УДК 37.02:004

СОВРЕМЕННЫЕ ПРАКТИКИ ПРИ СОЗДАНИИ ЭОР ДЛЯ ПОДДЕРЖКИ ПРЕПОДАВАНИЯ IT-ДИСЦИПЛИН ВЫСШЕЙ ШКОЛЫ

[1]Жуков Николай Николаевич, канд. физ.-мат. наук, доцент кафедры информационных технологий и электронного обучения;
[2]Мельников Фёдор Владиславович, магистрант

[1,2]Российский государственный педагогический университет им. А.И. Герцена, Санкт-Петербург, Россия, e-mail: nzhukov@herzen.spb.ru, balrun.dev@gmail.com

Прогрессивные методы DevOps, используемые при создании программных продуктов, проникают в разные области, включая образование. В статье описаны возможности применения этого подхода для улучшения процесса создания и поддержки электронных образовательных ресурсов и веб-сервисов, использующихся в образовательном процессе, указывается конкретный стек технологий, инструментов, которые могут быть использованы при создании собственного ЭОР, приведены примеры конфигурационных файлов для автоматизации процесса развертывания.

Методология DevOps (акроним, образованный из частей слов «development» и «operations») предполагает реализацию различных практик, подходов, применение специализированных инструментов, направленных на сокращение релизного цикла программного продукта, а также особой инженерной культуры внутри команды, разрабатывающей продукт. DevOps, как и некоторые другие IT-тренды, постепенно внедряется в другие области деятельности, в том числе и в образовательную. Данная методология напрямую влияет как на область разработки электронных образовательных ресурсов, так и, на более высоком уровне абстракции, может видоизменять жизненный цикл проектирования и реализации непосредственно образовательного контента. Рассмотрим подробнее, каким образом практики DevOps возможно применять при разработке ЭОР и образовательных веб-приложений.

В настоящее время для всех участников образовательного процесса очевидно положительное влияние применения электронных образовательных ресурсов на процесс обучения. В [1] отмечается, что представление учебного материала в электронном виде средствами современных электронных технологий даёт возможность ориентироваться на психологические особенности каждого обучающегося.

М. В. Махмутова, Е. И. Сеничева, О. А. Акимова выделяют три этапа разработки электронного образовательного ресурса: концепция, реализация и внедрение ЭОР в учебный процесс [2]. Реализация первого и третьего этапов зависит, прежде всего, от образовательных задач и дидактических принципов разработки. Второй этап значительно зависит от применяемых технологий.

Одним из видов электронных образовательных ресурсов является веб-сайт. Это один из способов размещения учебной информации в сети Интернет. Существует большое количество технологий и методологий разработки веб-сайтов. Например, современным инструментом разработки электронных образовательных ресурсов является генератор статических сайтов [3].

Электронные образовательные ресурсы создаются с применением современных технологий, часто требующих от разработчика знания программирования. В связи с этим актуальным является применение в образовании современных практик разработки программных продуктов.

Внедрение современных практик и методологий позволяет упростить разработку ресурса. Одной из таких практик является DevOps.

DevOps – это, в первую очередь, практики и инструменты, направленные на повышение эффективности и обеспечение возможности выпуска продукта в любой момент. Это обеспечивается за счёт интеграции и автоматизации повторяющихся операций.

Для применения DevOps требуются определённые компоненты, среди которых можно выделить следующие: система управления версиями (version control system), система непрерывной интеграции и непрерывного развёртывания (CI/CD), система контейнеризации.

DevOps может применяться при разработке электронных образовательных ресурсов, учебных приложений.

Система контроля версий является основным компонентом DevOps. Это программное обеспечение, предназначенное для работы с изменяющейся информацией. Система позволяет хранить разные версии одного документа.

Самой распространённой системой контроля версий является Git. Среди сервисов с реализацией Git можно выделить GitHub, GitLab, GitFlic.

