**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»**
**(РУТ (МИИТ)**

Рабочая программа дисциплины (модуля),
как компонент образовательной программы
базового высшего образования
по направлению подготовки
09.03.01 Информатика и вычислительная техника,
утвержденной директором ФГАОУ ВО РУТ
(МИИТ) Париновым Д.В.

# РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ (МОДУЛЯ)

## Проектирование API и интеграция приложений

Направление подготовки: 09.03.01 Информатика и вычислительная техника

Направленность (профиль): Технологии разработки программного обеспечения

Форма обучения: Очная

Рабочая программа дисциплины (модуля) в виде электронного документа выгружена из единой корпоративной информационной системы управления университетом и соответствует оригиналу

Простая электронная подпись, выданная ФГАОУ ВО РУТ (МИИТ)
ID подписи: 937226
Подписал: директор центра Проневич Ольга Борисовна
Дата: 12.11.2026

1. Общие сведения о дисциплине (модуле).

Дисциплина формирует способность проектировать, реализовывать и проверять взаимодействие независимо развивающихся компонентов бизнес-приложений. Она отвечает на потребность транспортной отрасли в инженерах интеграции, обеспечивающих обмен данными между учётными, расчётными и согласовательными системами на импортонезависимом технологическом стеке. Студент осваивает проектирование синхронных и событийных контрактов, документирование интерфейсов, выбор способов интеграции и управление их совместимостью. На практике последовательно создаётся интеграционный контур расчётного сервиса на Java с использованием PostgreSQL и Apache Kafka. Особое внимание уделяется идемпотентности, обработке повторов, частичным отказам и безопасной эволюции API. Итогом становится работающий выпуск с автоматизированными проверками, архитектурным решением и комплектом технической документации.

Целью освоения дисциплины является формирование способности проектировать, документировать, реализовывать и проверять программные интерфейсы и интеграционные процедуры, обеспечивающие однозначное, совместимое и устойчивое взаимодействие независимо развивающихся компонентов бизнес-приложения при повторах, задержках и частичных отказах.

Для достижения поставленной цели в рамках дисциплины решается комплекс задач, направленных на формирование у обучающихся способности: Декомпозировать интеграционные сценарии на участников, операции, данные, состояния и внешние эффекты. Обосновывать выбор синхронного, отложенного или событийного способа взаимодействия по заданным архитектурным требованиям. Проектировать машиночитаемые контракты запросов, ответов, событий и ошибок. Реализовывать интеграционные процедуры и адаптеры на Java с соблюдением опубликованных спецификаций. Обеспечивать идемпотентность и согласованность данных при повторах и частичных отказах. Выполнять контрактные, интеграционные и отказные проверки. Собирать интеграционные выпуски и сопровождать их документацией и планом безопасной эволюции.

2. Планируемые результаты обучения по дисциплине (модулю).

Перечень формируемых результатов освоения образовательной программы (компетенций) в результате обучения по дисциплине (модулю):

**ДПК-БП-4** - Способен проектировать и реализовывать архитектуру данных нагруженного бизнес-приложения;

**ОПК-3** - Способен использовать современные информационные технологии и программно-аппаратные средства, в том числе отечественного производства, при решении задач профессиональной деятельности;

**ОПК-5** - Способен участвовать в разработке технической документации, связанной с профессиональной деятельностью, с использованием стандартов, норм и правил;

**ППК-Р3** - Способен выполнять интеграцию программных модулей и компонентов и проверять работоспособность выпусков программного продукта;

**ППК-Р5** - Способен разрабатывать требования и проектировать программное обеспечение.

Обучение по дисциплине (модулю) предполагает, что по его результатам обучающийся будет:

**Знать:**

- виды и назначение технической документации в сфере разработки программного обеспечения.
- требования стандартов к разработке и оформлению технической документации (серии ГОСТ 19, ГОСТ 34, ISO/IEC/IEEE).
- правила версионирования и согласования документации.
- классы современных информационных технологий и программно-аппаратных средств.
- отечественные операционные системы, СУБД и облачные платформы (Единый реестр российского ПО).
- критерии выбора программно-аппаратных средств под профессиональную задачу.
- методы и средства сборки модулей и компонентов компьютерного программного обеспечения.
- интерфейсы взаимодействия с внешней средой.
- интерфейсы взаимодействия внутренних модулей системы.
- методы и средства сборки и интеграции программных модулей и компонентов.
- методы и средства проверки работоспособности выпусков программных продуктов.
- методы и средства проектирования компьютерного программного обеспечения.
- методы и средства проектирования программных интерфейсов.

