ООО «ИНСТИТУТ ПОВЫШЕНИЯ КВАЛИФИКАЦИИ ГЛАВПРО»

Обучающий центр дополнительного профессионального образования

ТЕХНИЧЕСКАЯ ДОКУМЕНТАЦИЯ

Описание процессов, обеспечивающих поддержание жизненного цикла программного обеспечения

Платформа обучения информационной безопасности: модель жизненного цикла, процессы разработки и поддержки, порядок приёма и обработки обращений пользователей, персонал, порядок совершенствования программного обеспечения, техническая поддержка, устранение уязвимостей и обновление зависимостей

Наименование программы	**Платформа обучения информационной безопасности**
Сведения о программе опубликованы	**https://xn--80aeb6arfh.xn--p1ai**
Класс программного обеспечения	**12.17 Программное обеспечение для решения отраслевых задач в области образования**
Модель жизненного цикла	**Эволюционная (итеративно-инкрементальная), с практиками непрерывной интеграции и непрерывной поставки**
Дата	**18.09.2026**
Редакция	**1.1**

Назначение документа

Документ описывает совокупность процессов, которыми правообладатель поддерживает жизненный цикл программного обеспечения: от сбора и анализа требований до вывода версий из эксплуатации, включая порядок работы с обращениями пользователей, состав и ответственность персонала, обеспечивающего эти процессы, и порядок устранения уязвимостей. Документ подготовлен в составе технической документации, прилагаемой к заявлению о включении сведений о программном обеспечении в единый реестр российских программ для электронных вычислительных машин и баз данных.

Документ описывает процессы в том виде, в котором они фактически действуют на дату составления документа, а не в виде намерений или планов. Показатели — число проверок, состав конвейера сборки, категории коммитов — сняты с действующей конфигурации репозитория и истории изменений, а не заданы как целевые ориентиры.

Оглавление

1. Общие сведения

1.1. Назначение документа и область действия

Документ описывает процессы, которыми обеспечивается поддержание жизненного цикла программного обеспечения: разработку новой функциональности, сопровождение уже выпущенных версий, устранение неисправностей и уязвимостей, работу с обращениями пользователей и развитие продукта. Документ адресован экспертам, рассматривающим заявление о включении сведений о программном обеспечении в реестр, и предназначен для подтверждения того, что жизненный цикл продукта организован, а не ведётся бессистемно.

Область действия документа — программное обеспечение «Платформа обучения информационной безопасности» целиком: серверная часть, клиентская часть, конвейер сборки и поставки, инфраструктура развёртывания. Документ не описывает процессы иных продуктов правообладателя и не заменяет собой руководство пользователя, руководство администратора или инструкцию по установке, которые входят в комплект технической документации отдельными документами.

1.2. Наименование и класс программного обеспечения

Полное наименование — «Платформа обучения информационной безопасности». Класс — 12.17 Программное обеспечение для решения отраслевых задач в области образования. Программное обеспечение поставляется как веб-приложение для установки в контуре эксплуатирующей организации: в поставку входят исходный код, сборочные файлы, инструкция по развёртыванию и сценарии первичной инициализации.

1.3. Модель жизненного цикла

Жизненный цикл организован по **эволюционной (итеративно-инкрементальной) модели** с практиками непрерывной интеграции и непрерывной поставки. Модель означает, что продукт развивается короткими итерациями — от отдельного исправления до законченной функциональной возможности, — каждая из которых проходит один и тот же путь: формализация требования, разработка, автоматическая проверка, публикация версии. Классическая каскадная модель с раздельными длительными фазами разработки и поддержки к продукту не применяется: поддержка выпущенных версий и разработка новых возможностей идут параллельно и используют один и тот же процесс контроля качества.

Свойства модели, обеспеченные техническими средствами, а не только соглашением между участниками:

1.3. Модель жизненного цикла (продолжение)

Свойство	Чем обеспечено
Штатный путь в эксплуатацию	В контуре правообладателя используются образы, собранные конвейером непрерывной интеграции и поставки. Независимая установка из исходного дистрибутива выполняется по инструкции по установке и проходит собственную приёмочную проверку
Автоматический контроль качества	Изменения проходят статический анализ и автоматические проверки — раздел 2.5, 2.6
Версионирование по содержанию изменений	Номер версии и журнал изменений формируются автоматически по типу внесённых изменений, а не назначаются вручную — раздел 6.2
Разделение сред	Изменения проходят проверку в отдельном окружении до развёртывания в контур эксплуатации — раздел 2.7

2. Процессы разработки

2.1. Сбор и анализ требований

Источником истины по функциональным требованиям служит единый документ технического задания, ведущийся в самом репозитории продукта под контролем версий, а не во внешней переписке. Такое решение принято намеренно: изменение требования и изменение кода, которое это требование реализует, проходят один и тот же процесс проверки и оставляют один и тот же след в истории изменений, что исключает расхождение между тем, что описано, и тем, что реализовано.

Уточнение и развитие требований оформляется отдельными изменениями технического задания, которые проходят тот же порядок рассмотрения, что и изменения кода, — предложение, рассмотрение, включение в основную линию разработки. На дату составления документа в истории репозитория зафиксировано 14 таких изменений технического задания, что подтверждает практическое применение процесса, а не декларируемое намерение.

