdb* работает поверх базы данных *PostgreSQL* и предоставляет вам список полезных команд для импорта и экспорта ваших результатов.

С помощью *msfdb* вы можете импортировать результаты сканирования из внешних инструментов, таких как *Nmap* или *Nessus*. *Metasploit* также предлагает встроенную команду *db_nmap*, которая позволяет сканировать и импортировать результаты с помощью *Nmap* в *msfconsole* [3].

**Атака уязвимостей *AD CS ESC***

Служба сертификатов *Active Directory Certification Services*, также известная как *AD CS*, позволяет администраторам выпускать и управлять сертификатами открытых ключей, которые можно использовать для подключения к различным службам и принципалам в домене. Он часто используется для предоставления сертификатов, которые можно использовать вместо учетных данных для входа в сеть, или для предоставления сертификатов, которые можно использовать для подписи и проверки подлинности данных.

Основными гарантиями, которые стремится предоставить *AD CS*, являются:

– конфиденциальность посредством шифрования;

– целостность посредством цифровых подписей;

– аутентификация путем связывания ключей сертификата с компьютерами, пользователями или учетными записями устройств в компьютерной сети.

Учитывая, что *AD CS* часто хранит высокочувствительные ключи и учетные данные доступа к корпоративной сети, это делает его основной мишенью для злоумышленников.

На приведенной выше схеме показано, как можно атаковать четыре распространенные уязвимости *AD CS*, используя различные недостатки в настройке шаблонов сертификатов на сервере сертификации *Active Directory*.

В следующих разделах мы рассмотрим каждый из этих шагов, начиная с перечисления шаблонов сертификатов, которые может предложить сервер, и выявления тех, которые уязвимы для различных неправильных конфигураций и недостатков безопасности, затем создания новых сертификатов с использованием этих шаблонов сертификатов с помощью модуля *icpr_cert Metasploit*, и, наконец, использования этих сертификатов для аутентификации в домене в качестве администратора домена через *Kerberos*.

Каждая уязвимость шаблона сертификата, о которой пойдет речь, имеет *ESC*-код, например *ESC1*, *ESC2*. Эти *ESC*-коды взяты из оригинального документа, опубликованного *SpecterOps,* который популяризировал эти атаки на шаблоны сертификатов, известные как *Certified Pre-Owned*. В этом документе Уилл Шредер и Ли Кристенсен описали 8 различных атак на эскалацию домена, которые, как они обнаружили, можно было провести с помощью неправильно сконфигурированных шаблонов сертификатов:

– *ESC1* – эскалация домена с помощью шаблонов *OID No Issuance* Requirements + *Enrollable Client Authentication*/*Smart* Card *Logon* + *CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT;*

– *ESC2* – эскалация домена через *No Issuance Requirements* + зачисляемый *EKU* любого назначения или без *EKU;*

– *ESC3* – эскалация домена через *No Issuance Requirements* + *Certificate Request Agent EKU* + отсутствие ограничений агента регистрации;

– *ESC4* – эскалация домена через неверно настроенный контроль доступа к шаблону сертификата;

– *ESC5* – эскалация домена через уязвимый контроль доступа к объектам *PKI AD;*

– *ESC6* – эскалация домена через настройку *EDITF_ATTRIBUTESUBJECTALTNAME2* на *Cas* + отсутствие одобрения менеджера + шаблоны *OID* аутентификации клиента/входа в систему смарт-карты;

– *ESC7* – уязвимый контроль доступа к центрам сертификации;

– *ESC8 – NTLM* ретрансляция на конечные точки *AD CS HTTP;*

– *ESC9 – No Security Extension* – флаг *CT_FLAG_NO_SECURITY_EXTENSION* установлен в *msPKI-EnrollmentFlag*. Также *StrongCertificateBindingEnforcement* не установлен в 2 или *CertificateMappingMethods* содержит флаг *UPN*;

– *ESC10* – слабое сопоставление сертификатов – *HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\SecurityProviders\Schannel CertificateMappingMethods* содержит бит *UPN aka 0x4* или *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Kdc StrongCertificateBindingEnforcement* установлен в 0;

– ESC11 – *Relaying NTLM to ICPR* – передача *NTLM*-аутентификации на незащищенный *RPC*-интерфейс разрешена из-за отсутствия флага *IF_ENFORCEENCRYPTICERTREQUEST* в *Config.CA.Interface.Flags*.

В настоящее время *Metasploit* поддерживает атаки только на *ESC1*, *ESC2*, *ESC3* и *ESC4*.

Прежде чем продолжить, следует отметить, что *ESC1* немного отличается от *ESC2* и *ESC3*, как показано на диаграмме выше. Это связано с тем, что в *ESC1* у пользователя есть контроль над полем *subjectAltName* в генерируемом сертификате, которое также известно как поле *SAN.* Это поле позволяет указать, от имени кого должен аутентифицироваться сертификат.

Поэтому злоумышленнику достаточно изменить это поле, чтобы получить сертификат, который позволит ему аутентифицироваться как любой пользователь [4].

**Заключение**

В ходе исследования проблемы защиты данных пользователя от хищения с использованием *Metasploit* было выявлено несколько ключевых аспектов. *Metasploit*, как мощный инструмент для тестирования на проникновение, обладает потенциалом стать серьезным угрожающим фактором для конфиденциальности данных, если не принимаются соответствующие меры по защите.

Одним из основных выводов является необходимость регулярного аудита системы на наличие уязвимостей, которые могут быть использованы *Metasploit* для вторжения. Это включает в себя постоянное обновление программных средств, контроль доступа и регулярные проверки на соответствие стандартам безопасности.

Кроме того, важным аспектом является обучение пользователей безопасным практикам в интернете и внутриорганизационным политикам безопасности. Сознание пользователей о потенциальных угрозах и методах защиты является неотъемлемой частью комплексной стратегии по предотвращению хищения данных.

Технические средства защиты, такие как брандмауэры, антивирусные программы и системы обнаружения вторжений, также играют важную роль в минимизации рисков. Эффективное использование этих инструментов требует постоянного мониторинга и актуализации, чтобы обеспечить надежную защиту от атак, в том числе с использованием *Metasploit*.

В целом, для успешной защиты от хищения данных пользователя с использованием *Metasploit* необходимо комплексный и системный подход к безопасности, включающий технические, организационные и образовательные меры.

**Цитированная литература**

1. Секреты *Metasploit* [Электронный ресурс] – URL : https://habr.com/ru/articles/234719/

2. Что такое *Metasploit* [Электронный ресурс] – URL : https://habr.com/ru/companies/varonis/articles/528578/

3. *Metasploit* пошаговое руководство – Википедия – URL : https://www.freecodecamp.org/news/metasploit-a-walkthrough-of-the-powerful-exploitation-framework/

4. Документация *Metasploit* [Электронный ресурс] – URL : https://docs.metasploit.com/

УДК 621.316.925.1

# ИССЛЕДОВАНИЕ ВОЗМОЖНОСТИ ПРИМЕНЕНИЯ СИСТЕМ ВЫПРЯМЛЕННОГО ОПЕРАТИВНОГО ТОКА НА ПОДСТАНЦИИ 110/35/10 кВ

Н. Г. Стайков, бакалавр
М. В. Киорсак, профессор

***Аннотация.*** *В статье рассматриваются особенности применения и устройства элементов системы выпрямленного оперативного тока. Рассматриваются преимущества и недостатки системы выпрямленного оперативного тока, проведено технико-экономическое сравнение.*

***Ключевые слова:*** *оперативный ток, блок питания, выпрямители, схема питания, стабилизация.*

Совокупность источников питания, кабельных линий, шин питания переключающих устройств и других элементов оперативных цепей составляет систему оперативного тока данной электроустановки [5]. Оперативный ток на подстанциях служит для питания вторичных устройств, к которым относятся оперативные цепи защиты, автоматики и телемеханики, аппаратура дистанционного управления, аварийная и предупредительная сигнализация. При нарушениях нормальной работы подстанции, оперативный ток используется также для аварийного освещения и электроснабжения электродвигателей (особо ответственных механизмов). Схемы с блоками питания выпрямленным током широко применяются на понижающих подстанциях с номинальным напряжением до 35 кВ, а также на подстанциях с номинальным напряжением 110–220 кВ с упрощенными схемами электрических соединений со стороны высшего напряжения. Также на современных подстанциях при строительстве применяется вариант модульными выпрямителями и их управлением.

Блоки питания типов БПТ-1002 и БПН-1002 выполнены в виде одинаковых по габаритам комплектов, состоящих из цоколя, на котором располагаются все элементы блока, и съёмного кожуха. Конструкция блоков предусматривает возможность переднего или заднего присоединения внешних приводов (рис. 1).

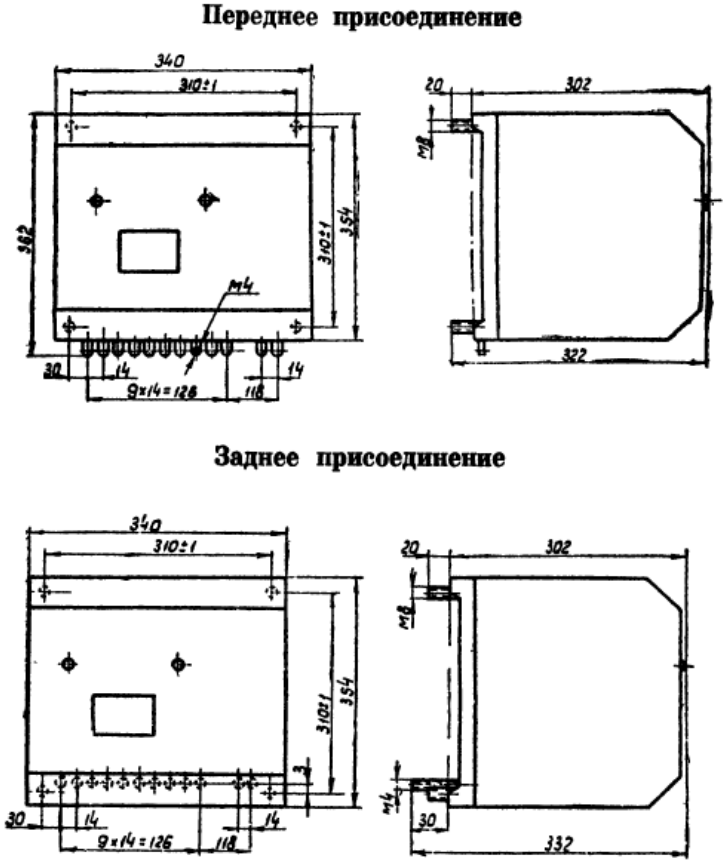


Рис. 1. Габаритные размеры БПТ-1002 и БПН-1002

БПТ-1002 (рис. 2) состоит из насыщающегося трансформатора тока с мостом из кремниевых выпрямителей на выходе. Для снижения амплитудных значений напряжения на выходе, а также для стабилизации его среднего значения параллельно вторичной обмотке насыщающегося трансформатора включаются ёмкость и дроссель, образующие с ветвью намагничивания трансформатора феррорезонансный контур. Переключения на плите дросселя для получения напряжения 110 и 220 В.

Во вторичной обмотке насыщающегося трансформатора и обмотке дросселя, и обмотке дросселя предусмотрены также ответвления для регулировки тока наступления феррорезонанса.

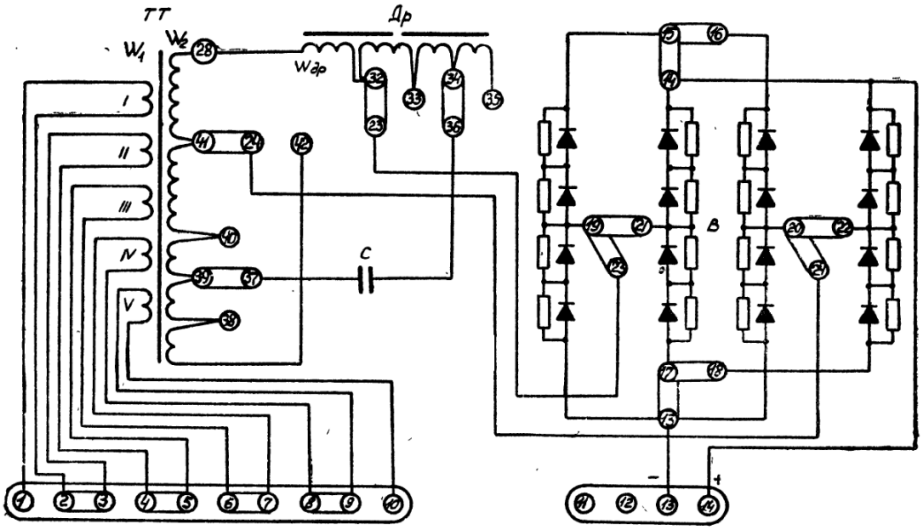


Рис. 2. Принципиальная схема блока питания БПТ-1002

БПН-1002 состоит из промежуточного трансформатора напряжения и двух трёхфазных мостов из кремниевых выпрямителей на выходе.

Параллельно мостам для защиты кремниевых выпрямителей от коммутационных перенапряжений во вторичных цепях включены селеновые столбы, примененные здесь в качестве нелинейных сопротивлений.

В зависимости от соединения мостов (последовательного или параллельного) на выходе блока можно получить напряжение 110 или 220 В постоянного тока. Для включения блока. На номинальные напряжения 110, 220 или 380 В предусмотрена возможность параллельного и последовательного соединения секций первичных обмоток каждой фазы трансформатора и включения их в звезду или треугольник [1].

Блок питания типа БПН-1002 (рис. 3) может включаться также на напряжения 100 и 127 В. Чтобы уровень выходного напряжения оставался постоянным при включении блока на различные входные напряжения, во вторичных обмотках предусмотрены ответвления, выведенные на плиту трансформатора к переключателям. Направление стрелок на переключателях указывает увеличение числа витков вторичных обмоток. При этом ответвления с максимальным числом витков используются на подстанциях с низким уровнем входного напряжения (80–95 В).

В основу принципа работы стабилизированных блоков питания (БПТ), подключенных к трансформаторам тока, положена стабилизация среднего значения вторичного напряжения. Применения блоков разработаны и комбинированные схемы. Стабилизированные блоки питания типа БПНС-1 вместе с токовыми типа БПТ-1002 используются для питания от ТТ, а БПН-от ТН, или ТОН. Блоки БПГ и БПН работают обычно на общие шины выпрямленного напряжения. Характерное отличие блоков ВПТ к БПН состоит в том, что блоки БПН обеспечивают питанием оперативные цепи в нормальных условиях, когда на подстанции заведомо имеется напряжение, а блоки БПТ – в режимах короткого замыкания, когда блоки БПН не могут обеспечить питание вторичных устройств из-за большого снижения напряжения в первичных цепях.

В основу принципа работы стабилизированных блоков питания (БПТ), подключенных к трансформаторам тока, положена стабилизация среднего значения вторичного напряжения (рис. 4). Применения блоков разработаны и комбинированные схемы. Стабилизированные блоки питания типа БПНС-1 вместе с токовыми типа БПТ-1002 используются для питания от ТТ, а БПН-от ТН, или ТОН. Блоки БПГ и БПН работают обычно на общие шины выпрямленного напряжения. Характерное отличие блоков ВПТ к БПН состоит в том, что блоки БПН обеспечивают питанием оперативные цепи в нормальных условиях, когда на подстанции заведомо имеется напряжение, а блоки БПТ – в режимах короткого замыкания, когда блоки БПН не могут обеспечить питание вторичных устройств из-за большого снижения напряжения в первичных цепях.

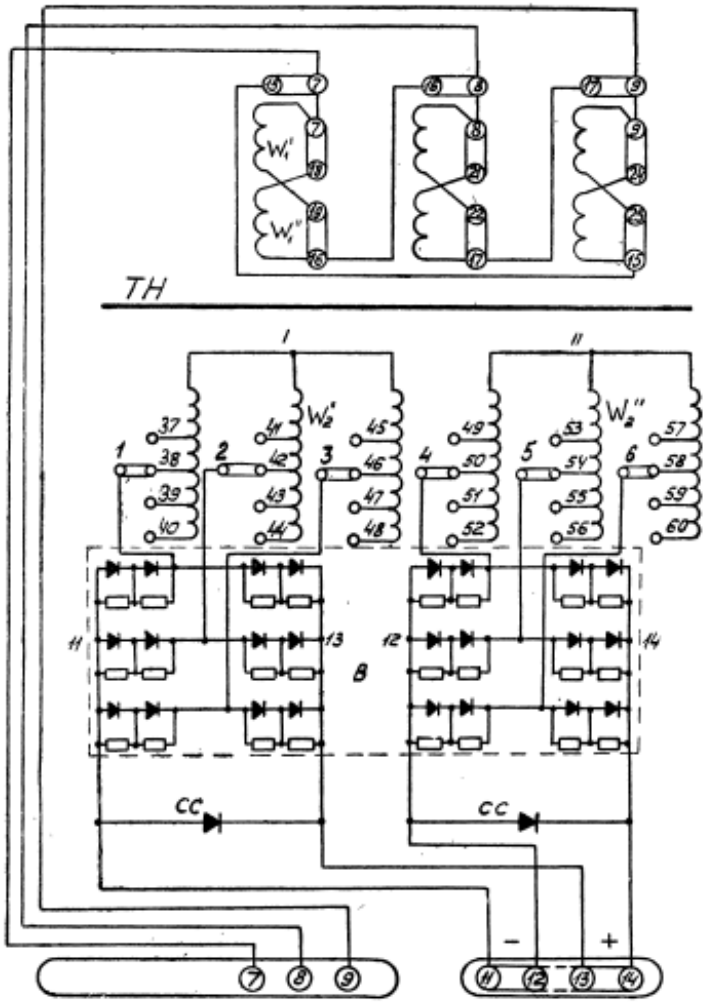


Рис. 3. Принципиальная схема блока питания типа БПН-1002

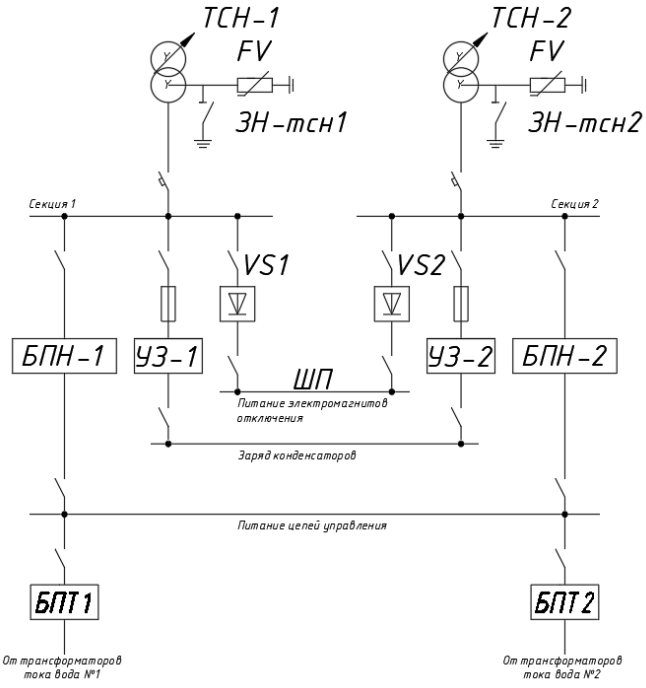


Рис. 4. Схема питания выпрямленный оперативным током

В состав системы выпрямленного оперативного тока входят следующие *компоненты:*

– блоки питания, стабилизированные типа БПНС-2 [3] совместно с токовыми типа БПТ-1002 – для питания цепей защиты, автоматики, управления;

– блоки питания нестабилизированные типа БПН-1002 – для питания цепей сигнализации и блокировки, что уменьшает разветвленность цепей оперативного тока и обеспечивает возможность выдачи всей мощности стабилизированных блоков для срабатывания защиты и отключения выключателей;

– блоки БПН-1002 вместо БПНС-2 – для питания цепей защиты, автоматики, управления, когда возможность их использования подтверждена расчетом и не требуется стабилизация оперативного напряжения (например, при отсутствии АЧР);

– силовые выпрямительные устройства ТЧ на УКП и УКПК с индуктивным накопителем – для питания включающих электромагнитов приводов масляных выключателей. Индуктивный накопитель обеспечивает включение выключателя на короткое замыкание при зависимом питании цепей включения;

– блоки питания нестабилизированные БПЗ-401 [4] применяются для заряда конденсаторов, которые используются для отключения отделителей, включения короткозамыкателей, отключения выключателей 10(6) кВ защитой минимального напряжения, а также отключения выключателей 35–110 кВ при недостаточной мощности блока питания.