- устройство и функционирование ИТ-систем/продуктов

- методы моделирования и описания устройства и функционирования ИТ-систем/продуктов, их частей, обеспечения и окружения

- базовые технологии взаимодействия и интеграции систем и компонентов

- методы функциональной декомпозиции ИТ-систем

**Уметь:**

- разрабатывать разделы технической документации (технические задания, спецификации, инструкции, руководства).

- применять стандарты, нормы и правила при разработке документации.

- проверять документацию на полноту, непротиворечивость и соответствие стандартам.

- обоснованно выбирать информационные технологии и программно-аппаратные средства.

- применять отечественные программно-аппаратные средства при решении задач.

- оценивать ограничения, совместимость и стоимость владения выбранных средств.

- писать программный код процедур интеграции программных модулей.

- использовать выбранную среду программирования для разработки процедур интеграции программных модулей.

- применять методы и средства сборки модулей и компонентов компьютерного программного обеспечения, разработки процедур для развертывания компьютерного программного обеспечения, миграции и преобразования данных, создания программных интерфейсов.

- выполнять процедуры сборки программных модулей и компонентов в программный продукт.

- производить настройки параметров программного продукта и осуществлять запуск процедур сборки.

- проводить проверку работоспособности программного продукта.

- выбирать средства реализации требований к компьютерному программному обеспечению.

- вырабатывать варианты реализации компьютерного программного обеспечения.

- проводить оценку и обоснование рекомендуемых решений.

- декомпозировать ИТ-системы и ИТ-продукты на подсистемы и элементы поставки

- описывать интерфейсы пользователя на логическом уровне

- описывать интеграции со смежными системами на логическом уровне

**Владеть:**

- навыками ведения проектной документации в инструментах совместной работы.
- навыками работы в отечественной ИТ-инфраструктуре (включая разработку под ОС «Аврора»).

3. Объем дисциплины (модуля).

3.1. Общая трудоемкость дисциплины (модуля).

Общая трудоемкость дисциплины (модуля) составляет 3 з.е. (108 академических часа(ов).

3.2. Объем дисциплины (модуля) в форме контактной работы обучающихся с педагогическими работниками и (или) лицами, привлекаемыми к реализации образовательной программы на иных условиях, при проведении учебных занятий:

| Тип учебных занятий | Количество часов | |
|---|---|---|
| | Всего | Семестр №6 |
| Контактная работа при проведении учебных занятий (всего): | 48 | 48 |
| В том числе: | | |
| Занятия лекционного типа | 16 | 16 |
| Занятия семинарского типа | 32 | 32 |

3.3. Объем дисциплины (модуля) в форме самостоятельной работы обучающихся, а также в форме контактной работы обучающихся с педагогическими работниками и (или) лицами, привлекаемыми к реализации образовательной программы на иных условиях, при проведении промежуточной аттестации составляет 60 академических часа (ов).

3.4. При обучении по индивидуальному учебному плану, в том числе при ускоренном обучении, объем дисциплины (модуля) может быть реализован полностью в форме самостоятельной работы обучающихся, а также в форме контактной работы обучающихся с педагогическими работниками и (или) лицами, привлекаемыми к реализации образовательной программы на иных условиях, при проведении промежуточной аттестации.

4. Содержание дисциплины (модуля).