Требование считается готовым к реализации, когда в техническом задании определены: решаемая задача, затрагиваемые роли пользователей, поведение системы в основном и исключительных сценариях, а также критерий, по которому приёмка отличает реализованное требование от нереализованного. Требования, прямо отнесённые техническим заданием к области, не подлежащей реализации, фиксируются как таковые, чтобы решение не пересматривалось неявно при последующей разработке.

2.2. Проектирование

Архитектурные решения фиксируются отдельным документом и обязательны к соблюдению при реализации. Продукт построен по слоистой архитектуре: тонкий пользовательский интерфейс на компонентном фреймворке, бизнес-логика — в доменных классах-действиях, модели данных — плоские, без бизнес-методов.

Слой	Назначение
Доменный слой (Actions)	Бизнес-логика в чистых классах с одним публичным методом и внедрением зависимостей через конструктор, без знания об HTTP или пользовательском интерфейсе. Разбит по контекстам (обучение, авторизация, интеграции), внутри контекста — по подобластям
Слой представления (компоненты пользовательского интерфейса)	Тонкие компоненты: приём ввода, вызов доменного действия, отображение результата. Бизнес-правила в этом слое не реализуются
Модели данных	Связи, области выборки, приведение типов. Без бизнес-методов

Точки входа HTTP	Только тонкие обработчики для случаев, не покрываемых компонентной моделью: внешние обратные вызовы, отдача файлов, служебные операции
Запросы к данным (Queries)	Сложные выборки вынесены из компонентов и доменных действий в отдельные классы, что предотвращает дублирование запросов и позволяет проверять их независимо

Для повторяющихся технических задач — интеграция с внешней службой, экспорт данных в CSV, загрузка файлов — приняты типовые решения (контракт с подстановкой демонстрационной или боевой реализации, единый реестр экспортов, единый порядок отдачи файлов через проверку прав), которые новая функциональность обязана переиспользовать, а не проектировать заново. Значимые отступления от принятых решений обсуждаются и фиксируются в архитектурном документе до начала реализации, а не постфактум.

2.3. Конструирование (написание кода)

Код пишется на языке PHP версии 8.4 с использованием веб-фреймворка Laravel 13 и компонентного фреймворка пользовательского интерфейса Livewire. Правила размещения кода задаются архитектурным документом (раздел 2.2): бизнес-логика — исключительно в доменных классах-действиях, а не в контроллерах или компонентах интерфейса.

Единообразие оформления кода обеспечивается автоматическим форматированием (набор правил для PHP-кода стандарта Laravel, дополненный правилами упорядочивания импортов по алфавиту, запретом неиспользуемых импортов, единообразным оформлением блоков документирования и завершающих запятых в многострочных конструкциях). Форматирование запускается автоматически перед каждой фиксацией изменений и повторно перед их отправкой в общий репозиторий — раздел 2.5.

Корректность типов и потенциальные ошибки времени выполнения проверяются статическим анализом на максимальном для используемого анализатора восьмом уровне строгости, расширенным правилами для веб-фреймворка. Анализу подвергается весь исходный код приложения, фабрики и заполнители тестовых данных, тесты и служебные скрипты репозитория.

Отдельная автоматическая проверка отклоняет фиксацию изменений, если в отправляемых файлах на языке PHP присутствуют отладочные вызовы (вывод содержимого переменных на экран, остановка выполнения для отладки), — такой код не должен попадать в общую линию разработки даже временно.

2.4. Управление репозиторием и ветвление

Исходный код ведётся в системе контроля версий Git, размещённой на внутренней площадке, управляемой организацией. Модель ветвления — с двумя долгоживущими ветками и короткоживущими ветками разработки:

Ветка	Назначение
master	Стабильная версия, соответствующая тому, что развёрнуто или готово к развёртыванию в контуре эксплуатации. Прямая запись в ветку запрещена организационно — попадание изменений производится только запросом на слияние
develop	Основная ветка разработки, соответствующая предпродакшн-окружению проверки. Ветки разработки ответвляются от неё и в неё же возвращаются
Ветки разработки	Одна ветка на одно логически законченное изменение, названная по типу изменения и краткому описанию задачи (например, ветка исправления или ветка новой возможности). Ответвляются от develop, возвращаются в неё запросом на слияние

Возврат ветки разработки в основную ветку оформляется запросом на слияние по установленному шаблону, в котором автор обязан описать, что изменилось и почему, каким способом изменение проверено (какие автоматические проверки пройдены, что осмотрено вручную), затронуты ли права доступа и персональные данные пользователей платформы, и что сознательно осталось нереализованным. Форма запроса на слияние хранится в самом репозитории и обязательна для каждого изменения.

Сообщения фиксации изменений оформляются по соглашению **Conventional Commits** на русском языке: тип изменения, необязательная область изменения, краткое описание. Тип определяет характер изменения и его влияние на номер версии — раздел 6.2. Соответствие формату проверяется автоматически на этапе фиксации изменения (перехватчик фиксации, установленный в рабочей копии каждого участника): сообщение, не соответствующее формату, не принимается.

Тип	Когда применяется	Влияние на версию
feat	Новая функциональность	минорная
fix	Исправление ошибки	патч
perf	Улучшение производительности	патч
revert	Откат ранее внесённого изменения	патч
refactor	Изменение устройства кода без изменения поведения	без влияния
docs	Документация, включая техническое задание	без влияния
style	Форматирование без изменения логики	без влияния

test	Добавление или правка автоматических проверок	без влияния
chore	Служебные изменения, включая обновление зависимостей	без влияния
ci	Изменения конвейера сборки и поставки	без влияния
build	Изменения сборки, статических ресурсов	без влияния

Область изменения (например, часть продукта, к которой оно относится) — необязательный, но поощряемый элемент сообщения; при существенном или несовместимом изменении сообщение отдельно отмечается как влекущее старшее (мажорное) изменение номера версии.