При разработке электронных образовательных ресурсов, выполненных в виде статических сайтов, помимо практик DevOps целесообразно применять подходы Jamstack.

Jamstack представляет собой методологию, подход к разработке веб-сайтов. Он позволяет быстро создавать и обслуживать статические веб-сайты. Свойствами подхода является предварительная сборка страниц и их хранение в сети доставки содержимого CDN – content delivery network. Типичным примером сервиса, предоставляющего CDN, является Cloudflare. Ключевым компонентом Jamstack является генератор статических сайтов.

Существуют разные подходы к созданию веб-сайтов с точки зрения применения системы контроля версий. Важным является выбор модели ветвления: Git Flow, GitHub Flow, GitLab Flow.

Git Flow реализует основные принципы разработки продуктов в системе контроля версий с учётом релизных циклов (см. Рис. 1). Применение данной модели позволяет выпускать стабильные версии ресурса. К недостаткам можно отнести сложность схемы работы, а также наличие задержки публикации готовой версии.

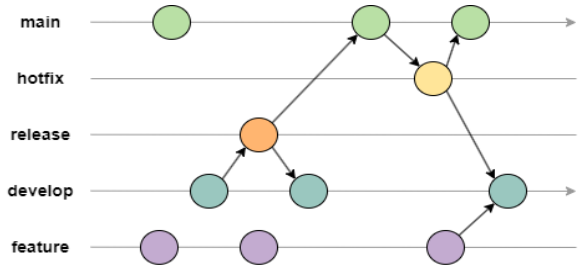

Рис. 1. Модель Git Flow

GitHub Flow реализует основные принципы разработки без учёта релизных циклов. При применении данной модели готовая версия продукта публикуется сразу. Модель является более простой и прозрачной для разработчика. Недостатком модели является то, что она требует высокой степени автоматизации процессов тестирования и публикации.

GitLab Flow объединяет преимущества рассмотренных моделей. Она схожа с GitHub Flow и при этом позволяет контролировать выпуски продукта, как при применении Git Flow.

Более подробно данные модели рассмотрены в статье [4].

Применение рассмотренных моделей может быть избыточным при разработке небольших ресурсов, но имеет большое значение, если ресурс разрабатывается совместно с обучающимися или другими преподавателями.

Важной составляющей DevOps является система CI/CD – непрерывной интеграции и непрерывного развёртывания. При этом исходный код хранится в системе контроля версий. Если используется Git, то разработка может производиться в соответствии с одной из рассмотренных моделей ветвления.

Различные операции, такие как сборка (для статического сайта), тестирование, публикация ресурса, часто выполняются в ручном режиме. Автоматизация повторяющихся операций позволяет уменьшить время, необходимое для внесения изменений.

При применении CI/CD производятся автоматизированные сборки, сайт или приложение обновляется короткими итерациями, публикация производится в автоматическом режиме. Стадии выполняются после загрузки изменений в репозиторий системы контроля версий.

Существует несколько основных способов размещения сайтов:

- использование сервисов бесплатного хостинга (например GitHub Pages, Bitbucket Pages, GitLab Pages);
- использование услуг провайдеров платного хостинга (например, Яндекс.Облако, Amazon Web Services, Google Cloud, Microsoft Azure и др.);
- размещение сайта на собственном сервере или виртуальной машине (например, используя такие технологии как Docker, Kubernetes и др.).

Из указанных выше способов можно выделить второй способ, в него отдельной категорией можно включить так называемые shared-хостинги (когда на одном физическом сервере располагается несколько сайтов). При выборе третьего способа, указанного выше, собственный сервер может быть виртуальным или реальным. В этом случае обычно упоминают аббревиатуры VPS и VDS.

Процесс публикации сайта и деятельность, которая должна быть выполнена в рамках этого процесса, во многом зависит от выбранного способа.