4.1. Занятия лекционного типа.

| № п/п | Тематика лекционных занятий / краткое содержание |
|---|---|
| 1 | Интеграционный сценарий и программный интерфейс как контракт<br>Рассматриваемые вопросы:<br>- роль интеграции в архитектуре бизнес-приложения;<br>- участники, операции, данные, состояния и внешние эффекты интеграционного сценария;<br>- границы ответственности производителя и потребителя;<br>- контракт как единый источник соглашений между независимыми компонентами;<br>- декомпозиция сценария и диаграммы последовательности взаимодействия;<br>- типовые противоречия требований и способы их проектного разрешения. |
| 2 | Выбор способа интеграции и проектирование моделей данных<br>Рассматриваемые вопросы:<br>- синхронное взаимодействие по модели запрос-ответ;<br>- отложенные операции и получение результата по идентификатору;<br>- событийное взаимодействие и слабая связанность компонентов;<br>- критерии выбора по задержке, надёжности, связанности и стоимости сопровождения;<br>- матрица сравнения архитектурных альтернатив;<br>- объекты передачи данных и отделение контрактных моделей от внутренних моделей;<br>- JSON Schema, сериализация и валидация интеграционных сообщений. |
| 3 | Проектирование синхронного API по спецификации OpenAPI<br>Рассматриваемые вопросы:<br>- назначение и структура спецификации OpenAPI 3.x;<br>- описание ресурсов, операций, параметров и тел запросов и ответов;<br>- обязательность полей и ограничения схем данных;<br>- контракт ошибок, HTTP-статусы и машиночитаемые сведения об ошибке;<br>- идентификаторы запроса и операции;<br>- жизненный цикл отложенной операции и получение её результата;<br>- правила идемпотентности создающих и изменяющих операций. |
| 4 | Реализация синхронного взаимодействия на Java<br>Рассматриваемые вопросы:<br>- подход specification-first и контроль соответствия кода спецификации;<br>- генерация моделей и клиентских интерфейсов из OpenAPI;<br>- реализация HTTP-интерфейса при помощи Spring Boot;<br>- клиентский адаптер и изоляция внешнего контракта от внутренней логики;<br>- обработка ошибок, тайм-аутов и повторных вызовов;<br>- хранение ключей идемпотентности и результатов обработки;<br>- ограничения PostgreSQL как механизм защиты прикладных инвариантов. |
| 5 | Согласованность данных и событийная архитектура<br>Рассматриваемые вопросы:<br>- транзакционные границы интеграционной операции;<br>- локальная и итоговая согласованность данных;<br>- проблема атомарной записи результата и публикации события;<br>- журнал исходящих событий как шаблон надёжной публикации;<br>- производитель, потребитель и брокер сообщений;<br>- устройство Apache Kafka, темы, разделы и группы потребителей;<br>- выбор событийного взаимодействия в интеграционном контуре. |
| 6 | Событийные контракты и поведение при отказах<br>Рассматриваемые вопросы:<br>- структура спецификации AsyncAPI;<br>- каналы, сообщения и схемы полезной нагрузки событий;<br>- семантика доставки «как минимум однократно»;<br>- повторная обработка и идемпотентный потребитель;<br>- нарушение порядка сообщений и проектные способы его учёта; |

| № п/п | Тематика лекционных занятий / краткое содержание |
|---|---|
| | - необрабатываемые сообщения и контролируемое завершение обработки;<br>- потеря ответа, недоступность участника и восстановление интеграционного контура. |
| 7 | Проверка контрактов и работоспособности интеграционного контура<br>Рассматриваемые вопросы:<br>- уровни проверок интеграционного решения;<br>- контрактные проверки схем, сериализации и обязательности полей;<br>- независимая проверка производителя и потребителя;<br>- интеграционные проверки в изолированной среде Testcontainers;<br>- проектирование отказных испытаний и контроль прикладных инвариантов;<br>- протокол воспроизведения повтора, задержки, недоступности и восстановления;<br>- критерии готовности интеграционного решения к выпуску. |
| 8 | Архитектурная эволюция распределённых систем<br>Рассматриваемые вопросы:<br>- распределённая система как совокупность независимо изменяющихся компонентов;<br>- зависимости между контрактами и ограничения архитектурной эволюции;<br>- обратная и прямая совместимость изменений;<br>- семантическое версионирование API и событийных контрактов;<br>- сосуществование версий и адаптация взаимодействия;<br>- непрерывность взаимодействия при обновлении компонентов;<br>- откат изменений и восстановление согласованного состояния;<br>- оценка архитектурных решений по устойчивости, сложности и стоимости сопровождения. |