На дату составления документа история репозитория насчитывает 169 фиксаций изменений; подавляющее большинство из них оформлено по этому соглашению: более 60 — с типом исправления ошибки, более 20 — с типом документации, более 20 — с типом новой возможности, остальные распределены между служебными изменениями, добавлением проверок, настройкой конвейера сборки и улучшением устройства кода. Распределение подтверждает, что соглашение применяется на практике, а не декларируется без использования; автоматическая проверка формата (раздел 2.4) действует с определённого момента разработки и на более ранние фиксации не распространяется.

2.5. Контроль качества кода

Контроль качества выстроен в три рубежа, каждый следующий шире предыдущего: локальная проверка перед фиксацией изменения, локальная проверка перед его отправкой в общий репозиторий, проверка в конвейере сборки при поступлении изменения в общую ветку.

Рубеж	Что проверяется	Когда срабатывает
Перед фиксацией изменения	Форматирование и статический анализ изменённых файлов, отсутствие отладочных вызовов — только по файлам, вносимым конкретной фиксацией	Автоматически на рабочем месте каждого участника разработки при выполнении фиксации
Перед отправкой изменений	Полное форматирование, полный статический анализ, полный набор модульных и функциональных автоматических проверок по всему проекту	Автоматически на рабочем месте при отправке ветки в общий репозиторий
В конвейере сборки	Те же проверки независимо от состояния рабочего места конкретного участника, плюс аудит уязвимостей используемых зависимостей — раздел 8.1	Автоматически при поступлении изменений в ветки develop и master

Третий рубеж существен потому, что первые два выполняются локально и полагаются на то, что инструменты проверки на самом деле установлены и включены на рабочем месте участника разработки. Конвейер сборки проверяет каждое изменение заново, в чистом окружении, независимо от того, что было или не было выполнено локально, и является тем рубежом, миновать который невозможно: слияние изменений в основные ветки не выполняется автоматически, пока связанные проверки не завершились успешно.

Дополнительным рубежом содержательного контроля служит запрос на слияние (раздел 2.4): второй участник разработки рассматривает изменение по существу — соответствие требованию, соблюдение архитектурных правил, соответствие уровням проверки прав доступа, обращение с персональными данными. Форма запроса на слияние прямо требует подтверждения того, что для нового или изменённого экрана, публичного метода компонента или маршрута проверка прав выполнена на всех предусмотренных уровнях, поскольку скрытый в разметке интерфейса элемент управления защитой не является: методы компонентов вызываются с клиента напрямую.

2.6. Тестирование

Автоматические проверки образуют несколько независимых наборов, различающихся по тому, что именно проверяется и в каком окружении выполняется.

Набор	Файлов	Что проверяет	Окружение выполнения
Модульные (Unit)	53	Доменные классы-действия, объекты переноса данных, вспомогательные службы в изоляции	Реальная система управления базами данных — часть проверок сеет роли и права доступа, выполняя настоящие запросы

Функциональные (Feature)	136	Компоненты пользовательского интерфейса, маршруты, авторизация, экспорты, входящие обращения внешних служб — весь стек приложения целиком	Реальная система управления базами данных: часть схемы данных (материализованный путь дерева подразделений) реализована триггерами и специализированными индексами базы данных, которых упрощённая тестовая база не поддерживает, поэтому проверяется поведение, действительно присутствующее в продукте
Архитектурные (Arch)	3	Правила, нарушение которых не проявляется ни в одном пользовательском сценарии и потому не ловится обычными проверками: доступные имена элементов интерфейса, отсутствие бесконтрольной передачи атрибутов компонентов, обязательная проверка прав в административных компонентах	—
Модульные проверки клиентского кода	6	Чистые функции клиентской части, не требующие браузера или сборки	Штатный исполнитель проверок среды выполнения JavaScript
Сценарии в браузере (дымовые проверки)	17	Сквозные пользовательские сценарии на работающем приложении: прохождение обучения, администрирование, разграничение доступа, отображение экранов, отсутствие визуальных дефектов	Управляемый браузер против развёрнутого окружения

Модульный, функциональный и архитектурный наборы вместе включают 192 файла проверок и не менее 2072 отдельных проверочных случаев; часть проверок дополнительно прогоняется на нескольких наборах входных данных, что увеличивает фактическое число выполняемых проверок сверх этой величины. Целевая практика проекта — модульная проверка на каждый доменный класс-действие и функциональная проверка на каждый сценарий, предусмотренный техническим заданием.

Автоматический прогон встроен в оба локальных рубежа контроля (раздел 2.5: полный прогон модульных и функциональных проверок перед отправкой изменений, с ограничением числа параллельных процессов, чтобы прогон не исчерпывал память рабочего места) и в конвейер сборки, где модульные и функциональные проверки выполняются отдельными шагами против настоящей системы управления базами данных, поднимаемой для этой цели заново при каждом прогоне. Сценарии в браузере выполняются конвейером автоматически после каждого развёртывания в предпродакшн-окружение и дополнительно — по запросу участника разработки на локальном окружении.