Однако, необходимо заметить, что объединяющим фактором вне зависимости от способа размещения будет использование механизма, при котором сборка производится автоматически с использованием технологии GitHub Actions или аналогов (например, GitLab CI/CD), которая предоставляет возможность запускать скрипты, тесты и задачи, помогающие поддерживать качество кода и ускорять выпуск новых версий приложений. С помощью GitHub Actions можно легко масштабировать процесс разработки и управлять рабочими процессами для различных платформ и окружений.

Основные принципы сборки и публикации статических сайтов у различных сервисов (например, таких как GitLab CI/CD и GitHub Actions) схожи, опишем эти принципы на примере GitHub Actions.

Для публикации требуется выполнить следующие шаги:

1. Создание репозитория на GitHub, если он отсутствует
2. Загрузка туда набора исходных файлов для сборки (для каждого генератора статических сайтов набор этих файлов может быть разным).
3. Создание файла рабочего процесса (workflow) в репозитории. Файл рабочего процесса содержит инструкции для GitHub Actions, которые определяют, какие задачи должны выполняться и при каких условиях.
4. Настройка файла рабочего процесса. В файле рабочего процесса указываются команды, которые должны выполняться при определенных событиях, таких как отправка изменений в репозиторий, создание запроса на операцию pull-request или создание нового тега.
5. Компиляция и упаковка статического сайта. Этот этап обычно включает сборку кода, минификацию и конкатенацию файлов, а также создание архива или пакета.
6. Тестирование сайта. На этом этапе выполняются автоматические тесты для проверки работоспособности сайта и качества кода.
7. Развертывание сайта. После успешной сборки и тестирования сайт может быть развернут на сервере или в облачной среде. GitHub Actions предоставляет интеграцию с различными сервисами, такими как Netlify, Vercel и AWS Amplify, для упрощения процесса развертывания.
8. Мониторинг и логирование. GitHub Actions также предоставляет возможности для мониторинга и логирования выполнения рабочих процессов, что позволяет отслеживать состояние сборок.

Этапы, начиная с 5 по 7 выполняются обычно внутри одного задания, в изолированной среде или «контейнере».

Сам по себе рабочий процесс или workflow может состоять из одного или нескольких «работ» или «заданий» (“jobs” в терминологии GitHub Actions), выполнение которых может быть запланировано создателем файла как последовательно, так и параллельно для заданий, не зависящих друг от друга (например, минификация html и компиляция css).

Каждое задание, в свою очередь, может быть разбито на еще более небольшие составляющие, называемые «шаги» («steps» в терминологии сервиса), которые реализуются в изолированном пространстве. Описанный выше рабочий процесс представлен на схеме (см. Рис. 2).

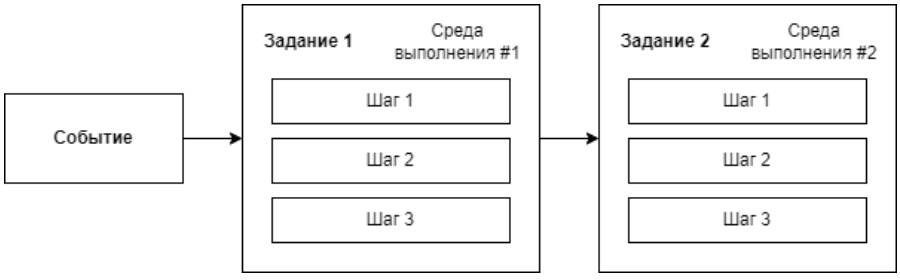

Рис. 2. Схема устройства рабочего процесса в среде Github Actions

Весь процесс описывается внутри файла YAML-файла, который состоит из нескольких секций, каждая из которых описывает определенный этап рабочего процеса или задачу, сам файл также должен находится в репозитории с содержимым развертываемого сайта. Каждая задача предполагает наличие какого-либо процесса, который будет выполняться, события, при котором это происходит и описания задачи, которые включаются в этот процесс. Имя процесса задается с помощью атрибута “name” и через двоеточие идет само название. Атрибут “on” определяет при каком событии задача запускается. Если необходимо срабатывать в нескольких ситуациях, то они указываются через запятую внутри квадратных скобок (например, при указании “[push, fork]” событие запускается при каждом push-обновлении или при создании ответвления репозитория).