## 4.2. Занятия семинарского типа.

### Лабораторные работы

| № п/п | Наименование лабораторных работ / краткое содержание |
|---|---|
| 1 | Декомпозиция интеграционного сценария расчётного контура<br>Студент анализирует описание запроса расчёта, согласования результата и фоновых интеграционных заданий. Он выделяет участников, операции, передаваемые данные, состояния обработки и внешние эффекты. Затем строит диаграммы последовательности для успешного сценария и сценария недоступности расчётного сервиса. Результатом становится реестр взаимодействий для последующего проектирования контрактов. |
| 2 | Обоснование способов интеграции участников<br>Студент сравнивает синхронный запрос, отложенную операцию и событийное уведомление для каждого взаимодействия из реестра. Он оценивает варианты по задержке, надёжности, связанности, сложности восстановления и сопровождения. На основе матрицы выбора определяет способ интеграции клиентского приложения, расчётного сервиса и сервиса фоновых заданий. Решение оформляет в виде краткого архитектурного документа. |
| 3 | Проектирование моделей данных и контракта ошибок<br>Студент определяет состав запроса расчёта, результата, детализации начислений и сведений о согласовании. Он описывает модели в JSON Schema, задаёт обязательность полей, типы, форматы и допустимые значения. Отдельно формирует каталог прикладных и технических ошибок с HTTP-статусами и машиночитаемыми кодами. Подготовленные схемы используются как части будущей спецификации API. |
| 4 | Разработка спецификации API первой версии<br>Студент собирает спецификацию OpenAPI для запуска расчёта, получения результата и передачи результата на согласование. Он включает идентификаторы запроса и операции, состояния обработки, ключ идемпотентности и описание ошибок. Затем проверяет спецификацию |

| № п/п | Наименование лабораторных работ / краткое содержание |
|---|---|
| | валидатором и наборе контрольных примеров. Утверждённая спецификация становится техническим заданием для программной реализации. |
| 5 | Реализация расчётного API по спецификации<br>Студент создаёт проект на Java и Spring Boot, подключает генерацию моделей из спецификации OpenAPI и реализует операции расчётного сервиса. Он связывает контрактные модели с внутренней расчётной логикой и настраивает возврат ответов и ошибок. После запуска сервиса выполняет вызовы по контрольным примерам и устраняет расхождения со спецификацией. Исходный код и конфигурацию размещает в репозитории. |
| 6 | Разработка клиентского адаптера расчётного сервиса<br>Студент генерирует клиентские модели по спецификации и реализует адаптер обращения к расчётному API. Он настраивает тайм-ауты, разбор ответов и преобразование ошибок сервиса в исключения клиентского приложения. Затем проверяет успешный расчёт, получение отложенного результата и обработку отказа вызова. Адаптер подключает к учебному клиентскому модулю без дублирования контрактных моделей. |
| 7 | Обеспечение идемпотентности операций в PostgreSQL<br>Студент проектирует таблицы запросов, операций и результатов обработки в PostgreSQL. Он добавляет уникальные ограничения для ключей идемпотентности и транзакционное сохранение результата расчёта. После этого выполняет повторные запросы с одинаковым и разными ключами и сравнивает состояние базы данных. Проверяет также повтор после искусственной потери первого ответа. |
| 8 | Контрактные проверки и испытание синхронного контура<br>Студент реализует автоматизированные проверки соответствия ответов схемам OpenAPI, обязательности полей и состава ошибок. Он запускает сервис и PostgreSQL в Testcontainers и воспроизводит интеграционные сценарии клиентского адаптера. Затем вносит временное нарушение в реализацию и убеждается, что проверка обнаруживает расхождение контракта. Итогом становится пакет проверок работоспособного синхронного контура. |
| 9 | Проектирование событийного контракта<br>Студент выделяет события о завершении расчёта и согласовании результата и определяет их производителей и потребителей. Он описывает каналы и сообщения в AsyncAPI, используя подготовленные ранее схемы данных. Для каждого события фиксирует идентификатор, время, версию и связь с расчётной операцией. Готовая спецификация передаётся в реализацию событийного канала. |
| 10 | Реализация производителя и потребителя Kafka<br>Студент разворачивает Apache Kafka в локальной контейнерной среде и создаёт необходимые темы. Он реализует публикацию события о завершении расчёта и потребитель сервиса фоновых заданий. Затем проверяет сериализацию сообщений, обработку ошибок и сохранение результата потребления. Для контроля использует сообщения, соответствующие спецификации AsyncAPI. |
| 11 | Атомарная публикация событий и идемпотентное потребление<br>Студент добавляет журнал исходящих событий в транзакцию сохранения расчётного результата и отдельный механизм их публикации. Он реализует регистрацию обработанных идентификаторов сообщений в потребителе. После этого проверяет отказ между записью результата и публикацией, а также повторную доставку одного события. По состоянию PostgreSQL и Kafka подтверждает отсутствие потери и дублирования прикладного результата. |
| 12 | Отказные испытания событийного канала<br>Студент составляет протокол испытаний для повтора сообщения, нарушения порядка, недоступности Kafka и остановки потребителя. Он воспроизводит сценарии управляемыми действиями в контейнерной среде и фиксирует состояние участников до и после отказа. Затем восстанавливает контур и проверяет сохранение прикладных инвариантов. Обнаруженные расхождения устраняет и повторяет проверки. |