2.7. Сборка и выпуск версий

В штатном контуре правообладателя используются образы, собранные и проверенные конвейером непрерывной интеграции и поставки. Независимая установка для целей экспертизы либо в контуре заказчика выполняется из исходного дистрибутива по инструкции по установке, после чего отдельно проверяется работоспособность полученного экземпляра.

Конвейер организован последовательными этапами; переход к следующему этапу происходит только при успешном завершении предыдущего:

Этап	Содержание
Проверка стиля и статический анализ	Форматирование кода, статический анализ восьмого уровня строгости, аудит уязвимостей зависимостей — раздел 8.1
Автоматические проверки	Модульный и функциональный наборы против настоящей системы управления базами данных, поднимаемой конвейером на время прогона — раздел 2.6
Сборка образов	Сборка образа приложения и образа веб-сервера из состояния ветки, публикация в собственный реестр образов контейнеров организации с меткой по идентификатору изменения
Выпуск версии	Только для ветки master: автоматический расчёт номера версии, формирование журнала изменений, простановка метки версии — раздел 6.2
Развёртывание	Обновление предпродакшн-окружения — автоматически при поступлении изменений в develop; обновление контура эксплуатации — только для ветки master, с обязательным ручным подтверждением

Дымовые проверки	Прогон сценариев в браузере против обновлённого предпродакшн-окружения — раздел 2.6

Ручное подтверждение перед обновлением контура эксплуатации — намеренный рубеж: развёртывание в предпродакшн-окружение происходит без вмешательства человека при каждом изменении основной ветки разработки, тогда как обновление контура, которым пользуются работники организации, требует осознанного решения человека, ответственного за выпуск. Применение миграций схемы базы данных выполняется автоматически точкой входа контейнера приложения при его запуске, а не отдельной ручной операцией, что исключает рассинхронизацию между кодом и структурой данных.

3. Процессы поддержки

3.1. Управление конфигурацией и версиями

Версия исходного кода однозначно определяется системой контроля версий: каждое изменение, дошедшее до основных веток, имеет неизменяемый идентификатор, а образы контейнеров помечаются этим идентификатором и, для веток develop и master, дополнительно именем ветки. Файл журнала изменений и файл текущего номера версии обновляются автоматически при выпуске версии (раздел 6.2) и хранятся в самом репозитории, поэтому соответствие между кодом и номером версии не может разойтись по забывчивости.

Конфигурация приложения разделена на два уровня по характеру изменения. Параметры окружения (адреса служб, признаки включения необязательных возможностей) задаются на уровне развёртывания и меняются вместе с ним. Учётные данные внешних интеграций — адреса и ключи почтового реле, службы сбора отчётов об ошибках и других подключаемых служб — хранятся не в конфигурации развёртывания, а в зашифрованном виде в базе данных самого приложения и изменяются через административный интерфейс без пересборки и без остановки приложения; каждое такое изменение фиксируется в журнале изменений интеграций с указанием, что и когда изменено.

3.1.1. Размещение технической инфраструктуры разработки

Исходный текст программы, система сборки и реестр контейнерных образов размещены на сервере gitlab.glav.pro (адрес 72.56.249.179), предоставленном поставщиком услуг ООО «ТАЙМВЭБ.КЛАУД». Страна размещения — Российская Федерация. Центр обработки данных: 117545, г. Москва, улица Подольских Курсантов, д. 15Б. Работающий экземпляр программы размещён на другой площадке — сведения о ней приведены в документе о технических средствах хранения исходного текста и объектного кода.

3.2. Порядок выпуска обновлений и исправлений

Исправление ошибки проходит тот же путь, что и любое иное изменение (раздел 2.4, 2.7): ветвь исправления от develop, автоматические и содержательные проверки, слияние, автоматическое развёртывание в предпродакшн-окружение. От обычной функциональной доработки исправление отличается только приоритетом рассмотрения и типом фиксации изменения (fix), который однозначно определяет его как патч-версию при выпуске (раздел 6.2).

Подтверждённое на предпродакшн-окружении исправление переносится в ветку master запросом на слияние; попадание в master немедленно и автоматически формирует новую версию продукта и запускает развёртывание в контур эксплуатации после ручного

подтверждения (раздел 2.7). Отдельного, отличного от общего, процесса выпуска для исправлений не предусмотрено: выделенный процесс усложнял бы порядок работы без выигрыша, поскольку конвейер и так проверяет и разворачивает изменения быстрее, чем занял бы отдельный внеочередной порядок.

3.3. Устранение неисправностей

Обращение о неисправности регистрируется в системе отслеживания задач репозитория по установленному шаблону, требующему: наблюдаемое поведение без предположений о причине; ожидаемое поведение; шаги воспроизведения; обстановка (окружение, роль учётной записи, браузер и устройство, курс и попытка тестирования, если неисправность относится к прохождению обучения); что уже проверено, чтобы не повторять уже отпавшие версии причины; и влияние — по фиксированной шкале, приведённой в разделе 4.2. Персональные данные обучаемых в описание неисправности не включаются: для разбора достаточно роли и признака, по которому запись находится.

Разбор неисправности начинается с воспроизведения, а не с предположений о причине: формулировка, с которой поступает обращение, описывает симптом, а фактических причин у одного и того же симптома обычно несколько. После установления причины исправление проходит обычный порядок разработки (разделы 2.4–2.7). Приоритет рассмотрения определяется влиянием неисправности по шкале раздела 4.2: неисправности, делающие обучение или тестирование невозможным либо искажающие результаты, либо раскрывающие сведения не тому, кому они предназначены, рассматриваются в первую очередь.