Модульное исполнение выпрямительной системы позволяет обеспечить непрерывную подачу электропитания и упростить техническое обслуживание системы. Модуль выпрямительной системы можно использовать как самостоятельную единицу оборудования или в составе нескольких подключенных друг к другу модулей, размещенных в шкафу. При необходимости любой модуль можно быстро заменить, не останавливая критично важное оборудование. Также модульная реализация упрощает процесс интегрирования модулей с другим оборудованием, что минимизирует затраты на создание системы электропитания на объектах с ограниченной площадью.

Модульный выпрямитель «Форпост» [2] БПС-3000-380/220В-15А-14 обеспечит электроэнергией высокого качества приборы и устройства с напряжением питания 220 В и потребляемым током до 15 А на объектах с сетью переменного тока 380 В (3 фазы). Выпрямитель предназначен как для автономной, так и для параллельной работы в составе выпрямительных систем «Форпост» с выходным током до 480 А, количество выпрямителей без гальванической развязки ограничено 32-мя штуками по правилам техники безопасности.

Выпрямитель представляет собой импульсный преобразователь напряжения (рис. 5) трехфазного переменного тока в напряжение постоянного тока, размещенный в прочном металлическом корпусе. На передней панели выпрямителя расположены вентилятор, индикаторы наличия входного и выходного напряжения и аварии, ручка для переноски. На задней панели изделия находится разъем для подключения сети и нагрузки.

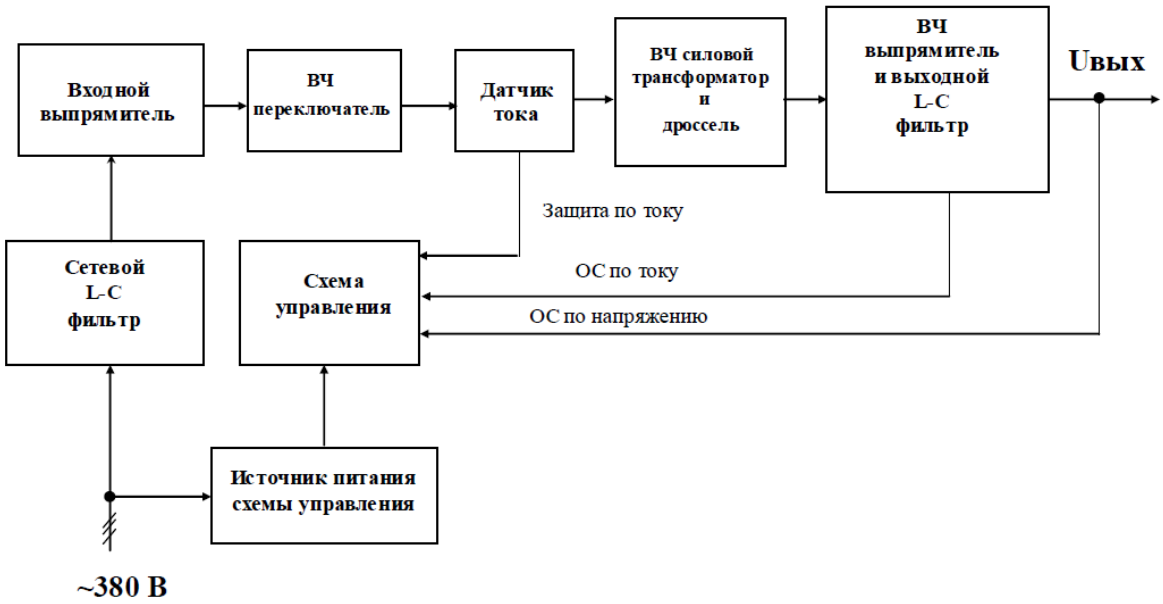


Рис. 5. Структурная схема БПС

*Достоинства:*

– возможность индивидуального обеспечения питания оперативным током одного защищаемого присоединения;

– возможность применения защитной аппаратуры, изготовляемой для установок с аккумуляторными батареями.

*Недостатки:*

– недостаточная мощность для питания катушек включения электромагнитных приводов, необходимо электропитание от ТСН;

– невозможность использования для минимальных защит напряжения, а также при отключении подстанции с упрощенной схемой соединений со стороны высшего напряжения для управления отделителем в бестоковую паузу;

– необходимость отдельных сердечников трансформаторов тока, когда требуется большая отдаваемая мощность.

*При технико-экономическом сравнении было выяснено следующее:*

– экономия средств при применении БПС-3000 составила 13,5 % относительно системы оперативного постоянного тока (СОПТ) на аккумуляторных батареях (АБ);

– экономия средств при применении БПТ/БПН составила 16,5 % относительно СОПТ на АБ.

Системы выпрямленного оперативного тока является необходимой для обеспечения стабильного питания различных устройств и систем. Применение данных систем обеспечивает экономию и позволяет осуществлять питание и контроль элементов оперативных цепей, также избавиться от колебаний напряжения.

**Цитированная литература**

1. Блоки питания типов БПТ 1002 и БПН 1002. Техническое описание и инструкция по эксплуатации. – Чебоксары, 1971. – 12 с.

2. БЛОКИ ПИТАНИЯ. Руководство по эксплуатации. – Москва, 2014. – 7 с.

3. БЛОКИ ПИТАНИЯ И ЗАРЯДА СЕРИИ БПЗ-400. Руководство по эксплуатации. – Москва, 1975. – 24 с.

4. Блок питания стабилизированным напряжением БПНС. Техническое описание и инструкция по эксплуатации. – Чебоксары, 2012. – 16 с.

5. Указания по областям применения различных видов оперативного тока на подстанциях 110 кВ и выше. Минэнерго СССР, ГПИО «ЭНЕРГОПРОЕКТ», ВГПИиНШ «ЭНЕРГОСЕТЬПРОЕКТ». – Москва, 1990. – 46 с.

УДК 004.056.55

# РЕАЛИЗАЦИЯ АЛГОРИТМА ШИФРОВАНИЯ ЭЛЬ-ГАМАЛЯ

В. В. Труханов, магистрант
Т. Д. Бордя, доцент

***Аннотация.*** *В статье описывается сам алгоритм шифрования «Эль-Гамаль», и что эта схема под собой подразумевает. Также рассмотрено проектирование приложения, ее структура и алгоритм работы. Проведено сравнение алгоритмов шифрования Эль-Гамаля, рассмотрены преимущества и недостатки.*

***Ключевые слова:*** *ПО – Программное Обеспечение, СУБД – Система Управления Базами Данных, SQLite – язык структурированных запросов и легкий, RSA – Rivest, Shamir и Adleman.*

Проблема защиты информации путем ее преобразования, исключающего ее прочтение мошенникам, волновала людей очень давно.

С широким распространением письменности криптография стала формироваться как самостоятельная наука.

Для поддержания работы ПО существует комплексная система автоматизации, которая позволяет собирать, анализировать и доводить до пользователей наиболее качественную и подробную информацию по заданной области [1].

Алгоритм Эль-Гамаля широко применяется в различных областях, связанных с криптографией, включая защиту информации в сетях передачи данных, цифровую подпись, аутентификацию и создание защищенных протоколов обмена данными. Его безопасность основана на сложности решения математических задач, связанных с дискретным логарифмом, что делает его устойчивым к атакам перебора и другим алгоритмическим методам расшифровки [2].

Выбор темы «Реализация алгоритма шифрования Эль-Гамаля» обуславливается тем, чтобы:

Изучить тему, как с математической точки зрения, так и с программной, чтобы понять сам алгоритм и предложить свои правки и улучшения в программу.

Составленные требования представляют как быстрое, так и удобное приложение для осуществления данной функции шифра данных в приложении. Сам список составлен с помощью графического веб-интерфейса.

Все рассмотренные требования к программному продукту «Алгоритм шифра Эль-Гамаля» изображены на рисунке 1.

Задачи программы, которые она будет выполнять [3]:

*Конфиденциальность данных* – работа алгоритма шифра сообщения для обеспечения безопасного прочтения сообщения другим абонентом.

*Проверка целостности данных* – работа цифровой подписи сообщения, для проверки сообщения, что абонент является владельцем данного сообщения.

*Аутентификация* – работа проверки входа в систему для работы в программе.

*Архивация сообщения* – уменьшение объема зашифрованного сообщения.

*Быстрота и удобство использования* – обеспечение правильной работы программы без единой ошибки и скорости работы приложения.

*Интеграция с СУБД* – синхронизация программы с СУБД, для работы с базой данных.


Рис. 1. Требования к ПО «Алгоритм шифра Эль-Гамаля»

Сама программа написана на языке программирования *C#* платформы *Windows Forms,* так как сам язык программирования прост для понимания и составлен на данной платформе, чтобы пользователь мог взаимодействовать с элементами формы про-

граммы. Программа может интегрировать с СУБД, чтобы хранить ключи и данные пользователя в базе данных. Сама база данных составлена на СУБД *SQLite.*

Структура программного продукта включает в себя формы и классы, также права доступа использования приложения (администратор и пользователь), изображенные на рисунке 2 [4]. Также приложение содержит библиотеки и базу данных личного аккаунта. Важно учитывать верную библиотеку при разработке ПО «Шифр Эль-Гамаля».

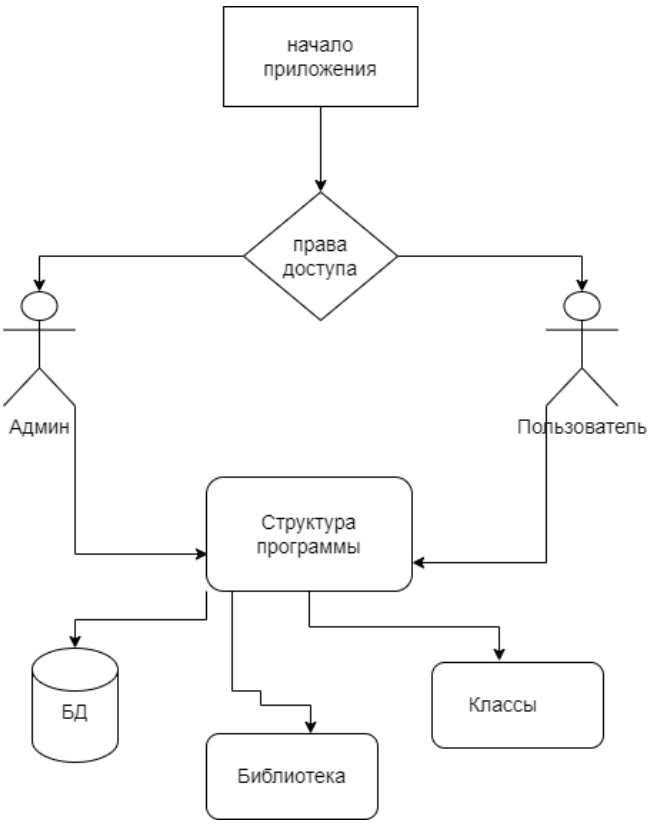


Рис. 2. Структура программного продукта

Как видно из рисунка 2, администратор и пользователь имеют доступ к форме приложения.

Алгоритм работы приложения имеет отдельные функции пользователя или администратора, изображенные на рисунке 3.

*Алгоритм работы программного продукта* начинается с таких пунктов, как:

1. Введение данных в форму авторизации.
2. Сравнение данных, введённых в форму.
3. Вход в форму администратора или в форму пользователя.
4. Функции админа – производить дешифровку шифра, в итоге получить полученное сообщение.
5. Функции пользователя – шифрование, расшифрование сообщения и подпись документа с сообщением.

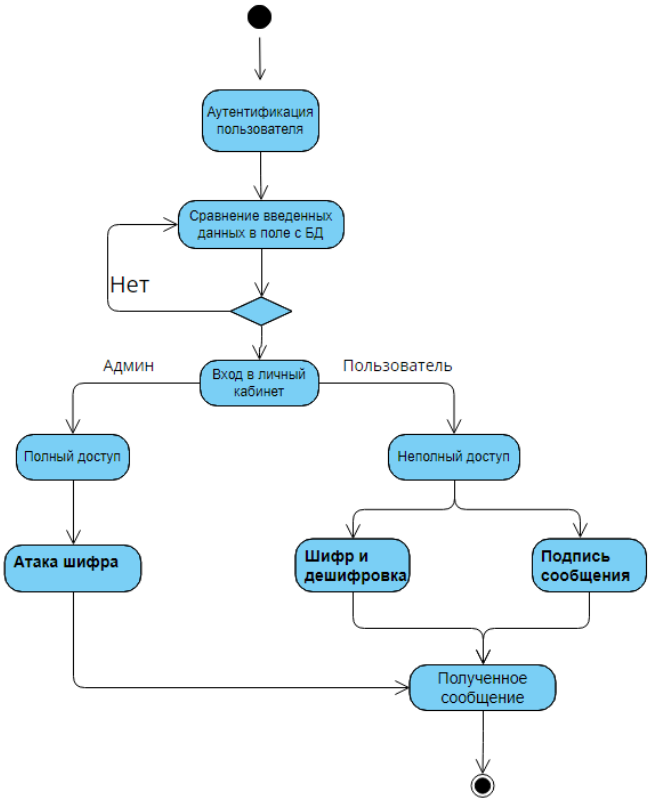


Рис. 3. Детальный алгоритм работы программы «Шифр Эль-Гамаля»

При реализации алгоритма Эль-Гамаля важно обратить внимание на безопасность и правильное использование криптографических примитивов. Лучшим подходом является использование проверенных и безопасных криптографических библиотек, которые обеспечивают надежность и защиту от уязвимостей.

Шифр Эль-Гамаля и алгоритм *RSA* являются двумя различными алгоритмами шифрования, применяемыми в криптографии для обеспечения конфиденциальности данных. Рассмотрим основные отличия между ними и предоставим краткий обзор их сильных и слабых сторон [7].

*Преимущества* шифра Эль-Гамаля:

1. Безопасность относительно задачи дискретного логарифмирования. Алгоритм Эль-Гамаля основан на сложности нахождения дискретного логарифма, что делает его стойким к атакам, связанным с этой задачей.

2. Открытый ключ может использоваться для шифрования. Эль-Гамаль можно использовать для шифрования данных и для подписи, что является дополнительным преимуществом.

*Недостатки* шифра Эль-Гамаля:

1. Вычислительная сложность. Алгоритм Эль-Гамаля обычно более вычислительно сложен, чем *RSA*, особенно при работе с большими числами.

2. Больший объем выходных данных. Зашифрованные сообщения в системе Эль-Гамаля имеют обычно больший объем по сравнению с *RSA*, что может быть проблемой в некоторых сценариях.

Теперь рассмотрим аналогичный алгоритм шифрования и проведем сравнение.

*Преимущества* алгоритма *RSA:*

1. Вычислительная эффективность для шифрования. *RSA*, как правило, более эффективен для операций шифрования, чем Эль-Гамаля.

2. Простота ключей. Генерация ключей *RSA* проще, чем в случае Эль-Гамаля.

*Недостатки* алгоритма *RSA*:

1. Уязвимость к факторизации. *RSA* подвержен атакам, основанным на факторизации больших чисел, что может сделать его менее стойким в сравнении с Эль-Гамаля.

2. Неэффективность подписи. *RSA* хотя и подходит для шифрования, но менее эффективен для создания цифровых подписей, чем Эль-Гамаля.

**Заключение**

В заключение следует отметить, что проблема обеспечения безопасности информации через криптографические методы сохраняет свою актуальность на протяжении всей истории человечества. Реализация алгоритма Эль-Гамаля в программном продукте предоставляет решение для различных областей криптографии, включая защиту информации в сетях передачи данных, цифровую подпись, аутентификацию и создание защищенных протоколов обмена данными. Безопасность алгоритма основана на сложности математических задач, связанных с дискретным логарифмом, что обеспечивает устойчивость к атакам перебора и другим алгоритмическим методам расшифровки.

Поставленные требования к программному продукту охватывают конфиденциальность данных, проверку целостности, аутентификацию, архивацию сообщений, обеспечивая при этом быстроту и удобство использования. Интеграция с СУБД расширяет функционал, обеспечивая работу с базой данных личных аккаунтов.

Структура программного продукта с отдельными формами и классами для администратора и пользователя обеспечивает безопасность и эффективность использования. Выбор правильной библиотеки при реализации алгоритма Эль-Гамаля становится важным моментом для обеспечения надежности и защиты от уязвимостей.

Сравнение алгоритма Эль-Гамаля с *RSA* выявляет их различия в вычислительной сложности и эффективности. Выбор между ними зависит от конкретных требований и особенностей сценария использования. Реализация алгоритма Эль-Гамаля и создание программного продукта подчеркивают важность развития методов шифрования для обеспечения безопасности в информационном пространстве.

**Цитированная литература**


1. *Bruce Schneier. Applied Cryptography: Protocols, Algorithms, and Source Code in C. – Hoboken : John Wiley & Sons Inc, 1*996. – p. 758.

2. *Douglas R. Stinson. Cryptography: Theory and Practice. – London : Chapman & Hall / CRC,* 2002. – p. 360.

3. *Wade Trappe, Lawrence Washington. Introduction to Cryptography with Coding Theory. – London : Pearson,* 2005. – p. 600.

4. *Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone. Handbook of Applied Cryptography. – Boca Raton : CRC Press,* 1996. – p. 780.

5. *Dan Boneh. A Graduate Course in Applied Cryptography.*

6. Алферов, А. П. Основы Криптографии / А. П. Алферов, А. Ю. Зубов, А. С. Кузьмин, А. В. Черемушкин. – Москва : Гелиос, 2005. – С. 5–53.

7. Баричев, С. Г. Основы современной криптографии / С. Г. Баричев, В. В. Гончаров, Р. Е. Серов. – Москва : Горячая линия. – Телеком, 2002. – С. 4–8.

8. Жельников, В. Криптография от папируса до компьютера / В. Жельников. – Москва : ABF, 1996. – URL : http://www.fidel-kastro.ru/crypto/zhelnik.htm

9. *Diffie W., Hellman M. E. New directions in cryptography. IEEE transactions on Information Theory IT-22.* 1976. – 644–654 p.