| № п/п | Наименование лабораторных работ / краткое содержание |
|---|---|
| 13 | Проектирование второй версии контрактов<br>Студент расширяет результат расчёта валютой, детализацией, источниками и правилами округления. Он определяет совместимые и несовместимые изменения и формирует спецификации OpenAPI и AsyncAPI второй версии. Затем строит матрицу существующих производителей и потребителей и назначает необходимые адаптеры. Отдельно планирует двойную публикацию событий и последующее прекращение первой версии. |
| 14 | Реализация совместимости версий<br>Студент реализует вторую версию API и адаптеры преобразования сообщений между версиями. Он настраивает одновременную публикацию событий обеих версий без изменения существующих потребителей. Затем выполняет контрольные сообщения и проверяет смысловую эквивалентность результатов, повторную обработку и нарушение порядка. Матрицу совместимости дополняет фактическими результатами проверок. |
| 15 | Сборка и проверка интеграционного выпуска<br>Студент собирает расчётный сервис, клиентский адаптер, сервис фоновых заданий, PostgreSQL и Kafka в единый контейнерный стенд. Он выносит параметры подключения и версии контрактов в конфигурацию и подготавливает сценарий запуска. После развёртывания выполняет полный набор контрактных, интеграционных и отказных проверок. По журналу запуска определяет готовность выпуска и устраняет найденные дефекты. |
| 16 | Воспроизводимое развёртывание и оформление портфолио<br>Студент разворачивает выпуск на чистом учебном стенде под Astra Linux или РЕД ОС по собственной инструкции. Он повторно запускает проверки и демонстрирует один отказный сценарий с последующим восстановлением. Затем оформляет в Р7-Офис архитектурное решение, протокол испытаний, матрицу совместимости и план вывода первой версии. Завершает работу индивидуальной защитой интеграционного контура и пакета документации. |

4.3. Самостоятельная работа обучающихся.

| № п/п | Вид самостоятельной работы |
|---|---|
| 1 | Изучение рекомендованной литературы. |
| 2 | Подготовка к лабораторным работам. |
| 3 | Подготовка к промежуточной аттестации. |
| 4 | Подготовка к текущему контролю. |

5. Перечень изданий, которые рекомендуется использовать при освоении дисциплины (модуля).

| № п/п | Библиографическое описание | Место доступа |
|---|---|---|
| 1 | Лёзин, И. А. Разработка веб-приложений с использованием Spring Boot : учебное пособие / И. А. Лёзин, И. В. Лёзина. — Самара : Самарский университет, 2023. — 96 с. — ISBN 978-5-7883-1989-6. — Текст : электронный | Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/406517 (дата обращения: 11.11.2026). |

| | | |
|---|---|---|
| 2 | Иванов, С. А. Системная интеграция приложений : учебно-методическое пособие / С. А. Иванов. — Санкт-Петербург : ИЭО СПбУТУиЭ, 2020. — 91 с. — ISBN 978-5-94047-823-2. — Текст : электронный | Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/246401 (дата обращения: 11.11.2026). |
| 3 | Евдошенко, О. И. Проектирование, разработка и использование баз данных в PostgreSQL : учебное пособие / О. И. Евдошенко. — Москва : РТУ МИРЭА, 2025. — 72 с. — ISBN 978-5-7339-2574-5. — Текст : электронный | Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/504835 (дата обращения: 11.11.2026). |

6. Перечень современных профессиональных баз данных и информационных справочных систем, которые могут использоваться при освоении дисциплины (модуля).