3.4. Резервное копирование и восстановление

Резервное копирование базы данных выполняется планово, перед обновлением контура эксплуатации и по требованию. Способы дополняют друг друга; независимость копий от отказа рабочего сервера обеспечивается их хранением вне этого сервера:

Уровень	Периодичность	Устройство
Плановое копирование	Ежесуточно, по расписанию планировщика	Выгрузка средствами системы управления базами данных в собственном формате с автоматическим удалением копий старше установленного срока хранения

Копирование перед развёртыванием в контур эксплуатации	Перед каждым обновлением контура эксплуатации	Выгрузка с сохранением заданного числа последних копий — рубеж, дающий точку возврата непосредственно перед изменением, а не только на начало суток
Разовое копирование по требованию	По решению ответственного за эксплуатацию	Ручной запуск того же механизма, что и плановое копирование, перед действиями повышенного риска (например, регламентными операциями с шифрованием)

Восстановление выполняется из плановой копии либо из копии, сделанной перед развёртыванием, штатными средствами системы управления базами данных, с проверкой состояния миграций после восстановления. Помимо базы данных, отдельного резервирования требуют файлы объектного хранилища, сам репозиторий, конфигурация окружения и ключи доступа: система контроля версий хранит историю изменений кода, но не заменяет собой резервную копию, а секреты хранятся отдельно от репозитория, с ограниченным кругом лиц, имеющих к ним доступ.

База данных копируется ежедневно в 02:30 по времени сервера; по умолчанию хранятся суточные копии за 30 суток и по одной месячной копии за каждый из 12 месяцев, а срок хранения настраивается параметрами окружения без изменения кода. Каталог хранения копий вынесен за пределы контейнера приложения, чтобы пересоздание контейнера не уничтожало накопленные копии, а дополнительная копия хранится вне рабочего сервера; размещение этого хранилища на территории Российской Федерации подтверждается документами поставщика услуг. Наличие годной копии проверяется пробным восстановлением в отдельной среде, а не только по размеру файла; организацию и контроль резервного копирования выполняет администратор системы (эксплуатация) — раздел 5.1.

Неизменяемый журнал юридически значимых событий платформы (подтверждения ознакомления, результаты проверки знаний, выдача и отзыв сертификатов) защищён дополнительно, независимо от резервного копирования: изменение и удаление его записей запрещено триггером базы данных, действующим независимо от того, под какой учётной записью выполняется обращение, и сохраняющимся при восстановлении схемы из резервной копии.

3.5. Мониторинг и реагирование на инциденты

Работоспособность контура эксплуатации проверяется на нескольких независимых уровнях:

Средство	Что показывает

Служебная конечная точка состояния	Доступность базы данных, кэша, обработчика фоновых задач и объектного хранилища в машиночитаемом виде, защищена отдельным токеном доступа для внешних систем контроля
Экран состояния системы в административном разделе	Те же сведения в человекочитаемом виде для ответственного персонала, включая диагностику причины при недоступности объектного хранилища
Панель управления фоновыми задачами	Состояние очередей и обработчиков фоновых задач, список задач, завершившихся с ошибкой, с возможностью повторного запуска, доступна только персоналу с правом администрирования
Система сбора отчётов об ошибках	Сведения о сбоях выполнения кода на сервере поступают в приёмник, размещённый в контуре организации; адрес приёмника хранится в зашифрованном виде и настраивается без пересборки и без остановки приложения
Ежесуточная проверка целостности неизменяемого журнала	Пересчёт хеш-цепочки записей юридически значимых событий; расхождение, означающее вмешательство в обход приложения, немедленно уведомляет ответственный персонал

Реагирование на инцидент начинается с диагностики по перечисленным средствам, продолжается операционными сценариями, описывающими типовые неисправности и порядок их устранения (недоступность объектного хранилища, задержка исходящей почты, переполнение очереди фоновых задач, расхождение отпечатка клиентских ресурсов после развёртывания), и завершается, если причина требует изменения кода, обычным порядком разработки (разделы 2.4–2.7) с приоритетом, определяемым влиянием на работу пользователей.

4. Приём и обработка обращений пользователей

4.1. Каналы обращения

Обращение адресуется в зависимости от его характера, чтобы попасть сразу к тому, кто способен его разрешить, минуя лишние промежуточные передачи:

Характер обращения	Кому адресуется
Не получается войти, забыт пароль, заблокирован вход, вопрос по учётной записи	Администратору платформы в эксплуатирующей организации
Не назначен нужный курс, неверный срок обучения, вопрос по содержанию учебного материала	Администратору обучения (методисту) в эксплуатирующей организации
Платформа выдаёт ошибку, экран отображается неверно, действие не выполняется	В службу технической поддержки правообладателя
Подозрение на доступ посторонних к учётной записи, подозрительное письмо, иное событие, затрагивающее безопасность	В службу информационной безопасности незамедлительно, независимо от того, кем и когда подозрение обнаружено

Первые два канала разрешают обращение силами эксплуатирующей организации без обращения к правообладателю: администратор платформы и администратор обучения выполняют операции над учётными записями и назначениями штатными средствами административного раздела. Обращение доходит до правообладателя, когда причина не в данных конкретной организации, а в поведении самого программного обеспечения, — тогда оно фиксируется как неисправность (раздел 3.3) и обрабатывается порядком разработки.