10. Сингх, С. Книга шифров. Тайная история шифров и их расшифровки / С. Сингх. – Москва : АСТ, Астрель, 2006. – 447 с.

УДК 004.4

# ИССЛЕДОВАНИЕ ФРЕЙМВОРКОВ И ИХ ПРИМЕНЕНИЕ В РАЗРАБОТКЕ ВЕБ-ПРИЛОЖЕНИЙ

К. А. Чайковский, магистрант
Ю. А. Столяренко, доцент

***Аннотация.*** *Фреймворк (с англ. framework – «каркас, структура») – заготовка, готовая модель в IT для быстрой разработки, на основе которой можно дописать собственный код. Он задает структуру, определяет правила и предоставляет необходимый набор инструментов для создания проекта. В основном фреймворки используются в веб-разработке.*

***Ключевые слова:*** *фреймворк, веб-разработка, оптимизация, алгоритмы.*

В современном мире веб-приложения стали неотъемлемой частью нашей повседневной жизни. Они предоставляют нам доступ к различным сервисам и функционалу, от социальных сетей до банковских операций и облачных хранилищ. Однако разработка таких приложений может быть сложной и трудоемкой задачей. Здесь на помощь приходят фреймворки, которые значительно упрощают и ускоряют процесс создания веб-приложений.

*Фреймворк* – это наборы инструментов, библиотек и стандартов, предназначенные для облегчения разработки программного обеспечения. Они предоставляют разработчикам готовые решения для ряда распространенных задач, таких как маршрутизация, обработка запросов, взаимодействие с базами данных и многое другое. Фреймворки упрощают разработку, улучшают безопасность и улучшают масштабируемость приложений, а также способствуют соблюдению современных стандартов разработки.

Существует множество фреймворков для разработки веб-приложений, каждый со своими особенностями и преимуществами. Давайте рассмотрим несколько популярных фреймворков и их применение в разработке веб-приложений.

*1. Django*: *Django* является одним из самых популярных фреймворков для разработки веб-приложений на языке *Python*. Он обладает широким функционалом, включая автоматическую административную панель, маршрутизацию *URL*-адресов, обработку форм и многое другое. *Django* также предлагает мощные инструменты для работы с базами данных и обеспечивает высокую безопасность приложений.

*2. Ruby on Rails*: *Ruby on Rails* (или просто *Rails*) – это фреймворк для разработки веб-приложений на языке *Ruby*. Он привержен принципам «соглашение перед конфигурацией» (*Convention over Configuration*) и «*DRY*» (*Don't Repeat Yourself*). *Rails* предоставляет гибкую и простую в использовании структуру, включая *ORM* (*Object-Relational Mapping*) для работы с базами данных, автоматическую генерацию кода и множество готовых компонентов. Это делает *Ruby on Rails* отличным выбором для быстрой разработки веб-приложений с минимальными усилиями.

*3. Laravel*: *Laravel* является фреймворком для разработки веб-приложений на языке *PHP*. Он предлагает множество инструментов и функций, таких как маршрутизация, обработка запросов, шаблонизация, работа с базами данных и аутентификация. *Laravel* также обладает простым и понятным синтаксисом, что делает его привлекательным для разработчиков *PHP*.

*4. Angular*: *Angular* – это фреймворк для разработки фронтенд-части веб-приложений. Он использует язык *TypeScript* и предоставляет мощные инструменты для создания динамических и интерактивных пользовательских интерфейсов. *Angular* основан на компонентной архитектуре и предлагает множество функций, включая маршрутизацию, обработку событий, валидацию форм и многое другое.

*5. React*: *React* – это *JavaScript*-библиотека для разработки пользовательских интерфейсов. Вместе с библиотекой *React* можно использовать фреймворки, такие как *Next.js* или *Gatsby*, чтобы создавать полноценные веб-приложения. *React* основан на компонентном подходе и позволяет эффективно управлять состоянием приложения и обновлять только необходимые части интерфейса, что обеспечивает высокую производительность [5].

Применение фреймворков в разработке веб-приложений имеет ряд преимуществ. Во-первых, они значительно упрощают и ускоряют процесс разработки, предоставляя готовые инструменты и решения для повседневных задач. Во-вторых, фреймворки способствуют структурированию кода и соблюдению современных стандартов разработки, что повышает качество и улучшает масштабируемость приложений. Кроме того, фреймворки обеспечивают безопасность и защиту от распространенных уязвимостей.

В ходе исследования будет рассмотрен один из вышеперечисленных веб-фреймворков – *React*. Будут приведены основные алгоритмы, которые обеспечиваю производительность и работоспособность, будет проведен сравнительный анализ и приведены основные преимущества и недостатки.

*Таблица 1*

**Сравнительный анализ**

| **Показатель** | ***React*** | ***Angular*** |
|---|---|---|
| Разработчик | *Facebook* | *Google* |
| Тип | Библиотека для создания приложений пользовательских интерфейсов | Фреймворк для создания одностраничных приложений |
| Язык | *JavaScript*/*TypeScript* | *TypeScript* |
| Первый выпуск | 2013 | 2010 (*AngularJS*), 2016 (*Angular* 2+) |
| Архитектура | *Component-based* | *Component-based*, *MVC*/ *MVVM* |
| Обучаемость | Легко начать, но глубже сложнее | Сложнее для начального освоения |
| Скорость разработки | Быстрая, благодаря простоте и гибкости | Медленнее из-за высокой сложности и структурированности |
| Подход к обновлению *DOM* | *Virtual DOM* | Реальный *DOM* с улучшениями (*Change Detection*) |
| Связь данных | Односторонняя (*unidirectional data flow*) | Двусторонняя (*two-way data binding*) |
| Шаблоны | *JSX* (*JavaScript XML*) | *HTML* и *Angular*-шаблоны |
| Управление состоянием | *Redux*, *MobX*, *Context API* | *NgRx*, *Akita*, встроенное управление состоянием |
| Инструменты | *Create React App*, *Next.js*, *Gatsby* | *Angular CLI* |
| Документация | Хорошая, но иногда фрагментированная | Обширная и структурированная документация |
| Сообщество | Очень большое и активное | Тоже большое и активное, но меньше, чем у *React* |
| Используемые алгоритмы | *Virtual DOM* для оптимизации перерисовки компонентов | *Change Detection* и *Zone.js* для управления обновлением данных |
| Совместимость с другими библиотеками | Легко интегрируется с другими библиотеками | Меньше возможностей для интеграции с другими библиотеками |

*Таблица 2*

**Сравнительный анализ используемых алгоритмов**

| **Алгоритм** | ***React*** | ***Angular*** |
|---|---|---|
| *Virtual DOM* | Использует *Virtual DOM*, который создает виртуальную копию *DOM* дерева и сравнивает его с реальным *DOM* для минимизации изменений и повышения производительности. | Не использует *Virtual DOM*. Вместо этого применяет алгоритмы для эффективного обновления реального *DOM* через *Change Detection*. |
| *Change Detection* | В *React* обновление происходит благодаря *Virtual DOM*, который сравнивает текущий и предыдущий состояния компонента (*Diffing Algorithm*) и минимизирует манипуляции с реальным *DOM*. | *Angular* использует *Change Detection* для отслеживания изменений в данных. Каждый компонент проверяется на изменения при каждом цикле обновления, что оптимизируется через *Zone.js* и различными стратегиями обнаружения изменений (*Default* и *OnPush*). |
| *Data Binding* | Односторонняя привязка данных: данные передаются от родительского компонента к дочернему и обновления происходят только в одну сторону. | Двусторонняя привязка данных: данные могут синхронизироваться как из модели в вид, так и обратно, что упрощает работу с формами и интерфейсами, но может усложнять отслеживание состояния. |

*Основные алгоритмы:*

1. ***Virtual DOM***. *React* использует виртуальную *DOM* (объектную модель документа) для обновления пользовательского интерфейса. Виртуальный *DOM* – это облегченное представление реального *DOM* в памяти, которое позволяет *React* вносить изменения в пользовательский интерфейс, не манипулируя реальным *DOM*. Это ускоряет обновления, поскольку изменение виртуального *DOM* обходится дешевле, чем изменение реального *DOM*.

2. ***Reconciliation***. *Reconciliation* – это процесс, посредством которого *React* обновляет пользовательский интерфейс, чтобы отразить изменения в состоянии компонента. Алгоритм согласования – это набор правил, которые *React* использует для определения наиболее эффективного способа обновления пользовательского интерфейса. Алгоритм *Reconciliation* работает путем сравнения текущего виртуального дерева *DOM* с обновленным виртуальным деревом *DOM* и внесения минимального количества изменений, необходимых для приведения виртуального *DOM* в соответствие с обновленным состоянием. *React* сравнивает текущее виртуальное дерево *DOM* с обновленным деревом виртуального *DOM* и определяет минимальное количество изменений, необходимых для приведения виртуального *DOM* в соответствие с обновленным состоянием.

3. ***Batching***. *React* объединяет несколько изменений в одно обновление, уменьшая количество обновлений виртуального *DOM* и, в свою очередь, реального *DOM*.

Ниже будут рассмотрены преимущества и недостатки *React.js,* которые были выявлены в ходе исследования.

**Преимущества:**

1. Высокая производительность благодаря использованию *Virtual DOM,* который минимизирует изменения в реальном *DOM.*

2. Компонентный подход позволяет разделять пользовательские интерфейсы на модули, что делает код модульным, повторно используемым и легко поддерживаемым.

3. Односторонняя передача данных обеспечивает предсказуемое управление состоянием и упрощает отладку приложений.

4. Большая экосистема и сообщество предлагают обширный набор библиотек и инструментов, таких как *Redux* и *React Router,* а также множество ресурсов и активное сообщество.

5. *JSX (JavaScript XML)* позволяет писать *HTML*-подобный код внутри *JavaScript,* что делает код более читаемым.

6. Универсальность: *React* подходит как для веб-приложений *(React),* так и для мобильных приложений *(React Native).*

7. Поддержка со стороны *Facebook* гарантирует активное развитие и стабильную поддержку.

**Недостатки:**

1. Высокий порог вхождения: полное освоение всех возможностей требует значительного времени.

2. Неопределенность архитектуры: отсутствие встроенных решений для управления состоянием, маршрутизации и других задач.

3. Частые обновления и изменения могут привести к необходимости частых обновлений и адаптации к новым версиям.

4. *JSX* может быть непривычным: смешивание *JavaScript* и *HTML* может быть сложным для разработчиков, привыкших к традиционному разделению этих языков.

5. Проблемы с *SEO*: приложения, работающие только на клиентской стороне, могут сталкиваться с проблемами индексации поисковыми системами.

6. Документация и ресурсы: хотя у самого *React* хорошая документация, многие используемые библиотеки могут иметь менее качественную документацию.

Итак, исследование фреймворков для разработки веб-приложений является важным шагом в поиске оптимальных инструментов для создания современного программного обеспечения. Рассмотренные фреймворки предоставляют разработчикам широкий спектр возможностей для реализации различных проектов, учитывая их требования к функциональности, производительности и безопасности.

Каждый фреймворк имеет свои особенности, преимущества и недостатки, что подчеркивает важность выбора инструмента в соответствии с уникальными потребностями конкретного проекта. При этом понимание особенностей каждого фреймворка, его экосистемы, поддержки и потенциала для масштабирования играет решающую роль в успешной разработке веб-приложений.

Использование фреймворков значительно упрощает процесс разработки, улучшает структурирование кода, способствует повышению производительности и облегчает поддержку приложений. Однако важно помнить, что выбор фреймворка должен быть обоснованным и основываться на анализе требований проекта, специфики задачи и потенциала инструмента для достижения поставленных целей.

**Цитированная литература**

1. *Amin A., Qureshi B. React.js: Building Modern Web Applications. Journal of Web Engineering,* 18(2). – 2020. – p. 113-129.

2. *Zhang H., Smith J. Performance Optimization in React Applications. International Journal of Frontend Development,* 12(4). – 2019. – p. 56-67.

3. Кузнецов, И. В. Применение *React.js* в разработке интерактивных пользовательских интерфейсов. Вестник Технического Университета /

И. В. Кузнецов, Е. А. Попова. – 29(3). – 2021. – С. 47–60.

4. *Griffin M., James T. State Management in React: An Overview of Redux and Context API. Proceedings of the International Conference on Web Development.* – 2018. – p. 302-310.

5. Официальный сайт *React.js.* – URL : https://reactjs.org/

6. Чернов, А. П. Архитектурные подходы к построению масштабируемых приложений на *React.js.* Вестник Программной Инженерии и Компьютерных Наук / А. П. Чернов. – (4). – 2022. – С. 85–98.

УДК 004.056.55

# АНАЛИЗ РЕДАКТОРОВ LaTeX

Ю. В. Чеботарь, магистрант
Т. Д. Бордя, доцент

***Аннотация.*** *Статья представляет собой обзор некоторых редакторов LaTeX. Рассмотрены их достоинства и инструменты. Был проведён обзор их возможностей, анализ скорости работы, выделены критерии сравнения. Уделено внимание вопросу об удобстве использования и работы редакторов, а также вопросу о поддержке макропакетов LaTeX редакторами.*

***Ключевые слова:*** *LaTeX, редактор, документ, функция, код.*

При использовании системы вёрстки *TeX (LaTeX)* неизбежно встаёт вопрос об инструментах, позволяющих работать в этой системе. За время существования *LaTeX* было создано множество редакторов для работы с ним, поэтому актуальна задача их анализа с целью нахождения наиболее оптимального варианта для будущих работ.

### *TexMaker*

Бесплатный редактор *LaTeX*.

Обладает возможностью «пошаговой компиляции». Большинство редакторов компилируют *.tex* в *.pdf* «сразу» , *TeXMaker* позволяет собрать документ в формате *.dvi* и перекомпилировать его в *.pdf* потом. Благодаря этому пользователь может получить документ в формате *.dvi*, если ему это нужно. Также доступен формат *.ps* из *.dvi*.

Отдельные меню редактора посвящены автовставке часто встречающегося кода, например, код класса документа, код для работы со списками, для вставки файлов, таблиц и т. д. (рис. 1, 2).

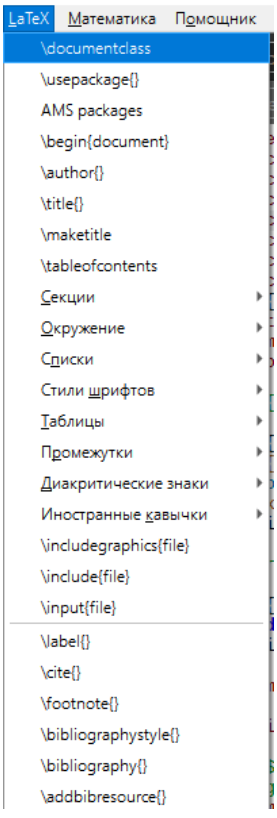


Рис. 1. Пример быстрых команд

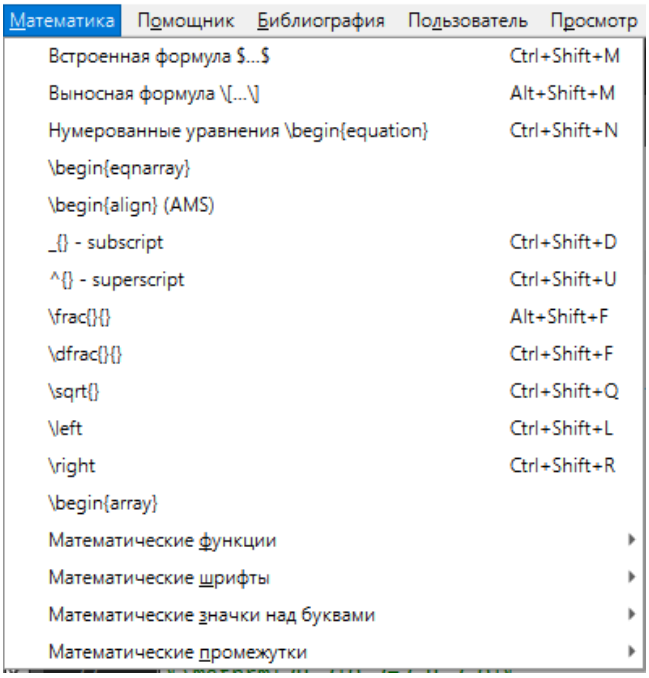


Рис. 2. Пример математических команд

Отдельная колонка интерфейса отдана под быстрый набор команд пакетов *TikZ*, *Asymptote*, *PSTricks* и команд *Metapost* (рис. 3, 4).

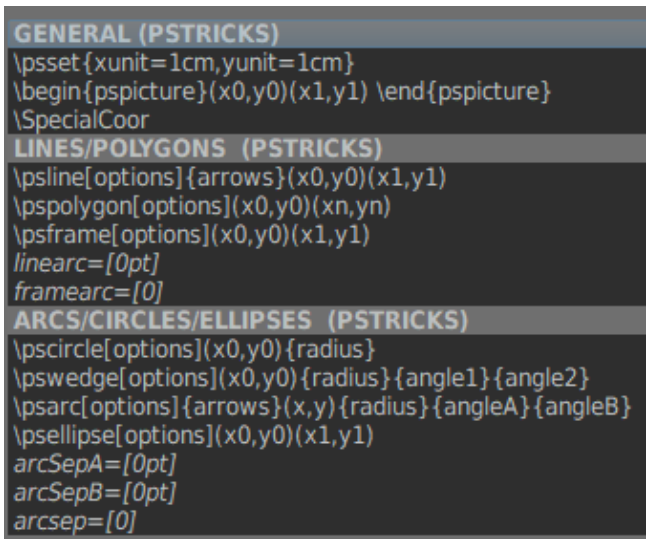


Рис. 3. Быстрые команды *PSTricks*

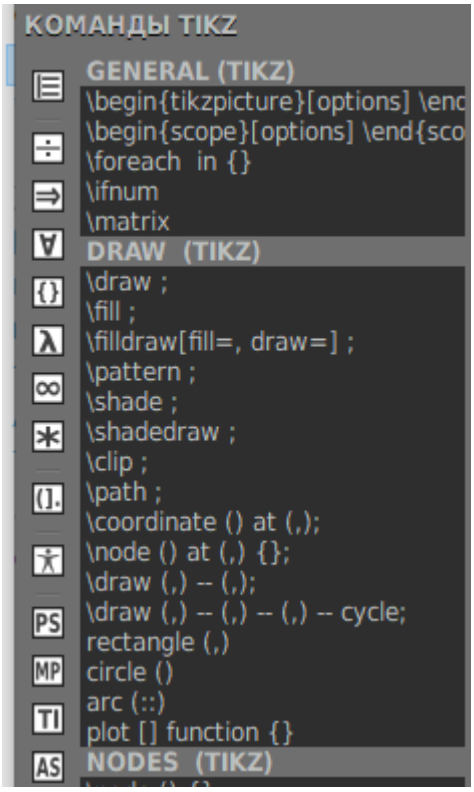


Рис. 4. Пример быстрого доступа к командам пакета *Tikz*

Также отдельная часть интерфейса отдана под быстрый набор символов – греческих букв, разделителей, стрелок (рис. 5).