Необходимо отметить, что при использовании любого генератора статических сайтов, выполнение этапов рабочего процесса, предполагает реализацию одних и тех же этапов: организацию среды выполнения, непосредственно реализацию сборки и публикацию на сервере.

Организация среды выполнения включает в себя копирование предварительно загруженных в репозиторий исходных файлов сайта в среду выполнения и инсталляцию в этой среде выполнения самого генератора статических сайтов. Как было указано выше, среда выполнения – изолирована, поэтому возможно использовать различные программы-генераторы.

Этап реализации сборки в GitHub Actions аналогичен тому как это происходит при выполнении на локальном компьютере.

Этап публикации на сервере или развертывании (deployment), обычно, требует указания параметров для подключения к серверу (установку ключей на сервер для удаленного подключения) и копирования собранных и готовых к отправке файлов сайта на сервер.

Разделение этапов на отдельные задания не имеет практического значения, поскольку это не только приводит к ненужному повторному копированию, но и увеличивает время сборки. Выбор генератора не является критичным для организации процесса сборки, он зависит от предпочтений и потребностей пользователя. Для сравнительно небольших сайтов, к которым можно отнести, например ЭОР по одной дисциплине, производительность генератора, вероятно, не будет являться значимым фактором и не окажет действия на время процесса сборки и развертывания, в силу того, что этап организации среды исполнения занимает значительно больше времени, чем сам время работы самого генератора статических файлов. Необходимо заметить, однако, что существуют работы, демонстрирующие, что критерий «количество файлов на сайте» имеет положительную корреляцию со скоростью сборки [5]. Время сборки сайтов, состоящих из нескольких страниц, для большинства современных генераторов вне зависимости от количества этапов сборки будет незначительным по сравнению со временем, затраченным на подготовку среды.

Процессы непрерывного развертывания (CI) и непрерывной доставки (CD) в рамках методологии DevOps для сравнительного небольшого статического сайта, в котором отсутствует необходимость тестирования, являются аналогичными для большинства генераторов.

Приведем пример реализации процесса сборки и размещения статического сайта на хостинге, в виде схемы и конкретного YAML-файла для генератора MkDocs с использованием сервиса GitHub Actions. Как было упомянуто выше, для других генераторов статических сайтов (например, Hugo, Pelican, Nikola), этот пример будет аналогичным. Схема сборки представлена на рисунке 3 ниже.

Рис. 3. Схема организации автоматизированной сборки статического сайта с использованием MkDocs

Рассмотрим реализацию приведенной выше схемы организации автоматизированной сборки (см. Рис. 4). Процесс стартует при выполнении операции «push» в репозитории с файлами сайта. Единственная задача “job” указанная в файле с названием “Build and Deploy” выполняется в изолированной среде с операционной системе Ubuntu последней официально доступной в GitHub Actions версии (это указывается в атрибуте “runs-on: ubuntu-latest”). В конце скрипта выполняется копирование собранных файлов внутрь каталога site/ удалённого сервера на хостинга.

```yaml
name: Build and Deploy

on:
  push:

jobs:
  build:
    name: Build and Deploy
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v3

      - name: Python Setup
        uses: actions/setup-python@v4
        with:
          python-version: 3.11.1

      - name: MkDocs Setup
        run: pip install mkdocs

      - name: Build
        run: mkdocs build

      - name: Install SSH key
        run: |
          install -m 600 -D /dev/null ~/.ssh/id_rsa
          echo "${{ secrets.PRIVATE_SSH_KEY }}" > ~/.ssh/id_rsa
          echo "${{ secrets.KNOWN_HOSTS }}" > ~/.ssh/known_hosts

      - name: Deploy
        run: rsync --archive --stats site/ ${{ secrets.REMOTE_DEST }}
```