ЭБС «Лань» — https://e.lanbook.com/ (дата обращения: 29.10.2026).

OpenAPI Specification — https://spec.openapis.org/oas/latest.html (дата обращения: 29.10.2026).

AsyncAPI Specification — https://www.asyncapi.com/docs/reference/specification/latest (дата обращения: 29.10.2026).

JSON Schema Documentation — https://json-schema.org/docs (дата обращения: 29.10.2026).

Spring Boot Reference Documentation — https://docs.spring.io/spring-boot/index.html (дата обращения: 29.10.2026).

Apache Kafka Documentation — https://kafka.apache.org/documentation/ (дата обращения: 29.10.2026).

PostgreSQL Documentation — https://www.postgresql.org/docs/ (дата обращения: 29.10.2026).

Testcontainers for Java — https://java.testcontainers.org/ (дата обращения: 29.10.2026).

Docker Documentation — https://docs.docker.com/ (дата обращения: 29.10.2026).

Arenadata Documentation — https://docs.arenadata.io/ (дата обращения: 29.10.2026).

7. Перечень лицензионного и свободно распространяемого программного обеспечения, в том числе отечественного производства, необходимого для освоения дисциплины (модуля).

Операционные системы: Astra Linux / РЕД ОС / ALT Linux.

Java-платформа: Liberica JDK 17 LTS / Eclipse Temurin или OpenJDK 17.

Среда разработки: IntelliJ IDEA Community Edition / VSCodium.

Каркас приложений: Spring Boot, Spring Web, Spring for Apache Kafka.

Системы сборки: Apache Maven / Gradle.

Управление версиями: Git, GitFlic / GitLab Community Edition.

СУБД: PostgreSQL / Postgres Pro.

Администрирование данных: DBeaver Community Edition.

Проектирование синхронных API: Swagger Editor, Swagger UI, OpenAPI Generator.

Проектирование событийных API: AsyncAPI Studio, AsyncAPI CLI, AsyncAPI Generator.

Валидация данных: JSON Schema Validator.

Событийная интеграция: Apache Kafka / Arenadata Streaming.

Наблюдение за сообщениями: Kafdrop / AKHQ.

Проверка HTTP API: Hoppscotch, curl, REST Assured.

Автоматизированное тестирование: JUnit 5, Testcontainers for Java, Schemathesis.

Контейнерное развёртывание: Docker Engine и Docker Compose / Podman и Podman Compose.

Проектная графика: diagrams.net.

Офисные пакеты: Р7-Офис для подготовки архитектурных решений, протоколов испытаний и презентаций.

Документирование: Markdown и Git для ведения спецификаций, архитектурных решений и инструкций совместно с программным кодом.

8. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине (модулю).

Учебные аудитории для проведения учебных занятий, оснащенные компьютерной техникой и наборами демонстрационного оборудования.

Для лабораторных занятий – наличие персональных компьютеров вычислительного класса.

9. Форма промежуточной аттестации:

Зачет в 6 семестре.

10. Оценочные материалы.

Оценочные материалы, применяемые при проведении промежуточной аттестации, разрабатываются в соответствии с локальным нормативным актом РУТ (МИИТ).

Авторы:

старший преподаватель кафедры
«Цифровые технологии управления
транспортными процессами»
Института железнодорожного
транспорта И.С. Разживайкин

Согласовано:

Заведующий кафедрой ЦТУТП В.Е. Нутович

Директор ЦС IT - программ ВИШ О.Б. Проневич

Председатель учебно-методической
комиссии Н.А. Андриянова