4.2. Классификация обращений

Обращения о неисправностях классифицируются по влиянию на работу — эта шкала одновременно служит полем формы обращения (раздел 3.3) и основанием для определения приоритета рассмотрения:

Влияние	Пример
Обучение или тестирование невозможно	Работник не может открыть назначенный материал или начать проверку знаний; администратор не может опубликовать версию курса
Результаты обучения искажаются или теряются	Неверно посчитан результат проверки знаний; не выдан либо неверно оформлен сертификат

Сведения видны тому, кому не должны быть видны	Раскрытие результатов или персональных данных за пределами прав доступа обратившегося
Неудобство, работа не блокируется	Некорректное отображение, не мешающее выполнить действие; неточная формулировка на экране

Обращения по вопросам, не являющимся неисправностью (методические вопросы, вопросы по учётной записи), в эту шкалу не попадают и обрабатываются каналами раздела 4.1 без формальной классификации по влиянию.

4.3. Сроки реакции

Обращения, отнесённые к первым двум ступеням шкалы влияния (обучение или тестирование невозможно; результаты обучения искажаются или теряются), а также любые обращения, связанные с безопасностью, рассматриваются в приоритетном порядке, вне общей очереди. Обращения об удобстве интерфейса и неточностях, не блокирующих работу, рассматриваются в общем порядке разработки, наравне с иными изменениями технического задания.

> **Числовые нормативы сроков реакции не зафиксированы в документации проекта**
>
> В отличие от приоритетности, конкретные предельные значения времени реакции и времени разрешения по каждой ступени шкалы влияния (часы или рабочие дни) на дату составления документа не установлены внутренним регламентом и подлежат определению эксплуатирующей организацией и правообладателем отдельно.

4.4. Порядок эскалации

Обращение, которое не удалось разрешить силами администратора платформы или администратора обучения в эксплуатирующей организации, передаётся в службу технической поддержки правообладателя (раздел 4.1). Если служба технической поддержки устанавливает, что причина — в поведении программного обеспечения, а не в данных или настройках конкретной организации, обращение регистрируется как неисправность в системе отслеживания задач репозитория (раздел 3.3) и передаётся в разработку.

Обращение, связанное с безопасностью (раздел 4.1), служба технической поддержки передаёт ответственному специалисту вне общей очереди в режиме работы, приведённом в разделе 7.1: пользователь обязан сообщить о подозрении незамедлительно, а экстренное реагирование внутри эксплуатирующей организации — её собственная задача. Обращение, не разрешённое службой технической поддержки или требующее решения, выходящего за пределы её полномочий, передаётся ответственному за поддержку жизненного цикла

продукта (раздел 5.1).

5. Персонал, обеспечивающий жизненный цикл

5.1. Роли и зоны ответственности

Ниже приведены роли, обеспечивающие процессы разделов 2–4, 6–8. Роль — это область ответственности, а не штатная единица: на практике один участник команды может исполнять несколько ролей, а объёмная роль — распределяться между несколькими участниками. Персональный состав исполнителей ролей не является предметом настоящего документа и может меняться без изменения самих процессов.

Роль	Зона ответственности
Ответственный за требования	Ведёт техническое задание как источник истины по функциональности (раздел 2.1), формулирует и приоритизирует новые требования совместно с эксплуатирующей организацией, принимает решение о готовности требования к реализации и о полноте его выполнения
Архитектор	Определяет и поддерживает актуальность архитектурных решений (раздел 2.2), рассматривает значимые отступления от принятых решений, следит за отсутствием повторного изобретения уже принятых типовых решений
Разработчик	Реализует требования кодом в соответствии с архитектурными решениями и правилами конструирования (раздел 2.3), обеспечивает прохождение локальных рубежей контроля качества перед отправкой изменения (раздел 2.5), участвует в содержательном рассмотрении чужих изменений
Специалист по тестированию	Формирует и поддерживает автоматические проверки (раздел 2.6), выполняет ручную проверку в браузере для изменений пользовательского интерфейса, участвует в приёмке версии на предпродакшн-окружении перед переносом в контур эксплуатации
Инженер сопровождения конвейера и развёртывания	Поддерживает конвейер непрерывной интеграции и поставки, образы контейнеров, конфигурацию окружений развёртывания (раздел 2.7), заведует процедурой выпуска версий
Администратор системы (эксплуатация)	Эксплуатирует контур предпродакшн-контроля и контур продуктивной работы: резервное копирование и восстановление (раздел 3.4), мониторинг и реагирование на инциденты инфраструктуры (раздел 3.5), применение обновлений в контуре эксплуатации
Специалист по информационной безопасности	Определяет политику безопасности разработки и эксплуатации, участвует в контроле уязвимостей зависимостей (раздел 8), реагирует на обращения по подозрению на компрометацию (раздел 4.1)

Служба технической поддержки	Принимает и обрабатывает обращения пользователей (раздел 4), выполняет типовые операции в пределах своих полномочий, классифицирует и эскалирует обращения, не решаемые силами эксплуатирующей организации
Ответственный за поддержку жизненного цикла продукта	Принимает решения о развитии продукта (раздел 6.1), рассматривает эскалированные обращения (раздел 4.4), отвечает за целостность процессов, описанных настоящим документом, в целом

Системное администрирование выполняет сотрудник ООО «ИНСТИТУТ ПОВЫШЕНИЯ КВАЛИФИКАЦИИ ГЛАВПРО». Тем самым роль администратора системы (эксплуатация) на дату составления документа замещена конкретным персоналом — в отличие от прочих ролей настоящего раздела, персональный состав исполнителей которых документом не фиксируется.