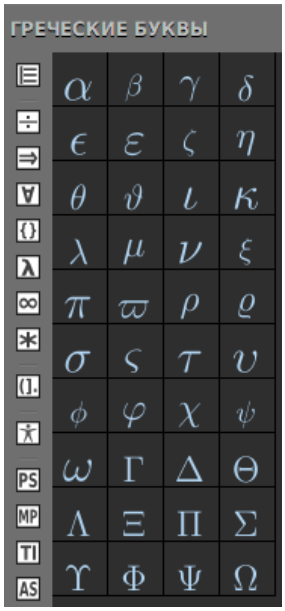


Рис. 5. Быстрый доступ к символам

Редактор обладает возможностью «быстрого создания» документа – пользователь выбирает класс документа, размер шрифта, бумаги и может подключить некоторые пакеты (рис. 6).

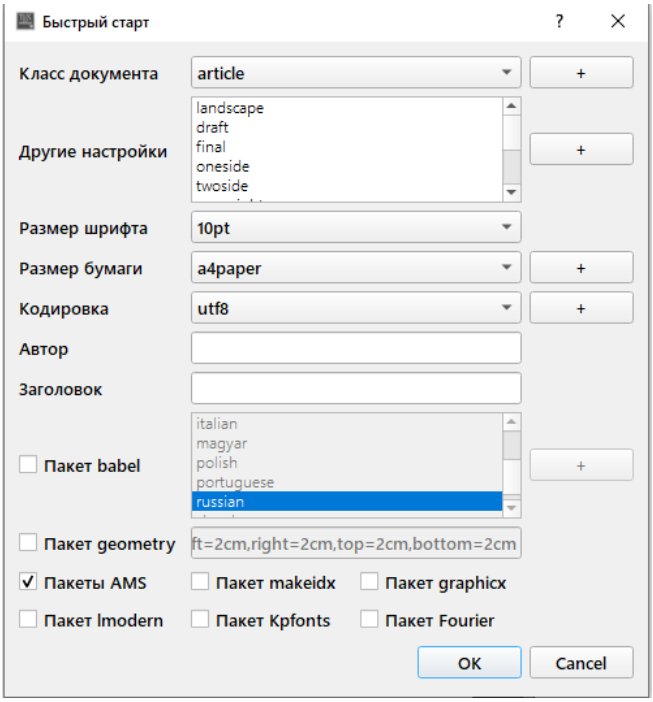


Рис. 6. Быстрое создание документа

Меню настройки *TeXMaker* состоит из четырёх подменю:

• Меню «Команды» позволяет настроить консольные команды для макропакетов.

• Меню «Быстрая сборка» позволяет настроить механизм быстрой сборки, выбирая, какие макропакеты использовать для компиляции или написать свою консольную команду для этого. Для удобства последнего доступен мастер быстрой сборки, позволяющий «собрать» команду компиляции, не прибегая к консольным командам.

• Меню «Редактор» позволяет выбрать стиль шрифта редактора (не документа) и его размер, выбрать словарь, включить перенос по словам, автозавершение команд, показ номеров строк и автосохранение документа. Присутствует возможность настроить стиль (цвета редактора).

• Меню «Быстрые клавиши» позволяет настроить быстрые клавиши для компиляции и некоторых команд *LaTeX*.

Редактор позволяет быстро вставить код библиотеки *Bibtex*(*BibLateX*) для более удобного написания библиографии. Поддерживается вставка множества форматов библиографий.

Редактор позволяет создавать свои макросы для быстрой вставки в документ нескольких строк одновременно. Макросам можно давать имена. Всего можно создавать до десяти макросов (рис. 7).

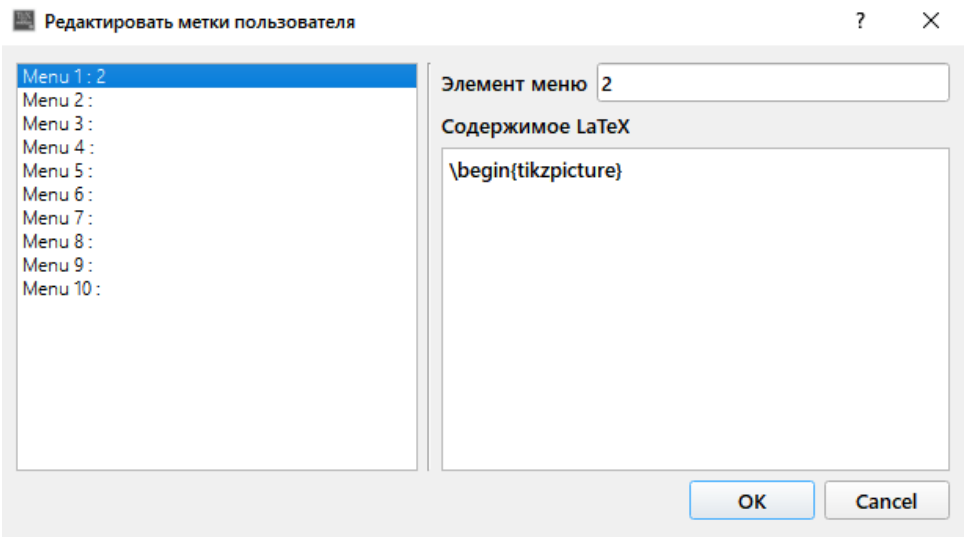


Рис. 7. Макросы

Так же можно создавать и свои команды, а также запускать *lua*-скрипты.

Можно настроить язык редактора (русский язык присутствует «из коробки»).

У редактора есть встроенные справки пользователя и *LaTeX*, а также автопроверка наличия *LaTeX* на компьютере.

Присутствует поиск по коду с возможностью замены, возможность перехода к определённой строке, а также быстрого комментирования и раскомментирования через опцию в меню (а не через код) и отмены изменений.

Есть возможность создавать «закладки» для строк кода для быстрого перемещения к ним [1].

***TexStudio***

Бесплатный *open-source* редактор *LaTeX*.

Редактор при установке ищет файлы *LaTeX* и его макропакетов, при отсутствии предлагает установить его через *MikTeX*. При попытке воспользоваться пакетом, отсутствующими в системе, вызывает меню установки пакета *MikTeX*.

Редактор имеет несколько инструментов для быстрой рутинной работы с текстом – выделение совпадающих с выделенной строк, поиск всех совпадений, перемещение, сортирование строк (по возрастанию или убыванию номеров либо случайным образом), быстрое дублирование (результат сразу записывается ниже оригинала), переход к строке по номеру. Можно также вместо того, чтобы перемещаться по номеру, поставить закладку на строке и перемещаться при необходимости уже по ним. Всего редактор поддерживает до 10 закладок. Также доступна опция комментирования и расскомментирования кода и быстрого создания и удаления отступов. Доступно закрытие незакрытых окружений. Можно выбирать содержимое внутри скобок.

Редактор обладает большим выбор символов для быстрой вставки через меню, чем *TexMaker* – обозначения тригонометрических функций, кириллица, кванторы, значки из пакетов *fontawesome*5 и *wasysym* и др. Также доступен по аналогии с *TeXMaker* быстрый набор команд пакетов *Tikz*, *Asymptote*, *PsTricks*, *MetaPost*, *Beamer* и *XY-pic* (последнего нет в *TeXMaker*)

*TeXStudio* обладает встроенным инструментом рисования математических формул с последующей вставкой их в код. Можно корректировать рисунок в процессе рисования – стереть определённую фигуру или скорректировать её (редактор выдаст список похожих фигур), отменять последний штрих или стирать рисунок полностью (рис. 8).

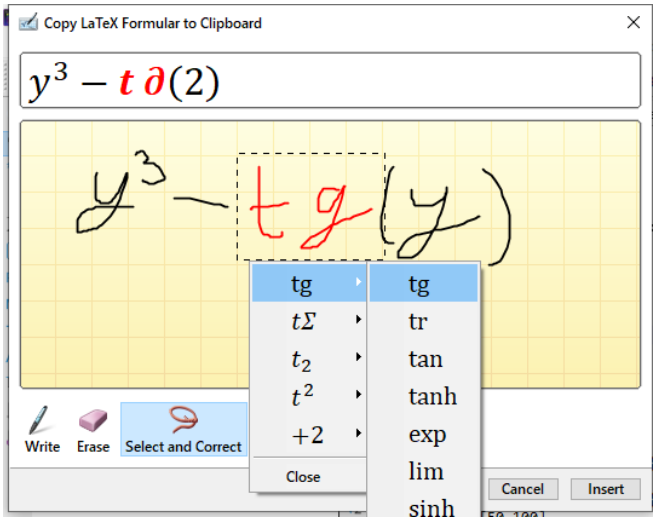


Рис. 8. Математический помощник

Также, как и *TeXMaker*, *TexStudio* позволяет скомпилировать *.tex*, *.dvi*, *ps* и *.pdf* файлы, как «сразу», так и через *.dvi*, также доступна компиляция в *html*.

Редактор позволяет писать свои макросы и экспортировать и импортировать их.

Можно сохранять текущие настройки конфигурации в файл и загружать их. Доступна обширная настройка горячих клавиш – на любое действие меню можно назначить горячую клавишу.

Доступна работа с системами управления версиями *SVN* (*TortoiseSVN*) и *Git* – загрузка файлов, сравнение версий и показ старых версий. Для этого нужно указать путь к исполняемым файлам желаемой системы.

Редактор позволяет просматривать не только весь документ целиком, но и его часть. Для этого нужно выделить желаемый фрагмент кода, и при успешной компиляции редактор даст результат обработки кода в рабочей области (можно изменить на всплывающую подсказку или просмотр в предпросмотре). Этот результат можно очистить специальной опцией меню. Также можно вставить скопированный текст как код *LaTeX*.

Есть быстрый доступ к командам окружения «сдвинутого» текста – по центру, по левому и правому краю, а также к разрыву строки.

Редактор не может, как *TeXMaker*, компилировать «комбинацией» макропакетов (но можно собирать с обработкой команд *Asymptote*).

Окно встроенного предосмотра можно редактировать – менять масштаб, разрешение экрана и форму лупы.

Также редактор обладает функцией автозавершения кода – редактор даёт список предполагаемых команд, похожих на введённый фрагмент кода, в выпадающем окне при написании кода. Набор команд, который использует редактор, можно поменять на часто используемый или весь набор (по умолчанию используется набор типичных, общеиспользуемых команд).

Редактор позволяет настроить кодировку итогового документа. Доступно множество кодировок, как семейства *UTF*, так и *ISO*.

Редактор обладает несколькими помощниками для быстрого создания элементов *LaTeX*:

– наглядный помощник вставки изображений – можно настроить масштаб изображения, выбрать центрирование, автоматически вставить окружение («кодовые скобки») и подписать рисунок. Настройка масштаба гибкая – можно выбирать длину в единицах измерения (миллиметры, сантиметры и пункты) (рис. 9);

– помощник создания документов – можно выбрать класс, размер шрифта, размер страницы, кодировку шрифта, язык и подключить некоторые пакеты (аналогично таковому в *TeXMaker*);

– помощник создания презентаций с помощью класса *Beamer* – можно выбрать тему и размер шрифта, вписать автора и заголовок;

– помощники создания таблиц и массивов – можно выбрать размер и заполнять колонки. В самом коде есть возможность быстро вставлять и удалять строки и столбцы через значки в панели быстрого доступа.

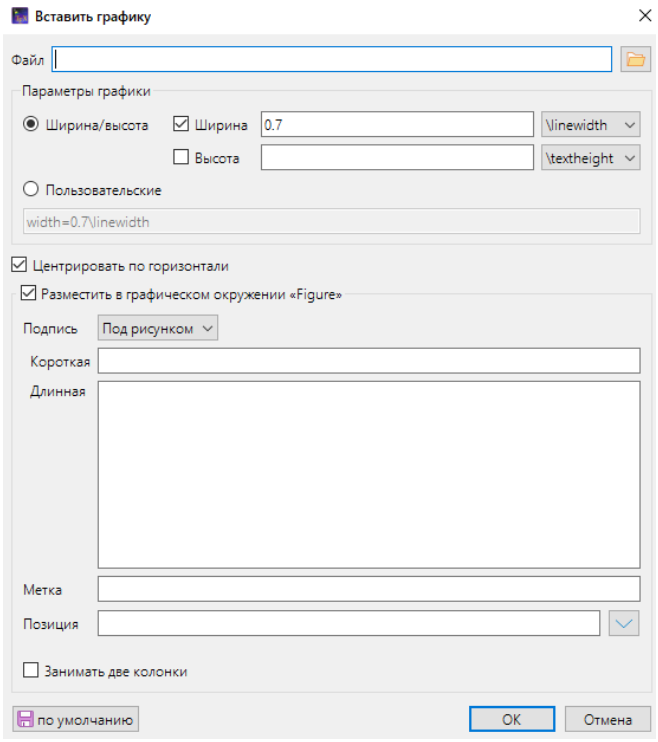


Рис. 9. Помощник вставки изображений *TeXStudio*

Предпросмотрщик *TeXStudio* позволяет просматривать страницу по номеру, а также менять размер окна документа.

Редактор позволяет быстро выбирать нужные документы в случае работы с несколькими как через функцию в меню, так и на верхней панели, где отображаются названия документов в виде вкладок. Некоторые быстрые команды также доступны на верхней панели [2].

### *VerbaTeX*

Простой редактор под *Android*. Обладает ограниченным набором команд (пакетов) при установке, годен только для простых документов. Способен сохранять документ как локально, так и загружать в *Dropbox*. Документ генерирует только в *.pdf*. Работает только с форматами *.tex* и *.bib*. Позволяет посмотреть лог ошибок.

### *Overleaf*

Это веб-редактор *LaTeX*, позволяющий создавать и работать с документами *LaTeX* в браузере. Редактор оперирует понятием «проект» . В проекте может быть один или несколько файлов. Может не только создавать документы, но и загружать их из другого источника – из файлов на компьютере, из другого проекта или получить файл по сторонней ссылке, и работать с ними. Также можно объединять файлы в папки, переименовывать или удалять их.

Главная особенность *Overleaf* – совместная работа над документом. Бесплатная версия позволяет пригласить одного человека в качестве помощника. Приглашение поработать присылается человеку на электронную почту, указанную работающим над документом – будущему помощнику достаточно за-

вести аккаунт на сервисе *Overleaf* и по ссылке на проект из письма приступить к редактированию документа. Все изменения видны обоим участникам – рабочее пространство синхронизируется между ними. Также есть возможность общения через чат между участниками проекта.

Редактор позволяет просматривать преобразованную в документ строку кода. Для этого нужно выбрать желаемую строку и воспользоваться специальным инструментом просмотра – в редакторе подсветится участок кода, в который «преобразилась» строка.

Редактор обладает возможностью просмотреть лог ошибок.

Документы создаются в формате *.tex*, с компиляцией их в *.pdf* и просмотром готового документа в этом формате. Документ можно скачать в обоих форматах.

Есть возможность просмотреть архив документа за последние 24 часа (полная версия предоставляет полную историю изменений) и отметить изменения в документе.

Также имеется визуальный редактор кода, позволяющий в режиме написания кода посмотреть текущий вид документа, а также позволяющий вставлять одним кликом математическую моду, ссылку, курсив, жирный шрифт, цитату, изображения и списки(в платной версии символы)

Поддерживает *LaTeX*, *pdfLaTex*, *XeLaTeX*, *LuaLaTeX*.

Есть автодополнение команд *TeX* и проверка грамматики на множестве языков.

Максимальное время компиляции документа ограничено (в бесплатной версии – 1 минута), поэтому редактор не рекомендуется к работе с файлами большого объёма. Также набор доступных пакетов для подключения ограничен.

***TeXworks***

Этот редактор поставляется вместе с установкой *MikiTeX*. *TeXWorks* представляет из себя простой редактор *LaTeX*, не содержащий в себе каких-либо «сложных» функций.

Редактор позволяет компилировать итоговый документ в *pdf*-формат и просматривать его в отдельном окне. Также в отдельном окне создаётся и редактируется сам документ (т. е. несколько документов редактируются в разных окнах).

Редактор позволяет производить поиск по коду и поиск по тексту в уже скомпилированном документе. Редактор также позволяет быстро поменять регистр букв в коде и сделать автоматические кавычки для команд *TeX* и автоотступы для комментариев. Редактор поддерживает *XeTex*, *XeLatex*, *BibTex*, *upTeX*, *upLateX*,*LuaLaTeX*[3].

Рассмотрим критерии использования редакторов:

– лёгкость использования для новичков – этому лучше всего отвечают *TeXMaker* и *TeXStudio*. Оба обладают удобным интерфейсом, множеством команд, вынесенных в быстрый набор, помощниками для создания документов и некоторых элементов документов;

– доступность – *TeXMaker* и *TeXStudio* доступны на *Windows*, *Linux* и *Mac*, *VerbaTeX* – только *Android*, *Overleaf* – браузерный редактор (*Chrome*, *Firefox*);

– цена – *TeXMaker*,*TexStudio*, *TexWorks* полностью бесплатны, *Overleaf* и *VerbaTe*X имеют платные версии;

– библиотека пакетов – *VerbaTeX* и *Overleaf* имеют свою непополняемую библиотеку пакетов, *TeXMaker*, *TeXStudio* и *TeXworks* позволяют обновлять библиотеку пакетов пользователю, поскольку требуют установленного дистрибутива *LaTeX*;

– настройка – возможностью настроить тему, шрифт редактора, словари обладают *TeXMaker*, *TeXStudio*. *VerbaTeX* позволяет настраивать только тему. В *TeXworks* и *Overleaf* отсутствует кастомизация и настройка редактора как таковая;

– производительность. Для измерения скорости обработки редакторами кода был создан тестовый документ и произведены измерения времени компиляции. Результаты (в секундах):*Verbatex* – 6, *Overleaf* – 6, *TeXmaker* – 7, *TexStudio* – 5, *TeXworks* – 8.

По результатам исследования можно сказать, что наиболее предпочтительным для работы с *LaTeX* на любом уровне будет редактор *TeXStudio*.

**Цитированная литература**

1. Документация *TeXMaker*. – URL : https://www.xm1math.net/texmaker/doc.html (дата обращения: 07.05.2023).

2. Официальный сайт *TeXStudio*. – URL : https://www.texstudio.org/ (дата обращения: 12.04.2023).

3. Официальный сайт *TeXworks*. – URL : https://tug.org/texworks/ (дата обращения: 24.04.2023).

УДК 004.056.53

# АНАЛИЗ ТРЕБОВАНИЙ И ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ, ПРЕДОСТАВЛЯЮЩЕГО ВОЗМОЖНОСТЬ ПОЛУЧЕНИЯ НАВЫКОВ В ОБЛАСТИ SQL-ИНЪЕКЦИЙ И СФЕРЕ ЗАЩИТЫ ОТ НИХ

В. С. Чиженков, магистрант
С. Л. Чирвина, ст. преподаватель

***Аннотация.*** *В данной статье описывается процесс анализа требований и проектирование приложения, с помощью которого пользователи могут получить навыки в области SQL-инъекций и сфере защиты от них. Рассматриваются и анализируются существующие разработки, определяются требования, предъявляемые к функциональным возможностям приложения. В статье при проектировании структуры и архитектуры программного продукта были построены и описаны различные схемы и диаграммы, а для схематичного отображения структур баз данных, проектируемых для приложения, были построены логические схемы для каждой базы данных.*

***Ключевые слова:*** *SQL-инъекции, проектирование приложения, информационная безопасность, анализ требований, разработка, СУБД.*

Высокая информатизация общества повышает значение вычислительной техники в управленческих процессах. Использование вычислительной техники для автоматизации процесса обработки информации позволяет повысить эффективность работы с документами, ускорить обмен оперативной информацией и увеличить производительность труда. Предприятия активно используют вычислительную технику для разнообразных целей в бухгалтерии, планировании, управлении [1].