Рис. 4. YAML-файл с реализацией workflow сборки и публикации сайта

Задача «Build and Deploy», в свою очередь, содержит 6 последовательно выполняемых шагов:

- операция выгрузки всего содержимого репозитория в изолированное окружение (Checkout);
- инсталляция интерпретатора языка Python (Python Setup) версии 3.11.1;
- инсталляция генератора статических сайтов (как указывалось выше, на данном шаге может устанавливаться другой генератор) (MkDocs Setup) с использованием команды pip install (стандарт для установки пакетов в экосистеме Python);
- сборка статического сайта из исходных файлов (обычно, исходными файлами являются текстовые файлы с разметкой markdown) (Build);
- копирование файлов с настройками ssh для удаленного подключения к серверу (Install SSH key);
- непосредственно развертывание сайта из каталога /site, в котором находится преобразованных генератором контент, на хостинг удаленного сервера (Deploy).

Следует отметить, что до выполнения операции развертывания, выполняющегося с помощью дан-ного скрипта, необходимо создать и настроить удаленный сервер: собственный или на мощностях хостинг-провайдера. Этап настройка включает установку ПО с веб-сервером и генерацию ssh-ключа root-пользователя для доступа без пароля на созданный сервер.

Указанные в скрипте с помощью двойных фигурных скобок значения переменных (например, “secrets.PRIVATE_SSH_KEY”, называемые также «переменными окружения») задаются в настройках репозитория и могут быть изменены в любое время. Хорошая практика не фиксировать в скрипте сборки эти параметры явно, поскольку это может снижать уровень безопасности и в итоге – привести к сбоям при выполнении сборки. Вручную эти параметры также не вводятся, поскольку основная идея – автоматизация всего процесса. Необходимо обратить внимание на переменную окружения REMOTE_DEST, в которой должен содержаться полный путь для отправки файлов на хостинг в формате «имя_пользователя@доменное_имя_или_IP_адрес_сервера/путь_к_директории» (например,

“root@78.155.216.235/app”). Часть адреса «путь_к_директории» должна быть указана в настройках веб-сервера как корневой каталог для поиска файлов, которые он будет «отдавать» пользователю в браузере. Копирование реализовано с помощью утилиты rsync с ключами, позволяющими отправлять на удаленный сервер только те файлы, которых на сервере нет или которые изменились по сравнению с уже там существующими.

Программные системы для организации DevOps, такие как GitHub Actions и GitLab CI/CD, позволяют отслеживать процесс выполнения сборок программного обеспечения. Они автоматизируют процесс сборки, тестирования и развертывания приложений, предоставляя подробную информацию о каждом этапе конвейера разработки и позволяют при необходимости повторить весь процесс заново.

При разработке веб-приложений целесообразно применять технологии контейнеризации, которые также рассматриваются в рамках практик DevOps. Они позволяют распространять приложения в виде пакета, выполнение которого не зависит от среды операционной системы. Распространённым решением для контейнеризации и автоматизации развёртывания является Docker. Актуальность его использования объясняется прежде всего: упрощением процесса развертывания и уменьшением вероятности ошибок этого процесса, улучшением изоляции (инкапсуляции) приложения вместе с его зависимостями, что в результате позволяет говорить об улучшении бесперебойной работы приложений и их более простой переносимости на другие платформы.

При использовании практик DevOps при организации деятельности рекомендуется использовать гибкие методологии управления программными проектами. Одними из самых универсальных методологии при разработке и поддержке ЭОР авторы считают методологии Kanban или SCRUM. Следует заметить, что при использовании специально адаптированной для организации образовательного процесса «версии» методологии SCRUM – EduScrum, упрощается процесс разработки ЭОР в целом, при желании преподавателя в этот процесс могут вовлекаться и студенты (например, в рамках практики или написании курсовых проектов).