5.2. Квалификационные требования

Общее требование ко всем ролям — высшее или среднее профессиональное образование по профилю, соответствующему исполняемой роли (программная инженерия, информационные системы и технологии, информационная безопасность, прикладная информатика), либо документально подтверждённый практический опыт в соответствующей области, признаваемый эквивалентным. Дополнительные требования по ролям:

Роль	Дополнительное требование
Разработчик	Практическое владение языком PHP и используемым стеком веб-фреймворка; понимание принципов безопасной разработки — обработки пользовательского ввода, разграничения прав доступа, работы с персональными данными
Архитектор	Опыт проектирования многослойных веб-приложений; владение принципами, применёнными в продукте (разделение бизнес-логики и представления, работа с внешними интеграциями через контракт)
Специалист по тестированию	Знание методик тестирования программного обеспечения и практическое владение инструментами автоматизированного тестирования, применяемыми в проекте
Инженер сопровождения конвейера и развёртывания, администратор системы	Знание операционных систем семейства Linux, средств контейнеризации, администрирования реляционной системы управления базами данных

Специалист по информационной безопасности	Знание законодательства Российской Федерации о персональных данных и требований к обработке защищаемой информации, практика реагирования на инциденты информационной безопасности
Служба технической поддержки	Знание функциональных возможностей продукта в объёме руководства пользователя и руководства администратора, навыки работы с обращениями пользователей

6. Совершенствование программного обеспечения

6.1. Порядок принятия решений о развитии

Решение о развитии продукта начинается не с кода, а с изменения технического задания (раздел 2.1): новая функциональная возможность формулируется, обсуждается и включается в документ требований прежде, чем начинается её реализация. Такой порядок исключает ситуацию, при которой функциональность появляется в продукте, не будучи нигде документально закреплённой, и наоборот — документ описывает то, чего в продукте на самом деле нет.

Решение о приоритете развития принимает ответственный за поддержку жизненного цикла продукта совместно с ответственным за требования, с учётом обратной связи, поступающей от эксплуатирующей организации через каналы раздела 4.1 и от службы технической поддержки. Технические предложения по улучшению устройства кода без изменения видимого поведения (сокращение технического долга) оформляются тем же порядком изменения технического задания или отдельным изменением с типом фиксации, отражающим переустройство кода без изменения поведения, и включаются в общую очередь разработки наравне с функциональными доработками.

6.2. Планирование версий

Номер версии продукта формируется автоматически из содержания фиксаций изменений, попавших в ветку master, по правилам семантического версионирования: наличие изменений с типом новой функциональности повышает минорную часть номера, наличие исправлений, улучшений производительности или откатов — патч-часть, а несовместимое изменение — старшую (мажорную) часть. Изменения типов, не влияющих на поведение (документация, форматирование, служебные правки, изменения конвейера сборки, тестов), номер версии не меняют.

Автоматизация означает, что решение о номере следующей версии не принимается человеком в момент выпуска — оно является прямым следствием того, какие типы изменений накопились с предыдущего выпуска, и потому воспроизводимо и не зависит от того, кто именно формирует выпуск. Одновременно с расчётом номера версии автоматически формируется журнал изменений на русском языке, сгруппированный по типу (новые возможности, исправления, производительность и так далее), и публикуется в системе контроля версий вместе с меткой версии.

Планирование версий тем самым сведено к планированию содержания: выпуск версии происходит по факту накопления готовых, проверенных изменений в ветке master, а не по календарному графику с фиксированной датой. Продукт развивается непрерывно короткими

шагами (раздел 1.3), поэтому крупных, редких «мажорных релизов» с заранее объявленной датой модель не предполагает; версия с повышением старшей части номера появляется тогда, когда накопленное изменение действительно несовместимо с предыдущим поведением.

6.3. Дорожная карта развития

Ближайшие двенадцать месяцев развития продукта ориентированы на укрепление эксплуатационной устойчивости платформы, расширение отчётности для эксплуатирующих организаций и подготовку инфраструктуры к росту нагрузки. План носит рамочный характер и уточняется по мере поступления обратной связи от эксплуатирующих организаций — раздел 6.1.

Срок	Направление развития
10.2026 —	Перенос резервных копий на отдельное хранилище
11.2026 —	Публикация документации из объектного хранилища
12.2026 —	Плановое обновление компонентов платформы
01.2027 —	Ввод стенда предварительной проверки обновлений
02.2027 —	Подключение сервиса наблюдения при аттестации
03.2027 —	Расширение отчётности по результатам обучения
05.2027 —	Нагрузочные испытания и уточнение требований к оборудованию
07.2027 —	Развитие имитационных рассылок
09.2027 —	Мониторинг доступности с оповещением дежурного

7. Техническая поддержка

7.1. Режим работы

Обработка обращений и приём телефонных звонков службой технической поддержки правообладателя ведутся в следующем режиме: понедельник — пятница, с 09:00 до 18:00 по московскому времени (MSK, UTC+3). Обращения по электронной почте принимаются круглосуточно, а их обработка начинается в ближайшее рабочее время службы поддержки. Обращения, отнесённые к приоритетным по шкале влияния раздела 4.2 (обучение или тестирование невозможно; результаты обучения искажаются или теряются), а также обращения, связанные с безопасностью, рассматриваются вне очереди в пределах указанного режима работы — раздел 4.3, 4.4. Круглосуточное дежурство настоящим документом не устанавливается; иные условия и сроки реакции могут определяться отдельным договором либо соглашением об уровне обслуживания между правообладателем и эксплуатирующей организацией. Числовые предельные сроки реакции и разрешения обращения, не согласованные сторонами отдельно, по каждому уровню приоритета внутренним регламентом на дату составления документа не установлены — раздел 4.3.