Ввиду этого многие крупные организации хранят свои данные в различных базах данных управляемых СУБД (система управления базами данных). Для управления и взаимодействия с этими данными через СУБД используются различные клиент-серверные приложения.

Однако не редко эти приложения становятся целью атаки злоумышленников, желающих заполучить ценные данные каких-либо компаний. Полученные в ходе атаки данные могут обеспечить конкурентов этой компании огромным преимуществом или вовсе вывести из строя работу данной корпорации [2].

Пытаясь заполучить данные организаций, злоумышленники пользуются различными средствами, тактиками, стратегиями и методами, в том числе и *SQL*-инъекциями.

Без эффективных средств защиты от *SQL*-инъекций так и будут продолжаться утечки конфиденциальной информации. Но для того, чтобы разработать эти средства защиты, нужно понять каким образом эти *SQL*-инъекции внедряются в *SQL*-запросы отправляемые на сервер СУБД. А понимание того какие виды *SQL*-инъекций существуют и какими способами они внедряются в запрос, является первым, но немаловажным шагом в построении эффективной защиты против них.

Исходя из этого, была определенна цель – описать виды *SQL*-инъекций, способы их внедрения, изучить средства и методы защиты от них, выделить эффективные способы, разработать приложение, позволяющее получить навыки в области *SQL*-инъекций и сфере защиты от них.

Для достижения указанной цели были выделены следующие задачи:

− исследовать принцип работы СУБД и клиент-серверных приложений взаимодействующих с ними;

− изучить виды и методы отправления *SQL*-запросов;

− изучить способы внедрения *SQL*-инъекций в *SQL*-запросы;

− рассмотреть средства и методы защиты от *SQL*-инъекций;

− разработать клиент-серверное приложение, которое бы давало пользователям возможность получить практический опыт в области *SQL*-инъекций и сфере защиты от них.

В ходе анализа требований к разрабатываемому продукту были рассмотрены и проанализированы существующие разработки и построена таблица их сравнения с обозначением сходств и различий (таблица 1).

1) *Hack The Box*:

*Hack The Box* предоставляет отдельным лицам, предприятиям и университетам инструменты, необходимые им для постоянного улучшения своих возможностей в области кибербезопасности.

Одна из самых масштабных платформ, где на текущий момент доступно 127 уязвимых машин, 65 *CTF*-задач и несколько видов сложных виртуальных лесов *AD*.

2) *Root-Me*:

Веб-сайт, который улучшает хакерские навыки и знания в области кибербезопасности с помощью более 200 хакерских проблем и 50 виртуальных сред. Задания сложные, но интересные.

Является быстрой, доступной и реалистичной платформой для проверки навыков взлома. Начиная от сценариев приложения и заканчивая криптоанализом.

3) *Hacker101*:

*Hacker101* – это бесплатный образовательный сайт для хакеров, управляемый компанией *HackerOne* – одной из ведущих передовых *BugBounty* площадок. Независимо от того, программист вы или опытный специалист по безопасности, у *Hacker101* всегда найдется, что вам показать.

*Таблица 1*

**Сравнение программных продуктов**

| № п/п | Список характеристик | *Hack The Box* | *Root-Me* | *Hacker 101* |
|---|---|---|---|---|
| 1. | Присутствие обучающих материалов | + | – | + |
| 2. | Возможность отслеживания успехов и прогресса | + | + | + |
| 3. | Наличие соревновательной, игровой формы обучения вида *CTF* | + | + | + |
| 4. | Наличие подсказок при решении задач | – | – | + |
| 5. | Возможность совместного решения задач с другими пользователями платформы | + | – | + |
| 6. | Наличие заданий на уязвимость к *SQL*-инъекциям | + | + | – |
| 7. | Наличие заданий ориентированных на взлом и получение данных, отличных от *CTF* | + | + | – |
| 8. | Возможность взаимодействия с различными СУБД при решении заданий | + | + | + |

После обзора и анализа существующих разработок, а так же на основании созданной таблицы было решено о необходимости разработки нового приложения, так как существующие аналоги не обладают всеми необходимыми функциональными возможностями для решения поставленной цели в рамках данной работы.

После анализа существующих аналогов, определения способов решения задач с учётом сформулированной цели было определенно, что для решения поставленных задач приложение должно обладать такими функциональными возможностями, как:

− обеспечение взаимодействия с такими СУБД как *MS SQL* и *MySQL*;

− обеспечение формирования и внедрения различных видов *SQL*-инъекций в *SQL*-запросы, отправляемые в СУБД;

− обеспечение пользователю возможности просмотра и анализа примеров средств и методов защиты *SQL*-запросов от *SQL*-инъекций;

− обеспечение получения подсказок пользователем при решении задач;

− обеспечение учёта и отслеживания прогресса пользователя в ходе выполнения заданий;

− обеспечение возможности получение практических навыков в области *SQL*-инъекций и методов защиты от них.

Перечисленные пункты отражают основную часть необходимых возможностей, которые должно предоставлять разрабатываемое приложение. Их реализация обеспечит решение поставленных задач, а также позволит пользователям получить навыки в области *SQL*-инъекций и сфере защиты от них. Используя полученные знания и навыки, программисты и будущие разработчики, будут в состоянии защитить свои приложения и формирующиеся в них *SQL*-запросы от инъекций.

Процесс разработки приложения является очень трудоёмкий, как в плане времени, так и в плане ресурсов будь то денежные или человеческие. Поэтому перед тем как приступать к нему, необходимо чётко построить план действий и определить требования к разрабатываемому продукту [3].

Так, для того чтобы определить, через какие этапы необходимо будет пройти при разработке, была построена контекстная диаграмма разработки приложения при помощи *CASE*-средства *BPwin* (рис. 1).

Рассмотрев данную диаграмму, можно увидеть, что для того, чтобы разработать приложение, необходимо проанализировать требования, предъявляемые к функциям, реализуемым приложением, ресурсы и бюджет, выделенный на разработку.

Так же необходимо определиться с выбором средства разработки и системой управления базами данных. И только с учётом всех пунктов можно создать функционирующее приложение, удовлетворяющее всем требованиям.

Для более глубокого анализа действий, протекающих во время процесса разработки, была сделана декомпозиция, представленная на рисунке 2.

На этой диаграмме можно увидеть, что разработка приложения состоит из нескольких основных этапов: проектирование БД, создание БД, кодирование, тестирование, каждый из которых начинается после завершения предыдущего. После завершения одного из этапов в следующий этап передаётся какой-то результат предыдущего, который используется во время выполнения следующего этапа.

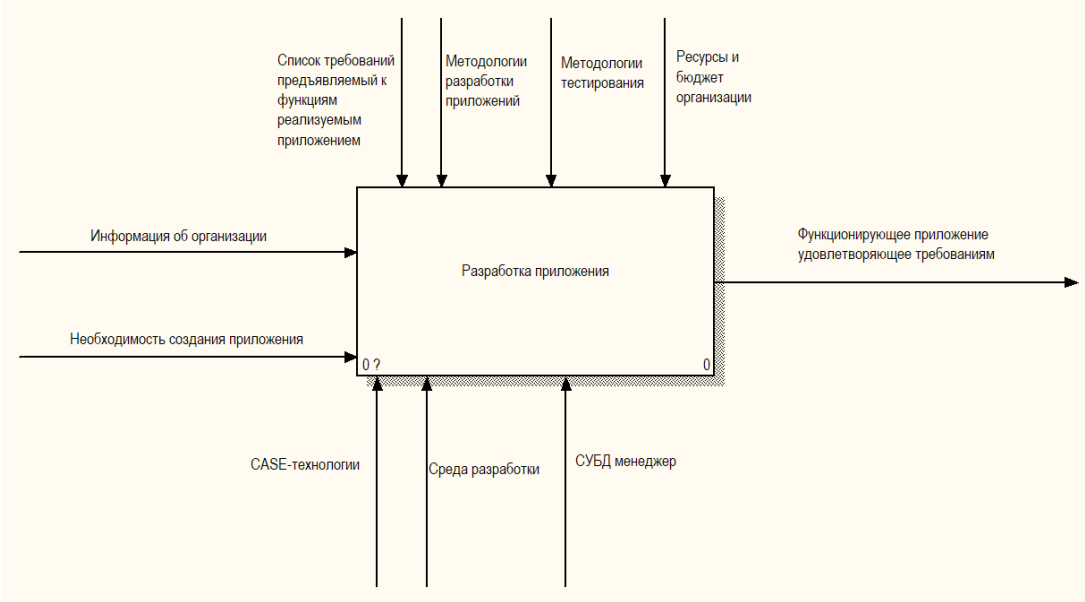


Рис. 1. Контекстная диаграмма разработки приложения

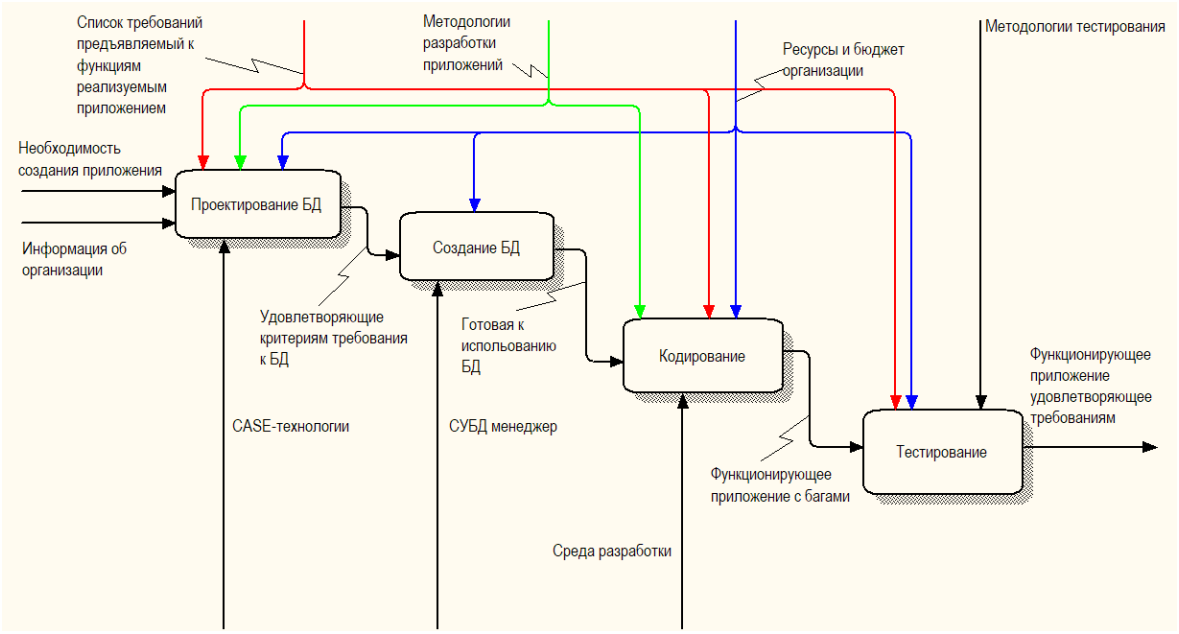


Рис. 2. Декомпозиция процесса разработки приложения

Проанализировав и изучив требования, предъявляемые к приложению, а также полученные диаграммы, было определено, через какие этапы необходимо пройти, чтобы разработать приложение, реализующее поставленную цель.

При проектировании структуры и архитектуры приложения было построено и рассмотрено множество схем и диаграмм.

Так, для определения того, какими возможностями будет обладать пользователь при взаимодействии с программным продуктом, была построена диаграмма вариантов использования (рис. 3).

Как можно увидеть на данной диаграмме (рис. 3), в роли пользователей будут выступать как разработчики, пентестеры, так и обычные пользователи, желающие получить знания и навыки в области *SQL*-инъекций и сфере защиты от них. При взаимодействии с приложением пользователю необходимо пройти авторизацию, после чего он сможет выбирать из предложенного списка задание, ознакомиться с условием выбранного задания, провести тесты на уязвимости и решить его с применением *SQL*-инъекций, а при возникновении трудностей получить подсказки по решению задания. Также пользователь может просматривать свой текущий прогресс решения заданий с возможностью выбора нового задания.

Для схематичного отображения процессов, возникающих при функционировании приложения, была построена диаграмма последовательностей (рис. 4).

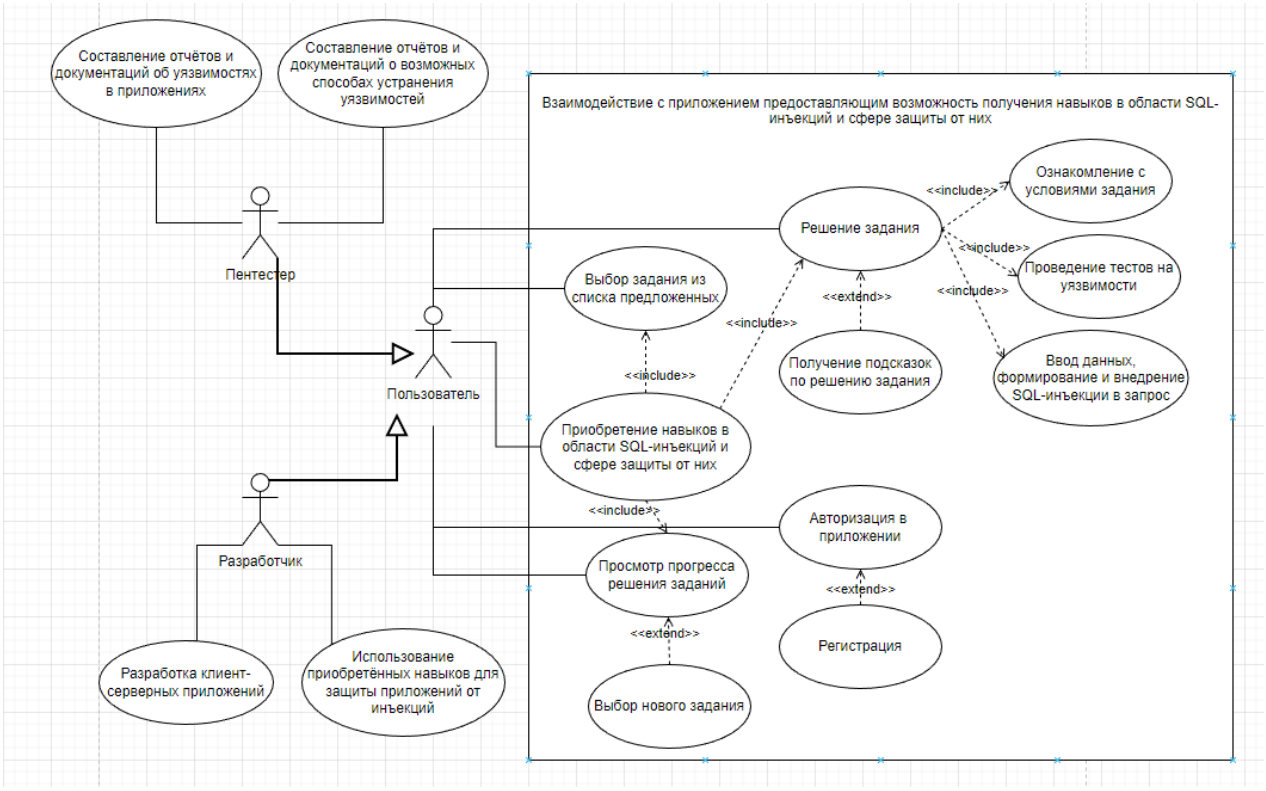


Рис. 3. Диаграмма вариантов использования

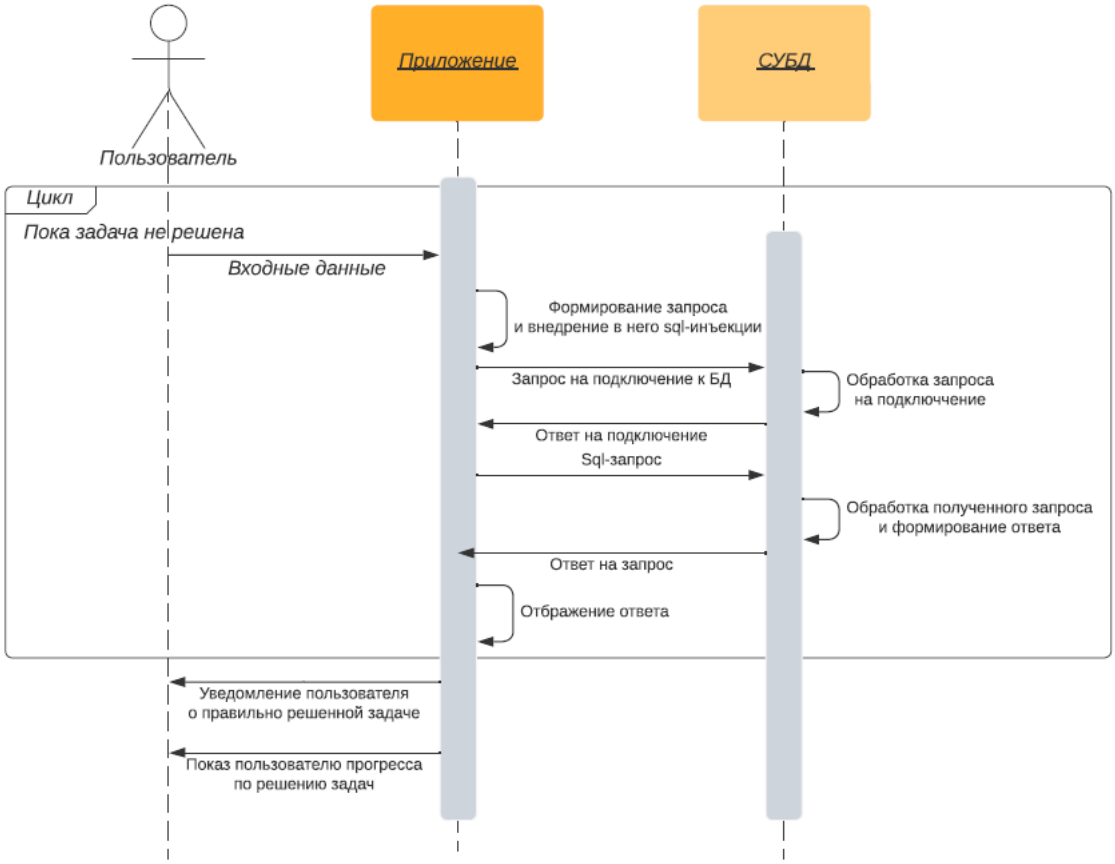


Рис. 4. Диаграмма последовательности

На данной диаграмме (рис. 4) можно увидеть, что программа будет представлять из себя клиент-серверное приложение. Взаимодействуя с клиентской частью, пользователь формирует *SQL*-инъекции, которые внедряются в *SQL*-запросы, отправляющиеся в СУБД для работы с базой данных. СУБД обрабатывает полученные запросы и возвращает результаты обратно в клиентское приложение, где они отображаются пользователю в удобной для работы форме с помощью графического интерфейса.