Очевидно, что в ближайшую перспективу, процесс разработки и обновления ЭОР может быть ускорен за счет использования инструментов машинного обучения (ML) как для генерации практических заданий, тестовых вопросов, элементов дизайна (например, графики, визуализаций).

Важным вопросом является готовность преподавателей к использованию рассматриваемых технологий. С одной стороны, применение средств автоматизации позволяет работать с сайтами и приложениями преподавателям, не имеющим опыта работы с данной технологией. С другой стороны, применение современных практик разработки, как и применение информационных технологий в образовательной деятельности в целом, требует от преподавателя отдельных навыков и компетенций, при этом достаточно специфических.

Актуальным является вопрос подготовки преподавателей к использованию современных технологий для разработки электронных образовательных ресурсов. Например, С. С. Бакулевская предлагает программу повышения квалификации для подготовки педагогических работников к использованию технологий HTML5 [6].

Применение рассмотренных подходов является целесообразным, если применяется генератор статических сайтов, ресурс имеет большой объём, разрабатывается совместно с обучающимися или другими преподавателями в системе контроля версий, для публикации выполняется большое количество рутинных операций, которые возможно автоматизировать.

Авторы считают целесообразной подготовку преподавателей к разработке электронных образовательных ресурсов и веб-приложений с применением практик DevOps.

DevOps – это современный подход для повышения эффективности процессов разработки и эксплуатации программного обеспечения, который может применяться при разработке электронных образовательных ресурсов и веб-приложений. Применение данного подхода способствует повышению эффективности разработки и сопровождения образовательных ресурсов.

СПИСОК ЛИТЕРАТУРЫ

1. Цифровая экосистема педагогического образования. Актуальные вопросы. Достижения. Инновации / А. М. Атаян, Е. А. Барахсанова, К. О. Вехова и др. – СПб: ООО "НИЦ АРТ", 2022. – 148 с.

2. Махмутова М. В., Сеничева Е. М., Акимова О. А. Технология разработки и применения электронных образовательных ресурсов в учебном процессе вуза // Открытое образование. – 2019. – Т. 23. – № 6. – С. 50–58.

3. Мельников Ф. В., Жуков Н. Н. Использование генератора статических сайтов как инструмента методической поддержки образовательного процесса // Современное образование. Традиции и инновации. – 2022. – № 4. – С. 156–161.

4. Giesel S. Git-Flow, GitHub-Flow, Gitlab-Flow and Trunk Based Development explained // Электрон. дан. Режим доступа URL: https://steven-giesel.com/blogPost/ff50f268-c0bf-44d8-a5b8-41554ab50ba8 (дата обращения 26.03.2023).

5. Comparing Static Site Generator Build Times // Электрон. дан. Режим доступа URL: https://css-tricks.com/comparing-static-site-generator-build-times/ (дата обращения 05.12.2022).

6. Бакулевская С. С. Подготовка педагогических работников к использованию технологий HTML5 для разработки электронных образовательных ресурсов // Информатика и образование. – 2019. – № 5. – С. 32–40.

STATE-OF-THE-ART PRACTICES IN THE CREATION OF DIGITAL EDUCATIONAL RESOURCES TO SUPPORT THE TEACHING OF HIGHER SCHOOL IT DISCIPLINES

[1]Zhukov Nikolai Nikolaevich, associate Professor of the Computer science and technological education institute, candidate of physical and mathematical sciences
[2]Melnikov Fedor Vladislavovich, master's student, "44.04.01 Corporate electronic learning" educational program

[1,2]The Herzen State Pedagogical University of Russia, Saint-Petersburg, Russia,
e-mail: nzhukov@herzen.spb.ru, balrun.dev@gmail.com

DevOps software methods used in the creation of software products are being implemented in various fields, including education. The article describes the possibilities of using this approach to improve the process of creating and supporting electronic educational resources and web services used in the educational process, shows a specific stack of technologies and tools that can be used when creating your own educational resource, provides examples of configuration files for deployment process automation.