7.2. Способы обращения

Обращения в службу технической поддержки правообладателя направляются по адресу электронной почты info@glav.pro либо по телефону +7 909 539-22-49 в режиме работы, приведённом в разделе 7.1. Актуальные сведения о способах обращения дополнительно размещаются на официальном сайте правообладателя — https://xn--80aeb6arfh.xn--p1ai. Адрес местонахождения службы технической поддержки правообладателя — 656031, Алтайский край, г. Барнаул, пр-кт Строителей, зд. 45, кабинет 251.

В обращении указываются контакт для обратной связи, адрес используемого экземпляра платформы, роль обратившегося, описание проблемы, время возникновения, шаги воспроизведения, ожидаемый результат и влияние на работу — по классификации раздела 4.2. Допускается приложить снимок экрана, не содержащий паролей, ключей доступа и персональных данных сверх необходимого для разбора обращения.

Способы обращения соответствуют каналам раздела 4.1: обращения по учётным записям и назначению обучения сначала решаются администратором платформы либо администратором обучения в самой эксплуатирующей организации без выхода за её пределы; если причина связана с поведением самого программного обеспечения, обращение передаётся в службу технической поддержки правообладателя по указанным контактам. При подозрении на инцидент безопасности пользователь незамедлительно уведомляет ответственного за информационную безопасность своей организации и, если инцидент затрагивает платформу, — также службу технической поддержки правообладателя; порядок

внутреннего экстренного реагирования определяет эксплуатирующая организация — раздел 4.4.

8. Устранение уязвимостей и обновление зависимостей

8.1. Контроль состава и уязвимостей зависимостей

Состав используемых сторонних компонентов зафиксирован файлами точной фиксации версий для серверной и клиентской части, обеспечивающими воспроизводимую установку одних и тех же версий на любой машине и в конвейере сборки: случайное расхождение версий между рабочим местом участника разработки, конвейером сборки и контуром эксплуатации исключено.

На этапе проверки качества конвейер сборки автоматически выполняет аудит серверных зависимостей на известные уязвимости, сверяя точно зафиксированные версии пакетов производственного окружения с базой данных известных уязвимостей. Отдельно репозиторий подключает штатное средство поиска секретов (учётных данных, ключей доступа), случайно попавших в фиксируемые изменения, — проверка выполняется автоматически для каждого изменения независимо от ветки.

8.2. Порядок обновления зависимостей

Обновление зависимостей выполняется осознанно и раздельно от прочей разработки: изменение версии зависимости фиксируется отдельным изменением с соответствующим типом (служебное изменение или исправление, если обновление устраняет уязвимость) и проходит тот же порядок проверки, что и любое иное изменение кода (разделы 2.5–2.7), — обновлённая зависимость подтверждается полным набором автоматических проверок прежде, чем попадает в основную ветку.

Установка и обновление зависимостей выполняются только внутри контролируемого окружения сборки, а не на произвольном рабочем месте, что исключает попадание в проект версии, отличной от той, что зафиксирована файлом точной фиксации версий и впоследствии соберётся конвейером.

8.3. Реагирование на выявленные уязвимости

Уязвимость, выявленная автоматическим аудитом (раздел 8.1), внешним обращением или иным путём, обрабатывается как неисправность повышенного приоритета (раздел 3.3): обновление или замена уязвимого компонента оформляется изменением с типом исправления, отдельная область изменения для вопросов безопасности предусмотрена соглашением об оформлении изменений (раздел 2.4). Изменение проходит обычный порядок проверки и выпуска (разделы 2.5–2.7); тип исправления обеспечивает автоматическое повышение патч-версии продукта при выпуске, поэтому факт устранения уязвимости

отражается в номере версии и в журнале изменений без дополнительных ручных действий.

Для уязвимости, признанной активно эксплуатируемой либо создающей непосредственный риск для персональных данных или результатов обучения, применяется тот же приоритетный порядок рассмотрения, что и для неисправностей первых двух ступеней шкалы влияния (раздел 4.3): исправление готовится вне общей очереди, а перенос в контур эксплуатации не откладывается до плановой раскатки очередной версии.

9. Актуализация документа

Документ описывает процессы в том виде, в котором они действуют на дату составления. Числовые показатели (состав автоматических проверок, распределение фиксаций изменений по типам, состав этапов конвейера сборки) сняты с фактического состояния репозитория и подлежат пересчёту при очередном пересмотре документа, а не переносятся из предыдущей редакции без проверки.

Документ пересматривается при существенном изменении описанных процессов: смене модели ветвления, изменении состава этапов конвейера сборки и поставки, изменении порядка выпуска версий, изменении состава ролей персонала, обеспечивающего жизненный цикл. Внесение изменений в документ оформляется как изменение технического задания (раздел 2.1) и проходит тот же порядок рассмотрения, что и любое иное изменение технической документации проекта.