Так, для более детального анализа процессов, происходящих во время функционирования приложения, а так же переходов приложения из одного состояния в другое, была построена диаграмма состояний и действий (рис. 5).

Как можно увидеть на данной диаграмме (рис. 5), при функционировании приложение будет получать входные данные от пользователя, после чего будет отправляться запрос к СУБД на подключение к БД. При ошибке будет происходить проверка соединения с СУБД, и отправляться повторный запрос. После получения данных о подключении будет формироваться *SQL*-запрос с внедрением в него *SQL*-инъекции введённой пользователем.

Затем сформированный запрос будет отправляться к СУБД, и будет приниматься ответ, после чего полученные данные будут отображаться пользователю с помощью графического интерфейса. В зависимости от того, удовлетворяет ли полученный результат условиям выполнения задачи или нет, будет выводиться подсказка по решению текущей задачи или будет показываться уведомление об успешном решении задачи. После чего пользователь сможет перейти либо к следующей задаче, либо к просмотру всего списка задач с указаниями текущего прогресса пользователя.

Проанализировав полученные диаграммы, можно с точностью определить, какие средства и методы реализации необходимо использовать для обеспечения бесперебойной работы процессов, возникающих при функционировании приложения.

Так, учитывая требования к функциональным возможностям приложения и проанализировав взаимодействие процессов, происходящих во время работы приложения, было решено, что для реализации данного приложения необходимо использовать следующие средства и методы:

1) СУБД *Microsoft SQL Server* и *MySQL* для создания базы данных проекта.

*SQL Server* занимает одно из ведущих мест среди систем для проектирования, создания и обработки баз данных.

Он является надежной базой данных для любых целей, обладает возможностью расширения по мере заполнения информацией без существенного уменьшения быстродействия операций с записями в многопользовательском режиме.

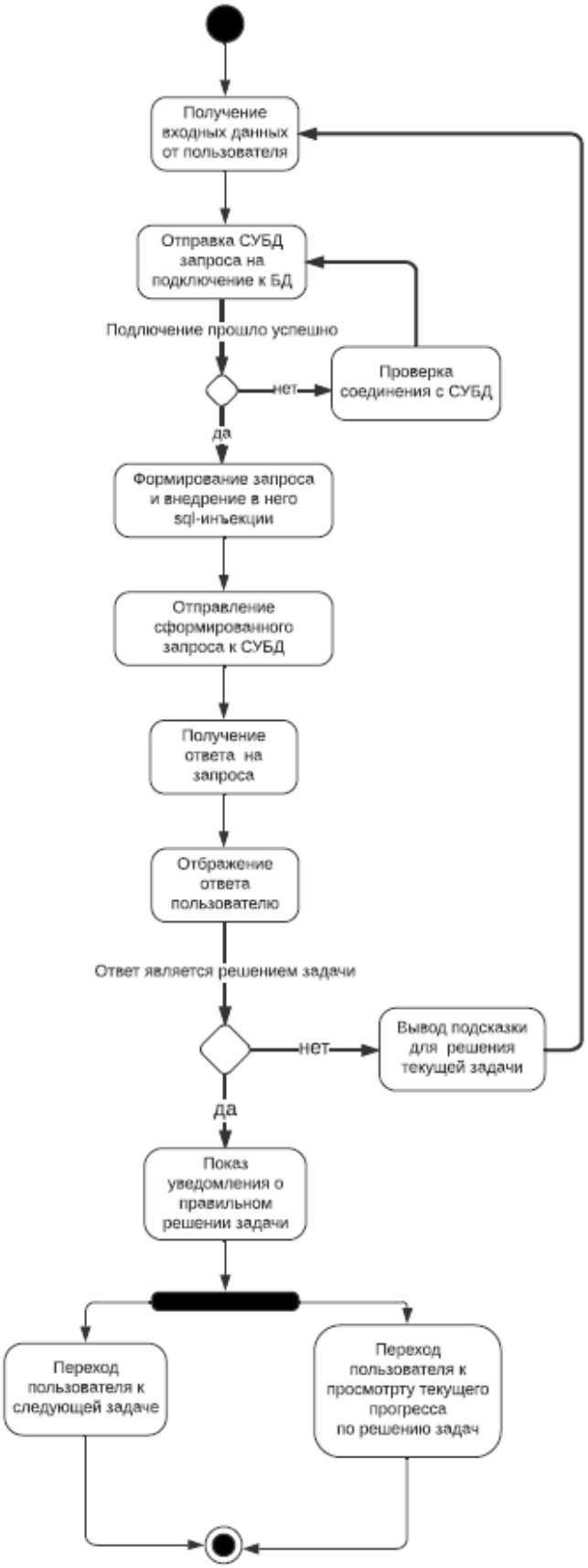


Рис. 5. Диаграмма состояний и действий

Обеспечивается максимальная безопасность. Данные находятся под защитой от несанкционированного доступа за счет интеграции сетевой безопасности с сервером безопасности. Поскольку безопасность организуется на уровне пользователя,

пользователи могут иметь ограниченный доступ при взаимодействии с данными, тем самым осуществляется защита данных от модификации или поиска, с помощью указания доступа на уровне пользовательских привилегией. Кроме того, с данными, хранящимися на отдельном сервере, сервер работает как шлюз, обеспечивая защиту от несанкционированного доступа.

*SQL Server* обрабатывает запросы от пользователей и отправляет пользователю только результаты запроса. Таким образом, минимизируется количество информации передаваемой по сети. Это улучшает время отклика и устраняет узкие места в сети. Также это позволяет использовать *SQL Server* в качестве идеальной базы данных для сети интернет.

*MySQL* – свободная система управления базами данных. Распространяется под *GNU General Public License* и под собственной коммерческой лицензией. Она является решением для малых и средних приложений. Обычно данная СУБД используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать *MySQL* в автономные программы.

С помощью *MySQL* пользователь может легко получит доступ из программы к системе управления базами данных, на каком бы языке программирования она не была написана.

2) *C#* в качестве языка программирования.

Выбранный язык является языком программирования высокого уровня. Он вобрал в себя все лучшее из таких популярных языков как *C++, Visual Basic, Java* и *Object Pascal. C#* обеспечивает быструю разработку приложения, но в то же время позволяет писать достаточно эффективный код.

Язык программирования *C#* характеризуется следующими двумя *преимуществами:*

− он спроектирован и разработан специально для применения с платформой *.NET Framework*. Таким образом, он позволяет наиболее полно и эффективно использовать все возможности, предоставляемые данной платформой;

− это объектно-ориентированный язык программирования. При его создании был использован весь более чем двадцатилетний опыт разработки объектно-ориентированных языков программирования [4].

3) *Visual Studio* в качестве среды разработки.

Выбор среды разработки не составил каких-либо проблем, так как достойных представителей разработки на *C#* не так уж и много. Самой продвинутой, как очевидно, является *Visual Studio*. Кроме того, от удобств *Visual Studio* в качестве удобного использования *NuGet*'ов, огромного списка библиотек из коробки, *TFS* и многого другого сложно отказаться.

Данная среда позволяет разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, и обладает следующими *преимуществами:*

− широкие возможностей различных языков программирования;

− наличие всех необходимых средств и инструментов для разработки приложений любой сложности;

− удобство в использовании и гибкость в настройке;

− создание эффективного кода;

− возможность использования готовых библиотек классов.

Изучив требования, предъявляемые к функциональным возможностям приложения, а так же рассмотрев и проанализировав диаграммы, полученные в ходе проектирования, можно увидеть, что для отправки *SQL*-запросов необходима база данных. Исходя из этого, было решено разработать базу данных магазина сотовой связи так, чтобы она была наибольшим образом приближена к реально существующим аналогам для того, чтобы поместить пользователя при выполнении заданий в среду, максимально приближенную к реальной.

При проектировании базы данных была построена логическая схема для отображения структуры БД (рис. 6).

Как видно на схеме база имеет различные таблицы, в которых хранятся данные по тому или иному аспекту.

Так, например, можно увидеть, что сущность «Клиенты» содержит атрибуты: *id*_клиента, Фамилия, Имя, Отчество, Дата рождения, Адрес, Телефон. В качестве ключевого можно принять атрибут *id*_клиента, так как это уникальное числовое значение, соответствующее каждому клиенту.

А сущность «Аккаунты сотрудников» содержит атрибуты: *id*_аккаунта, Логин, Пароль, Права доступа, *id*_сотрудника. В качестве ключевого можно принять атрибут *id*_аккаунта, так как это уникальное числовое значение, соответствующее каждому аккаунту.

Так же было решено, что для обеспечения функционирования системы авторизации необходимо создать базу данных приложения.

В ходе её проектирования также была построена логическая схема для отображения структуры БД (рис. 7).

В ходе проектирования приложения были описаны основные алгоритмы, применяемые в нём. Основным алгоритмом в приложении, предоставляющим возможность получения навыков в области *SQL*-инъекций и сфере защиты от них является алгоритм решения заданий. Он был изображен на следующей блок-схеме (рис. 8).

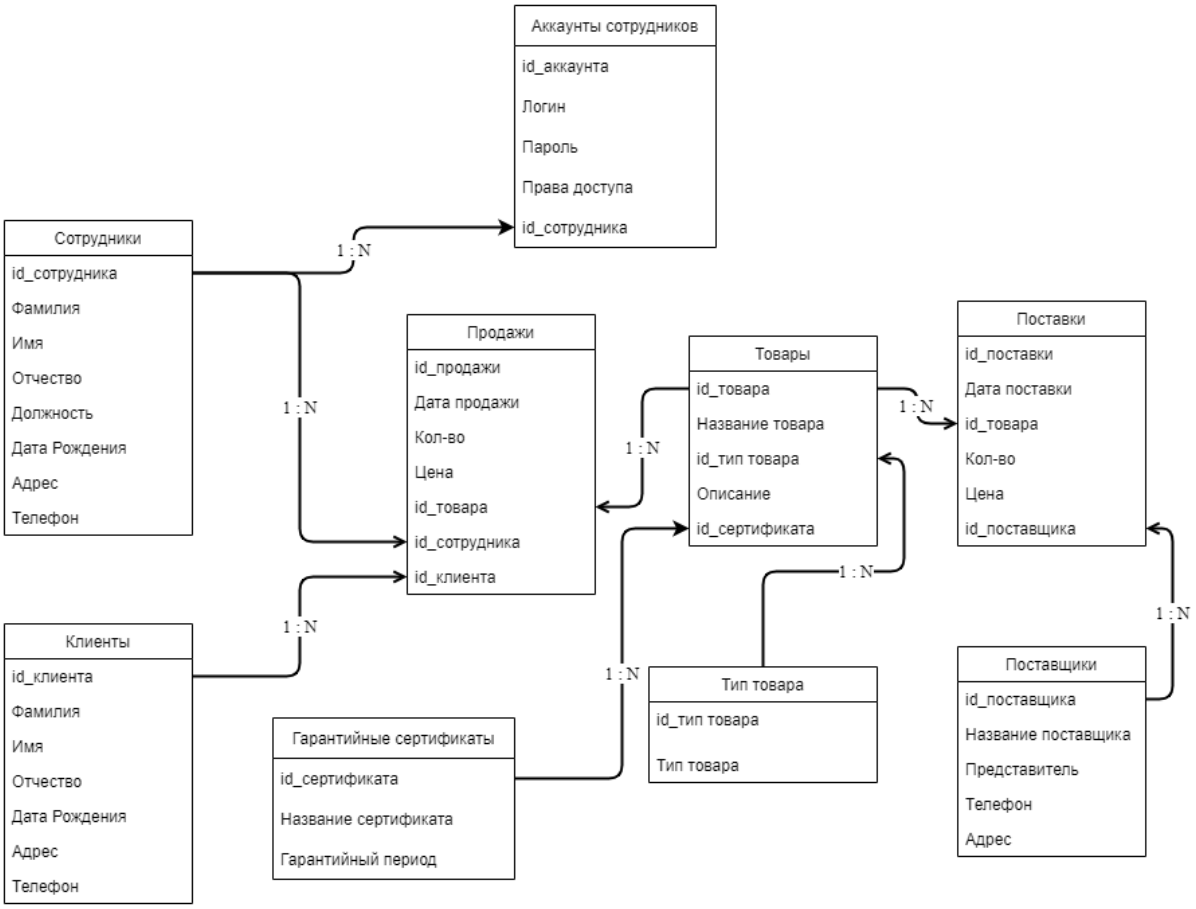


Рис. 6. Логическая схема базы данных магазина сотовой связи

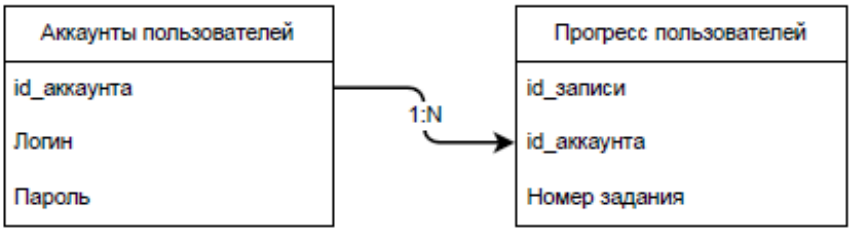


Рис. 7. Логическая схема базы данных приложения

Как видно на данный схеме (рис. 8), для того, чтобы приступить к решению задания, пользователю необходимо выбрать его из списка предложенных заданий. После чего будет показана предварительная информация о выбранном задании с указаниями того, что предстоит сделать для его выполнения. Затем необходимо нажать кнопку «Приступить», после чего в новом окне появится форма с полями, в которые пользователю необходимо вводить данные для решения текущего задания.

При решении заданий пользователю необходимо будет вводить в поля *SQL*-инъекции и по нажатию кнопки «Отправить» они будут внедряться в *SQL*-запросы отправляемые к СУБД.

СУБД, в свою очередь, будет обрабатывать полученные запросы и возвращать ответы. Полученные ответы будут отображаться пользователю в графическом интерфейсе приложения, после чего ответ, удовлетворяющий критериям решения задания, необходимо будет ввести в соответствующее поле и нажать кнопку «Принять».

Если введенный ответ не будет являться решением, то пользователю будет предложена подсказка по решению текущего задания. Иначе, если ответ будет являться решением, то будет выведено соот-

ветствующее уведомление об успешном решении задания, и пользователь будет возвращён в окно выбора заданий.

Таким образом, в ходе анализа требований и проектирования данного приложения была определена цель и выделены задачи для её решения; проанализированы и исследованы существующие разработки; определены требования к функциональным возможностям приложения. В соответствии с данными требованиями в ходе проектирования структуры и архитектуры программного продукта были построены и описаны различные виды диаграмм; спроектированы структуры баз данных приложения и построены логические схемы для их схематического отображения. Так же был описан основной алгоритм, применяемый в приложении, и построена его блок-схема.

На основе результатов, полученных в ходе проделанной работы, возможна реализация приложения, предоставляющего возможность получения навыков в области *SQL*-инъекций и сфере защиты от них. С помощью него программисты и будущие разработчики смогут получить знания и навыки, с помощью которых они смогут защитить свои приложения и формирующиеся в них *SQL*-запросы от инъекций.

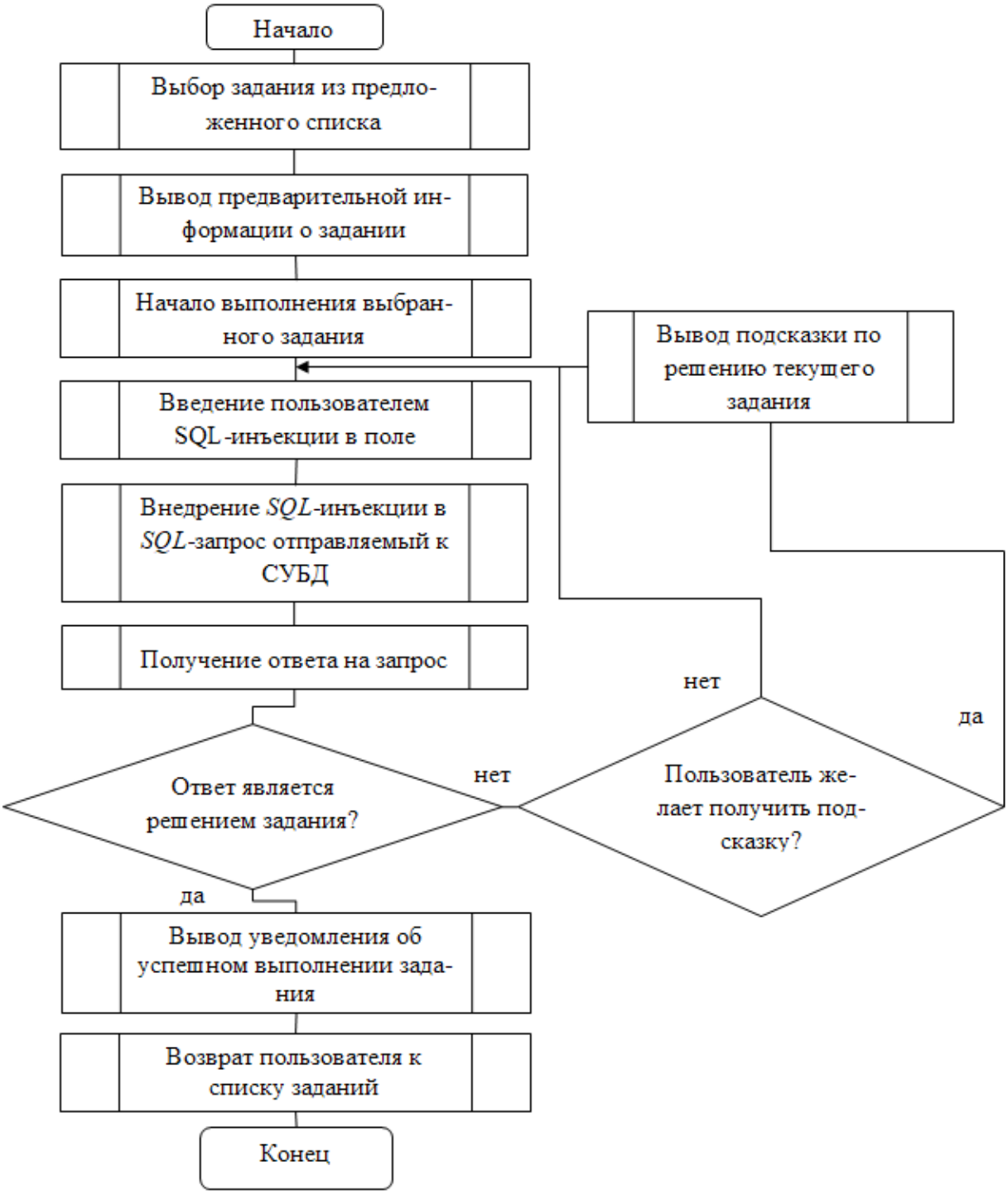


Рис. 8. Блок-схема алгоритма решения заданий

**Цитированная литература**

1. Нестеров, С. А. Основы информационной безопасности: учебное пособие / С. А. Нестеров. – 3-е изд., стер. – Санкт-Петербург : Лань, 2017. – 324 с.

2. Кузмичев, И. С. Методы защиты данных на прикладном уровне веб ориентированных информационных систем / И. С. Кузмичев // Научные достижения и открытия современной молодёжи. Пенза: «Наука и Просвещение»; (ИП Гуляев Г. Ю.), 2017. – С. 94–96.

3. Вигерс, К. Разработка требований к программному обеспечению, пер. с англ. – Москва : Русская Редакция, 2011 [Электронный ресурс]. – URL : http://ab.kh.ua/books/Vigers%20Karl%20-%20Razrabotka%20Trebovany%20K%20Programmnomu%20obespecheniu.pdf (дата обращения: 22.03.2024).

4. Албахари, Д., Албахари Б. *C#* 6.0. Справочник. Полное описание языка. – Москва : Вильямс, 2018 [Электронный ресурс]. – URL : http://lib.maupfib.kg/wp-content/uploads/2015/12/dzhozef_albakhari_ben_alb akh-ari__c_5_0_karmann.pdf (дата обращения: 22.03.2024).

УДК: 621. 01. 001; 004.42

# ЦИФРОВАЯ ТРАНСФОРМАЦИЯ В АГРОПРОМЫШЛЕННОЙ СФЕРЕ: ИННОВАЦИОННЫЕ РЕШЕНИЯ С LABVIEW И SOLIDWORKS

А. С. Царюк, бакалавр
В. П. Юсюз, ст. преподаватель

***Аннотация.*** *В данной статье анализируются преимущества цифровизации в сельском хозяйстве и демонстрируются, как эти инструменты могут быть эффективно применены для оптимизации производственных процессов, улучшения качества продукции и снижения затрат.*

***Ключевые слова:*** *программное обеспечение, SolidWorks, LabVIEW, компьютерная модель, виртуальное прототипирование.*

В современном мире технологический прогресс неуклонно внедряется в различные отрасли, внося новые возможности и изменяя привычные способы работы. Одной из областей, которая активно вступает в эпоху цифровой трансформации, является агропромышленный сектор. Стремительное развитие информационных технологий, аналитики данных и автоматизации процессов позволяет сельскохозяйственным предприятиям существенно повысить эффективность производства, улучшить качество продукции и оптимизировать расходы.

В данной статье уделяется внимание ключевым аспектам цифровой трансформации в агропромышленной сфере, сосредотачиваясь на инновационных решениях, предоставляемых программными продуктами *LabVIEW* и *SolidWorks*. Эти инструменты не только обеспечивают возможность создания высокотехнологичного оборудования и систем контроля, но и способствуют интеграции данных, управлению ресурсами и оптимизации процессов производства. Далее рассматривается, как именно эти технологии изменяют агропромышленную отрасль и какие преимущества они приносят для улучшения ее конкурентоспособности и устойчивости.

Агропромышленная сфера является одной из важнейших отраслей экономики, обеспечивающей продовольственную безопасность и удовлетворение потребностей населения. В современном мире, где стремительно развиваются технологии, использование программного обеспечения для автоматизации и оптимизации процессов в агропромышленной сфере становится неотъемлемой частью деятельности.

Мехатронные системы в агрономии представляют собой комплексное сочетание механических, электронных, программных и управляющих компонентов, разработанных для автоматизации и оптимизации сельскохозяйственных процессов. Используя принципы робототехники, мехатронные системы могут выполнять различные задачи, такие как посев семян, полив растений, уборка урожая, определение состояния почвы и диагностика заболеваний растений.

В последние годы мехатроника и программное обеспечение для проектирования, такие как *Solidworks*, находят всё большее применение в агрономии, что открывает новые перспективы для разработки инновационных систем и устройств.

*SolidWorks* – это трехмерная система компьютерного моделирования и автоматизированного проектирования, которая позволяет создавать сложные 3D-модели и подробные технические чертежи. Это программное обеспечение широко используется в агропромышленной сфере для разработки и оптимизации механических систем, создания инновационных сельскохозяйственных машин и оборудования.

С помощью *SolidWorks* можно проектировать и моделировать такие агротехнические решения, как тракторы, комбайны, и другие сельскохозяйственные механизмы (рис. 1).

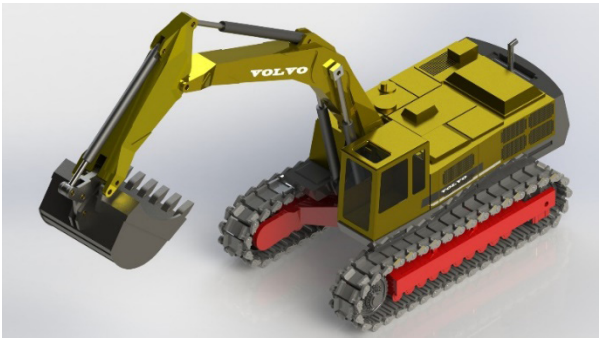


Рис. 1. Компьютерная модель экскаватора EC650, спроектированного в среде *SolidWorks*

*Solidworks*, мощное программное обеспечение для трехмерного моделирования и проектирования, что в настоящее время является незаменимым инструментом для разработки мехатронных систем в агрономии. С его помощью можно создавать и оптимизировать детали и механизмы, проводить анализ прочности и динамики, моделировать взаимодействие с окружающей средой и проверять работоспособность системы в виртуальной среде.

В данной статье рассматривается два ключевых инструмента – *LabVIEW* и *SolidWorks* – и их применение в данной отрасли (рис. 2).

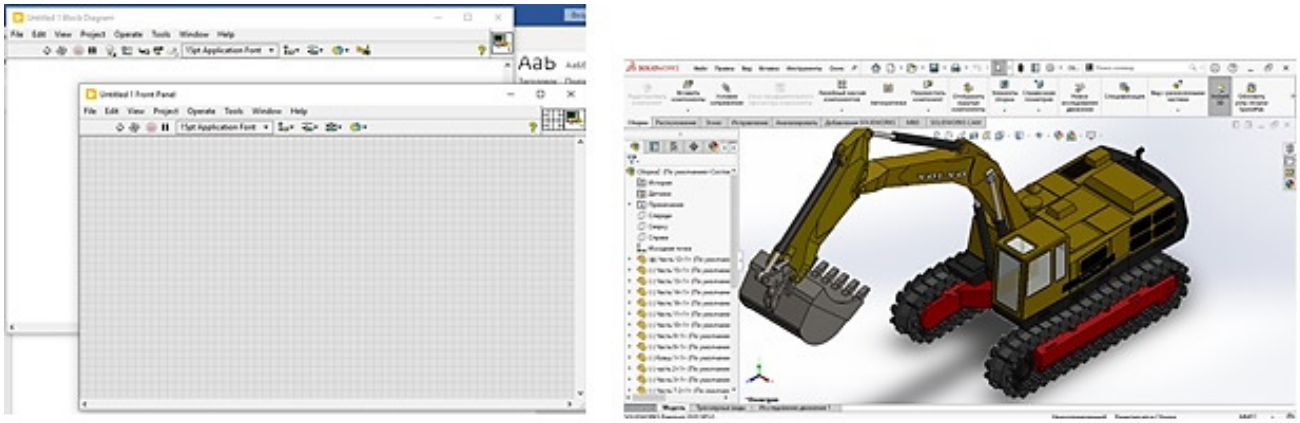

Рис. 2. Среды проектирования, исследования и управления компьютерными моделями (слева *NI LabVIEW,* справа *SolidWorks*)

*LabVIEW (Laboratory Virtual Instrument Engineering Workbench)* – это графическая среда программирования, разработанная для автоматизации и контроля научных и промышленных процессов. Она широко используется в агропромышленном секторе для контроля качества, управления различными агротехническими системами, а также для мониторинга и анализа данных.

Одним из основных преимуществ *LabVIEW* является его удобный интерфейс, основанный на графическом представлении программы в виде блок-схемы (рис. 3).

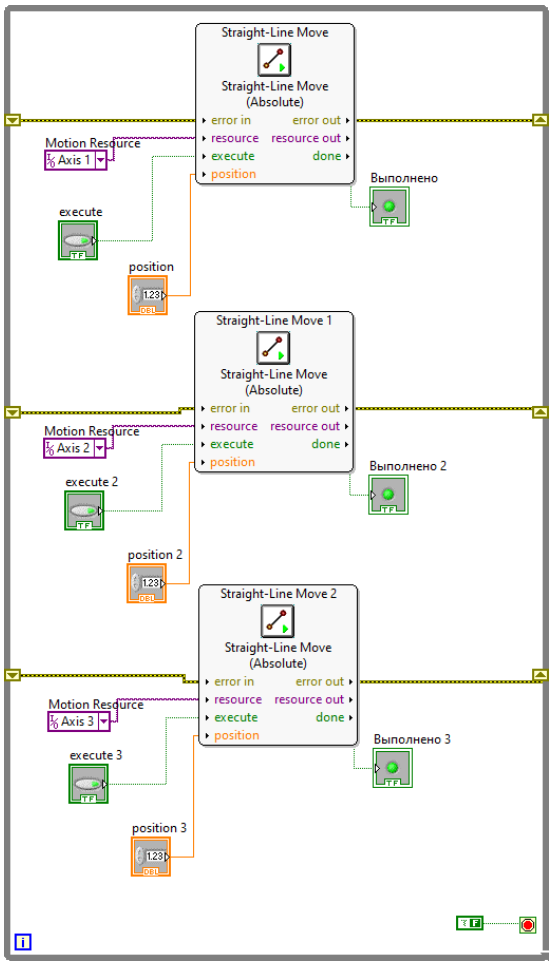


Рис. 3. Блок-схема управления 3D моделью экскаватора

Это позволяет разработчику легко создавать и модифицировать программы, что особенно полезно в динамичной агропромышленной среде. Кроме того, наличие большого количества готовых модулей и библиотек делает *LabVIEW* универсальным инструментом для любых задач в агропромышленности.

Использование совместно *LabVIEW* и *SolidWorks* позволяет синхронизировать (и взаимодействовать) между собой агротехнические системы и механизмы. Например, с помощью *LabVIEW* можно создать программное обеспечение для мониторинга и управления работой трактора, в то время как *SolidWorks* позволит разработать оптимальное механическое решение для его исполнения (рис. 4).

Интеграция программных платформ *LabVIEW* и *SolidWorks* является существенной составляющей в развитии и механизации сельского хозяйства. Объединение этих двух мощных инструментов открывает широкие возможности для повышения эффективности и автоматизации процессов, связанных с сельским хозяйством.

С помощью *LabVIEW* и *SolidWorks* возможно создание сложных графических интерфейсов, включающих трехмерные модели оборудования и симуляции работы на них. Это значительно упрощает восприятие и взаимодействие с системой для операторов и персонала сельскохозяйственных предприятий.

Благодаря интеграции можно реализовать мониторинг и управление различными параметрами сельскохозяйственных систем в реальном времени. Это позволяет оптимизировать процессы и достигать максимальной производительности при минимальных затратах.

Преимущества виртуального прототипирования *LabVIEW* и *SolidWorks* для сельскохозяйственной отрасли очень заметны. Позволяя работать в виртуальной среде, эти программные решения снижают риски и затраты на разработку, ускоряют процессы тестирования и испытаний новых реше-

ний. Они обеспечивают возможность детального анализа и оптимизации систем до их физической реализации, что позволяет предотвратить ошибки и улучшить работу разработанных изделий. Кроме того, использование виртуального прототипирования позволяет предусмотреть возможные проблемы и сложности, а также осуществить процессы обучения сельскохозяйственного персонала до внедрения новых технологий.

Результаты исследования показали, что применение программных продуктов *LabVIEW* и *SolidWorks* в агропромышленной сфере имеет значительный потенциал для улучшения производственных процессов и повышения эффективности предприятий. В частности, выявлены следующие положительные моменты:

1. Увеличение производительности оборудования благодаря автоматизации производственных процессов с использованием *LabVIEW*.

2. Улучшение качества продукции за счет внедрения виртуальных прототипов и анализа конструкций с помощью *SolidWorks*.

3. Сокращение времени на разработку и внедрение инноваций благодаря гибкости и масштабируемости разработки программного обеспечения с помощью *LabVIEW*.

Таким образом, использование *LabVIEW* и *SolidWorks* в агропромышленной сфере открывает широкие перспективы для повышения эффективности и развития данной отрасли. Эти инструменты позволяют ускорить процессы, снизить затраты, повысить качество продукции и улучшить условия труда сельскохозяйственных работников. Внедрение современных технологий и программного обеспечения в агропромышленную сферу помогает справиться с современными вызовами и обеспечить устойчивое развитие данной важной отрасли экономики.

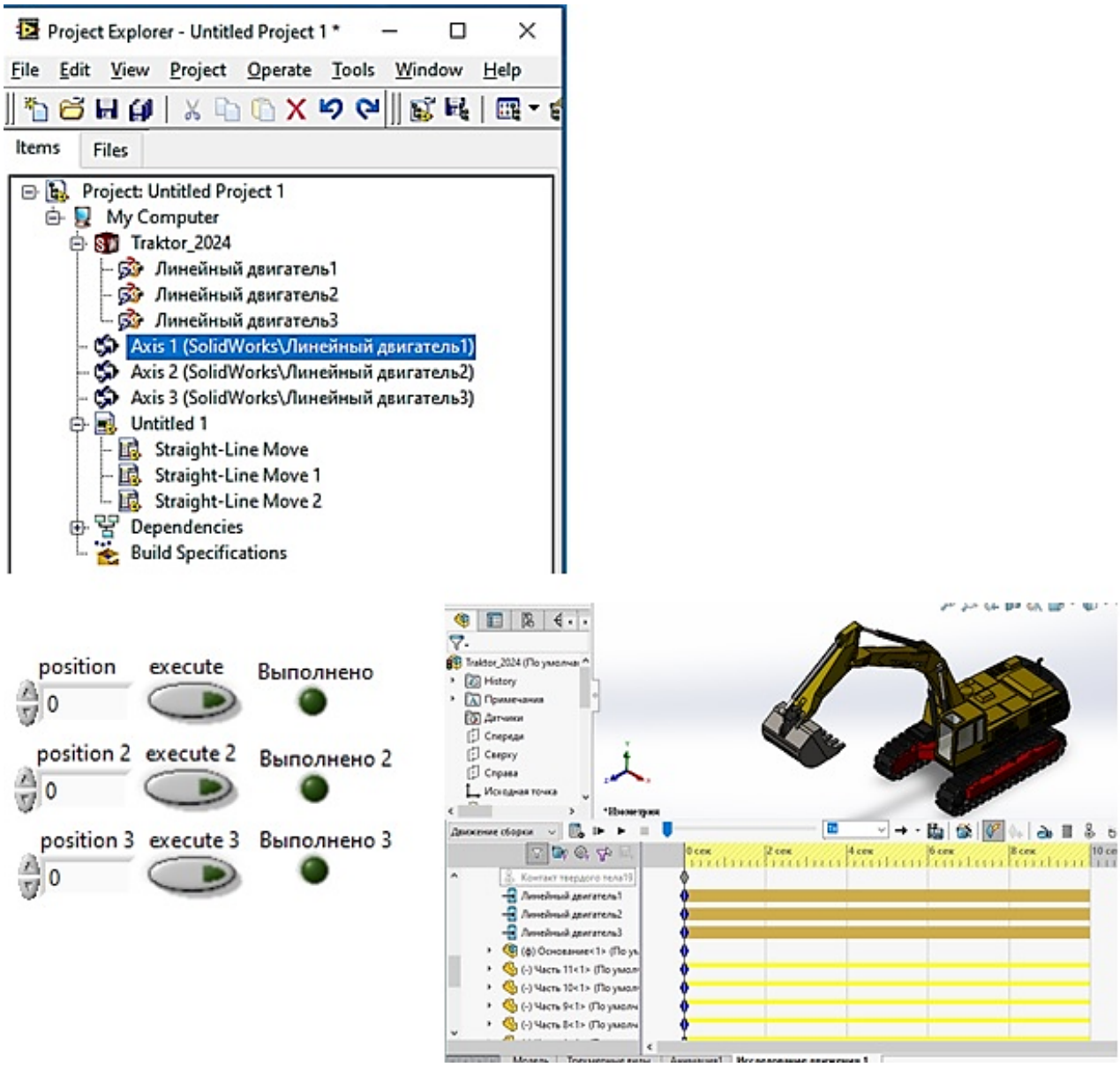


Рис. 4. Дерево элементов управления и его пульт с управляемой моделью.

**Цитированная литература**

1. Алямовский, А. А. *SolidWorks Simulation*. Как решать практические задачи / А. А. Алямовский. – Санкт-Петербург : БХВ-Петербург, 2012. – 445 с.

2. Васильев, А. С. Основы программирования в среде *LabVIEW* / А. С. Васильев, О. Ю. Лашманов. – Санкт-Петербург : Университет ИТМО, 2015. – 82 с.

3. Джеффри, Т. *LabVIEW* для всех: пер. с англ. Клушин Н. А. / Т. Джеффри. – Москва : ДМК Пресс; ПриборКомплект, 2005. – 544 с,: ил.

4. Зиновьев, Д. Основы проектирования в *Solidworks* 2016 / Д. Зиновьев. – Москва : Студия *Vertex*, 2017. – 277 с.

5. Лукинских, С. В. Компьютерное моделирование и инженерный анализ в конструкторско-технологической подготовке производства : учебное пособие / С. В. Лукинских. – Екатеринбург : Уральский федеральный университет имени первого Президента России Б. Н. Ельцина, 2020. – 168 с.

УДК [378.147.3: 004] (478)

# СОВРЕМЕННЫЕ РЕШЕНИЯ ПО ИСПОЛЬЗОВАНИЮ РАСТИТЕЛЬНЫХ ОТХОДОВ В ПЕРЕРАБАТЫВАЮЩЕЙ ПРОМЫШЛЕННОСТИ ПМР

И. Р. Яковец, магистрант
В. Г. Звонкий, доцент
И. В. Яковец, доцент

***Аннотация.*** *В статье предложены конкретные меры для стимулирования взаимосвязанного развития отрасли переработки отходов в органические удобрения и отрасли производства органической продукции. Особое внимание уделено технологиям переработки отходов в органические удобрения и процессу гранулирования с целью получения био-пеллет, био-гранул.*

***Ключевые слова:*** *вторичные ресурсы, отходы переработки кукурузы, био-пеллеты, био-гранулы, процесс гранулирования.*

Любое производство всегда на выходе имеет, помимо готового товарного продукта, и побочный продукт – отходы [1]. Рачительное обращение с отходами, в частности пищевой индустрии и предприятий пищевой отрасли, создаёт благоприятные условия снижения их уровня и возможности для получение дополнительных экологически чистых видов товарной продукции, а также увеличения прибыли.

Оптимальный вариант решения проблемы отходов – переход к экономике замкнутого цикла. Такая модель не только производства, но и потребления, базируется на возобновлении ресурсов, когда отходы не накапливаются, а используются вторично в производстве. Разработка методов вторичного использования отходов позволяет в значительной степени снизить затраты производства [2].

Сегодня в промышленное производство всё шире внедряются инновационные технологии, предусматривающие использование отходов в качестве вторсырья. Такой подход позволяет частично решить проблему утилизации отходов и способствует ресурсосбережению. Проблемой, с которой сталкиваются предприятия, в том числе пищевой промышленности, создающей значительную финансовую нагрузку, является утилизация отходов и содержание мусорных полигонов. В результате повседневной деятельности и осуществления технологических процессов на предприятиях пищевой отрасли образуются фактически не отходы, а ценное сырье, которое следует использовать вторично для изготовления экологически чистых био-продуктов. Рециклинг отходов и дополнительная продукция из вторичных ресурсов обеспечивает расширение сырьевой базы при снижении издержек на единицу конечной продукции [3].

В силу сложившейся непростой международной ситуации для экономики Приднестровской Молдавской Республики период, начиная с 2022 года, был весьма сложным. На деятельности хозяйствующих субъектов внешнеполитические процессы сказались негативным образом: от усложнения логистики поставки товаров, необходимых для производства продукции (например, семенной материал и удобрения, медицинские препараты, запасные части, ГСМ и ряд других) до проблемы реализации продукции на экспорт (блокировка банковских операций, цена реализации) [4]. Стоимость, например, удобрений, применение которых является важной составляющей для получения достойного урожая, выросла в 2,5–3 раза с 2022 года по сравнению с предыдущим периодом, при том, что доставка не была гарантирована.

Таким образом, ситуация вследствие событий в соседних государствах доказала правильность курса правительства на самообеспеченность Приднестровья продукцией отечественного производства, обозначенной в Стратегии развития Республики на 2019–2026 годы [4], что требует, в том числе, развития, технической и технологической модернизации сельского хозяйства. Переход на инновационный путь внедрения биотехнологий – одно из приоритетных направлений государственной политики Приднестровья в сфере АПК. В сложных внешнеполитических и климатических условиях инновационные подходы позволяют поддерживать рост производства основных видов сельскохозяйственной продукции, обеспечивать продовольственную безопасность Республики и повышение качества производимой продукции [7].

Достигнутые показатели урожайности разных культур явились следствием внедрения современных ресурсосберегающих технологий [7].

В настоящее время в земледелии особую актуальность имеют проблемы, как:

1. Постоянно растущие затраты из-за постоянного роста цен минеральных удобрений и средств защиты растений [7].

2. Потеря плодородных почвенных ресурсов и ухудшение экологической обстановки окружающей среды из-за процессов эрозии почв, чрезмерной минерализации гумуса [7].

В таких сложных условиях аграрии должны обеспечить уменьшение производственных затрат и снижение себестоимости, рост урожайности и повышение качества производимой продукции, расширенное воспроизводство плодородия почвы и сохранение окружающей среды. Решение этих задач возможно только при переходе на ресурсосберегающие технологии [7].

*Сберегающее земледелие* – это изменение способа основной обработки почвы. Система, включающая в себя:

1. Внедрение безотвальной и мелкой обработок почвы с сохранением растительных остатков и измельченной соломы в верхнем слое или на поверхности почвы. Главная цель при этом – стабильные экономически выгодные уровни урожайности при низкой себестоимости [7].

2. Обеспечение круглогодичной мульчи из растительных остатков не менее 3–5 т/га, использования сидератов и измельченной соломы на удобрение, что позволяет уменьшить потери гумуса и пополнить запасы элементов питания для сельхозкультур в почве. Наиболее эффективно удешевление производимой продукции и повышение плодородия почвы достигаются за счёт мульчирования и сидерации [7].

Возможным решением этих проблем может стать внесение в поверхностные почвенные слои гранул или пеллет, изготовленных их вторичных ресурсов – отходов растительного происхождения, образующихся в больших количествах в результате переработки сельхозпродукции, например, кукурузы [7].

Несмотря на неблагоприятные погодные условия (засуха и высокие температуры в летний период), в стране удалось получить неплохой урожай по важнейшим сельскохозяйственным культурам, в том числе включая кукурузу [4].

Основные посевы сахарной кукурузы находятся в Слободзейском районе республики, которой было собрано порядка 25 тысяч тонн. Часть данной культуры перерабатывается путём заморозки, как в початках, так и в зерне, а часть – идёт на консервирование [5]. В таблице 1 представлены данные по сбору кукурузы в ПМР за 5 лет [4, 5].

ОАО «Завод консервов детского питания» в г. Тирасполе осуществляет переработку сахарной кукурузы в среднем от 10 до 25 тысяч тонн в последние годы, производя порядка 5 тысяч тонн только замороженной кукурузы. После переработки кукурузы как на зерно, так для заморозки и консервирования, остаётся большое количество отходов: стебли, листья, стержни початков и другие. Основную часть отходов составляют стержни початков: в 1 тонне зерновой кукурузы содержится 200 кг стержней, т. е. они составляют как минимум пятую часть. В таблице 2 приведены статистические данные по количеству образования стержней початков кукурузы в ПМР за 5 лет в период с 2018 по 2022 годы.

Данные, представленные в таблице 2 по количеству образования отходов переработки кукурузы в виде стержней початков, показывают, что в среднем ежегодно образуется порядка 25 тысяч тонн. Лишь малая часть этих отходов используется на корм животным. В большей части стержни початков вывозятся на мусорные полигоны.

На рисунке 1 представлена динамика образования отходов переработки кукурузы – стержней початков за период 2018–2023 гг.

*Таблица 1*

**Производство кукурузы в Приднестровской Молдавской Республике за период 2018–2022 годы, тонн**

| Наименование культуры | 2018 г. | 2019 г. | 2020 г. | 2021 г. | 2022 г. | 2023 г. |
|---|---|---|---|---|---|---|
| Зерновые и зернобобовые культуры – всего, в т. ч. | 480 126,6 | 471 031,9 | 133 587,7 | 644 814,5 | 350 081,9 | 536 500 |
| кукуруза на зерно | 134 176,7 | 133 805,5 | 36 039,4 | 184 152,7 | 108 855,0 | 120 000 |
| сахарная кукуруза | 18 686,8 | 24 259,5 | 25 052,4 | 17 630,1 | 14 196,5 | 15 800 |
| Итого кукурузы: | 152 863,5 | 158 065 | 61 084,4 | 201 782,8 | 123 051,5 | 135 800 |

*Таблица 2*

**Количество образования стержней початков кукурузы Приднестровской Молдавской Республики за 2018–2022 годы, тонн**

| Наименование культуры | 2018 г. | 2019 г. | 2020 г. | 2021 г. | 2022 г. | 2023 г. |
|---|---|---|---|---|---|---|
| кукуруза на зерно | 26 835,3 | 26 761 | 7 207,9 | 36 830,5 | 21 771 | 24 000 |
| сахарная кукуруза | 3 737,4 | 4 852 | 5 010,5 | 3 526 | 2 839,3 | 3 160 |
| Итого: | 30 572,7 | 31 613 | 12 218,4 | 40 356,5 | 24 610,3 | 27 160 |

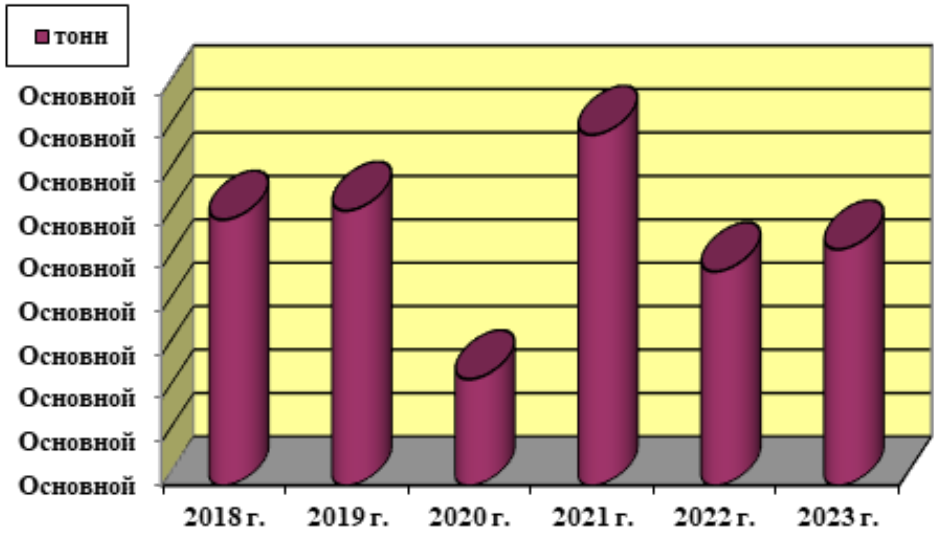


Рис. 1. Образование отходов переработки кукурузы – стержней початков за период 2018–2023 гг., в тоннах

Стержни кукурузных початков можно рассматривать как ценный побочный продукт сельского хозяйства и консервной промышленности, биомассу, являющуюся фактически органическим вторичным сырьём для производства био-пеллет и био-гранул, которые могут быть использованы не только как био-топливо, но и как сырьевой товарный продукт для производства био-удобрений, био-пластиков, бумаги и прочего.

Производство прессованных материалов позволяет утилизировать различные виды отходов, получить экологически чистые био-гранулы и био-пеллеты для высококалорийного биотоплива, кормов, био-удобрений и других материалов, обеспечить основному производству статус малоотходного и экологически «чистого», снизить затраты на перевозку и хранение топлива по сравнению с древесными отходами (кусковыми и мягкими) или дровами, сеном и соломой, силосом, повысить культуру производства и получить дополнительную прибыль от реализации био-гранул и био-пеллет, обеспечить условия для создания новых рабочих мест, сократить уровень отходов производства и потребления [8, 9].

Одним из способов утилизации образующихся на предприятии побочных ресурсов является их брикетирование, пеллетирование и гранулирование [9]. Применение технологии прессования (гранулирования) позволяет получать пеллеты или гранулы заданного размера, формы и необходимых физико-механических характеристик, что сокращает потери при транспортировке, хранении и переработке, а также улучшает показатели дальнейшего использования [9].

Гранулированные био-продукты имеют ряд существенных преимуществ перед рассыпными: изменяется структура продукта, снижается бактериальная загрязненность, улучшаются условия хранения, уменьшается объем и т. д. [9].

Однако, существует категория материалов, гранулирование которых в традиционных аппаратах затруднительно, связано с высокими энергозатратами или снижением качества пеллет и гранул [9]. В связи с этим разработка процесса гранулирования органического сырья ставит перед собой задачи:

– проведение анализа существующих технологий и технических средств для гранулирования органического сырья;

– исследование основных закономерностей процесса гранулирования органического сырья с различными физико-механическими характеристиками;

– изучение влияния технологических параметров процесса гранулирования органического сырья на эффективность и выявление рациональных и энергозатратных режимов его работы.

Технология производства, например, древесных гранул предполагает реализацию следующих процессов: измельчение крупных кусков в рубительной машине и удаление из сырья инородных включений; измельчение высушенного материала в мельнице с отделением и возвратом крупных частиц; обработка мелкодисперсного сырья в смесительной камере гранулятора паром или водой и его прессование в гранулы; охлаждение гранул с удалением и возвратом в технологический цикл отделившейся крошки, транспортировка на склад [9].

Технология переработки отходов кукурузных початков отличие в отличие от древесного сырья позволяет исключить операции измельчение крупных кусков в рубительной машине и сушки перед измельчением стволов початков, т. к. их влажность не превышает 8–10 %, что сокращает длительность производственного цикла, сокращает энергозатраты на производство био-пеллет и био-гранул из такого сырья, снижает их себестоимость [10].

Были произведены литературный обзор и патентный поиск с целью изучения степени влияния

различных факторов на процесс гранулирования сырья растительного происхождения, результаты которого представлены в таблице 3 [10].

Сложное взаимное влияние ряда физических процессов, протекающих одновременно в пресс-грануляторе, усложняет интерпретацию влияния каждого параметра на процесс получения топливных гранул из сырья растительного происхождения [9]. В современной научной литературе существует пробел в понимании взаимодействия между некоторыми параметрами гранулирования и их влиянием на результаты процесса, что требует проведения дополнительных исследований.

Анализ результатов исследований показал, что предварительная тепловая обработка и увлажнение растительного сырья, а также его состав и размеры частиц, являются факторами, которые оказывают наибольшее влияние на качество кормовых и биотопливных пеллет и гранул. Что касается значений этих параметров, то ученые считают оптимальным при гранулировании растительного сырья размер частиц от 1 до 3 мм. Содержания влаги для биотопливных пеллет должно быть в пределах от 10 до 15 %, для кормовых гранул – 15–20 % [10].

*Таблица 3*

**Степень влияния факторов на процесс гранулирования сырья растительного происхождения и качество гранул [10]**

| № п/п | | Доля влияния фактора на процесс гранулирования и качество гранул, % | | |
|---|---|---|---|---|
| | | К. Мурамацу | А. Клименко, А. Гущева-Митропольская | М. Иванов |
| 1. | Тепловая обработка (кондиционирование) | 44 | 20 | 30 |
| 2. | Влажность сырья | 16 | – | 2 |
| 3. | Содержание жиров | 9 | – | – |
| 4. | Размер частиц сырья | 1 | 20 | 10 |
| 5. | Состав сырья | – | 40 | 45 |
| 6. | Конструктивные параметры пресс-гранулятора | – | 15 | 10 |
| 7. | Охлаждение | – | 5 | – |
| 8. | Температура | – | – | 3 |
| 9. | Прочие факторы | 31 | – | – |

На качество пеллет и гранул, помимо этого, большое влияние оказывает содержание жиров в исходном сырье. Сырье оптимально должно содержать от 2 до 10 % жира [10].

Следует отметить существенный недостаток вторичного сырья из стержней початков кукурузы для получения качественных био-пеллет и био-гранул – отсутствие в их составе жиров, необходимых для реализации процесса гранулирования. Соответственно, для получения качественного продукта необходима разработка рецептуры сырьевой смеси с добавлением жиросодержащих компонентов. В качестве такой добавки, например, может выступить кофейный жмых или кофейная гуща, из которой сегодня изготавливают в ряде стран, например, в Англии топливные брикеты и пеллеты [3].

Регулирование в процессе гранулирования температуры и давления позволяет добиваться получения качественных гранул. Повышение давления в диапазоне 20–200 МПа приводит к увеличению прочности гранул, а температура матрицы около 100 °С является оптимальной для получения плотных качественных гранул из растительного сырья [10].

Результаты анализа показывают, что в получении качественных гранул и пеллет из сырья растительного происхождения также важную роль играют конструктивные параметры пресс-гранулятора. В частности, исполнение входной части фильеры матрицы в форме сужающегося конуса способствует снижению энергоемкости и давления гранулирования, а увеличение соотношения длины канала фильеры к его диаметру *L/D* экспоненциально увеличивает давление гранулирования и его энергоемкость. Значимыми параметрами являются конструкция прессующих роликов и их зазор с матрицей, определяющим протекание процесса гранулирования, однако установление оптимальных значений этих параметров требует проведения дополнительных исследований [10].

Таким образом, несмотря на значительный объем информации, накопленной к настоящему времени, можно заключить, что необходимы дальнейшие исследования взаимодействия факторов, влияющих на качество процесса гранулирования и самих гранул из сырья растительного происхождения. Сложное взаимообуславливающее воздействие много-

численных физических процессов, одновременно протекающих в пресс-грануляторе, усложняет интерпретацию влияния каждого из параметров на процесс гранулирования. В следствие указанных причин исследователи по-разному оценивают вклад разных факторов в получение качественных пеллет и гранул из различных видов сырья, что требует уточнения.

**Цитированная литература**


1. Современные решения по переработке растительных отходов / Е. Б. Аносова, С. М. Ляшенко, И. В. Сусоева, Т. Н. Вахнина. Научные и образовательные проблемы гражданской защиты. – 2017. – № 4 (35). – С. 116–120. – [Электронный ресурс]. URL : https://cyberleninka.ru/article/n/sovremennye-resheniya-po-pererabotke-rastitelnyh-othodov/viewer.

2. Использование вторичного сырья: Краткий курс лекций для студентов / составление: А. В. Бороздина // ФГБОУ ВПО Саратовский ГАУ. – Саратов, 2015. – 79 с.

3. Производство пеллет из кофейного жома – делаем деньги из отходов / Гран Дмитрий. – [Электронный ресурс]. – URL : https://truckmix.ru/ articles/ proizvodstvo-pellet-iz-kofeinogo-joma---delaem-dengi-iz-othodov.

4. Доклад об итогах деятельности Министерства сельского хозяйства и природных ресурсов Приднестровской Молдавской Республики в сфере агропромышленного комплекса за 2022 год. О. И. Дилигул. – [Электронный ресурс]. – URL : https://ecology-pmr.org/files/doklad/2022-03-01-doklad-kollegia-apk.pdf.

5. Международная научно-практическая конференция (2020; Тирасполь). Доклад об итогах деятельности в сфере агропромышленного комплекса за 2020 год. «Селекция, семеноводство и технологии возделывания сельскохозяйственных культур» / Селекция, семеноводство и технологии возделывания сельскохозяйственных культур: Доклады международной научно-практической конференции: посвященной 90-летию со дня основания института. 10 апреля 2020 / отв. ред.: А. В. Гуманюк. – Тирасполь: *Eco-TIRAS,* 2020 (*Tipogr. «Arconteh»*). – 376 р.: *fig., tab. Antetit.:* Гос. учреждение «Приднестр. НИИ сел. хоз-ва». – *Text parțial: lb. engl. – Rez.: lb. engl.* – [Электронный ресурс]. – URL : https:// ecology-pmr.org/files/doklad/2022-03-01-doklad-kollegia-apk.pdf.

6. В Республике завершается уборка сельхозкультур. Новости Прднестровья. Экономика. 21.11.23. – [Электронный ресурс]. – URL : https:// novostipmr.com/ru/news/23-11-21/v-respublike-zavershaetsya-uborka-selhozkultur.

7. Инновации – основа развития сельского хозяйства / Ф. С. Цыбульский // Торгово-промышленная палата Приднестровской Молдавской Республики. https:// tiraspol.ru/news/innovatsii-osnova-razvitiya-selskogo-hozyaystva/.

8. Любов, В. К. Повышение эффективности энергетического использования биотоплив: учебное пособие / В. К. Любов, С. В. Любова. – Архангельск : Изд-во ОАО «Солти», 2010. – 496 с.

9. Попов, А. Н. Исследование процесса производства древесных гранул с целью повышения эффективности их энергетического использования: дисс. на соискание уч. степени канд. техн. наук / А. Н. Попов. – ФГАОУ ВО «Северный (Арктический) федеральный университет имени М. В. Ломоносова». – Архангельск, 2010. – 151 с.

10. Брагинец, С. В. Влияние различных параметров на процесс гранулирования растительного сырья и качество гранул (обзор) / С. В. Брагинец, О. Н. Бахчевников, К. А. Деев. – Аграрная наука Евро-Северо-Востока. – 2023. – 24 (1). – С. 30–45. – [Электронный ресурс]. – URL : https://cyberleninka.ru/ article/n/vliyanie-razlichnyh-parametrov-na-protsess-granulirovaniya-rastitelnogo-syrya-i-kachestvo-granul-obzor/viewer (дата обращения: 27.04.24.).

Периодическое издание

**АЛЬМАНАХ ТЕОРЕТИЧЕСКИХ И ПРИКЛАДНЫХ МЕЖОТРАСЛЕВЫХ ИССЛЕДОВАНИЙ**
**ежегодный научно-практический (методический) журнал**
**Физико-технического института ПГУ им. Т. Г. Шевченко**

Редактор: Запорожан Ю. А.
Компьютерная верстка: Запорожан Ю. А.

ИЛ № 06150. Сер. АЮ от 21.02.02.
Подписано в печать 25.06.24. Формат 60*84/8.
Усл. печ. л. 7,1. Электронное издание. Заказ № 391.

Подготовлено в Издательстве Приднестровского университета
3300, г. Тирасполь, ул. Мира, 18