**1. Процесс разработки**

**Обязательно**

Знать

* Типы ПО и их особенности (системное/прикладное/инструментальное, индивидуальное/групповое, web/desktop/app).
* Жизненный цикл разработки программного обеспечения (Software development lifecycle - SDLC): сбор и анализ требований, проектирование, разработка, тестирование, внедрение, сопровождение.
* Основные модели управления разработкой ПО (водопадные, итерационные). Основные гибкие методологии (Agile, Scrum, Kanban).
* Основные участники команды разработки и сопровождения IT-проектов, их роли (разработчик, аналитик, тестировщик, product и project менеджер, сетевой инженер, DevOps). Разные виды аналитиков и чем они занимаются (системный, бизнес, продуктовый, веб, BI, аналитик данных).
* Виды документации и их назначение (BRD, FSD/SRS, руководства пользователя, инструкции, регламенты, база знаний и т.д.).
* Понятие «фреймворк».
* Суть концепция CI/CD.
* Общее представление о системе контроля версий.

Уметь

* Выстраивать взаимодействие и совместную работу с заказчиками, командой, экспертами и другими участниками процесса создания, развития и сопровождения проектов.
* Планировать и выстраивать процесс работы; выделять, декомпозировать и приоритизировать задачи, управлять сроками и рисками.
* Создавать разные виды проектной документации.

**Продвинутый уровень**

Знать

* Прочие модели, методологии, подходы к управлению разработкой ПО (инкрементная, V-образная, Domain-driven design (DDD), Lean и т.д.).

Уметь

* Планировать процесс разработки фич и проектов, декомпозировать сложные процессы и задачи, ставить задачи другим участникам проекта.
* Руководить реализацией отдельной фичи, руководить проектом, руководить другими сотрудниками. Планировать ресурсы. Управлять рисками.

**2. Работа с требованиями**

**Обязательно**

Знать

* Что такое требования к разработке ПО. Виды требований (бизнес/пользовательские/системные, функциональные/нефункциональные и т.д.).
* Источники требований, способы и инструменты сбора требований:
  + интервью, опросы, анкетирование, наблюдение за процессом
  + анализ внутренних документов компании (бэклог, планы развития, обращения в тех.поддержку и т.п.)
  + анализ требований законодательства и других внешних для компании документов
  + анализ текущих решений (собственных и внешних), систем-аналогов, ранних версий, прототипов
  + фокус-группы, мозговой штурм
  + анализ предметной области
* Этапы работы с требованиями:
  + сбор требований
  + анализ, выстраивание приоритетов и устранение противоречий, согласование
  + моделирование системы, описание процессов
  + ревью и итоговое согласование
  + сопровождение и управление изменениями
  + тестирование функционала, презентация результатов
* Структура документов с требованиями (проблемы и потребности, цели и задачи, заинтересованные лица, пользователи системы, описание как есть и как будет, ролевая модель, интеграция с другими системами, этапы реализации, границы проекта, ограничения, риски, глоссарий и т.д.).
* Форматы описания требований: текстовый, графический и т.п.

Уметь

* Выделять заказчиков, лиц принимающих решения, заинтересованных лиц проекта, экспертов, пользователей, специалистов создания, внедрения и сопровождения данного проекта.
* Выделять источники и собирать требования разными способами (особенно через интервью).
* Выявлять истинные причин появления проекта и требований:
  + почему появилась задача: какую проблему решаем или что хотим улучшить
  + зачем это реализовывать: ради какого измеряемого результата
  + определение цены и ценности: какой результат получить реально в рамках данного проекта и имеющихся ресурсов, устроит ли он, стоит ли он потраченного времени и других ресурсов.
  + определять границы проекта: что будет, а что не будет реализовано в текущем проекте, выделать MVP
* Описывать разные виды требований, в разных форматах.
* Выделять и описывать user story, use cases (основные и альтернативные сценарии).
* Управлять требованиями: детализировать, проверять на однозначность, полноту, измеримость, проверяемость и осуществимость, проводить трассировку, устранять противоречивость, приоритизировать, согласовывать, реализовывать изменения, проводить ревью.

**Продвинутый уровень**

Знать

* Расширенные знания стандартов описания требований: ГОСТ 19, ГОСТ 34, EARS (The Easy Approach to Requirements Syntax) и т.п.

Уметь

* Выделять и описывать job story (Jobs-To-Be-Done).
* Строить Impact Map (структура «зачем, кто, как, что»).

**3. Моделирование систем**

**Обязательно**

Знать

* Основные виды архитектур, их описание, преимущества и недостатки, когда используются (локальная, монолитная, клиент/сервер/БД, SOA: сервис-ориентированная, MSA: микросервисная).
* Понятия «хореография» и «оркестрация».
* Понятия «фронтенд» и «бэкенд».
* Описание процессов с помощью блок-схем (Flowchart).
* Описание процессов в нотации BPMN
* Описание процессов и системы в нотации UML:
  + Диаграмма вариантов использования - Use Case Diagram
  + Диаграмма активностей - Activity Diagram
  + Диаграмма состояний - State Machine Diagram
  + Диаграмма последовательностей - Sequence Diagram

Уметь

* Моделировать системы и описывать их с помощью разных нотаций (Flowchart, BPMN, UML) и разных способов (тексты, таблицы, схемы, диаграммы и т.п.).
* Описывать бизнес-процессы, поведение пользователей и отклик системы, системные функции (процессы, осуществляемые системой без участия пользователя).

**Продвинутый уровень**

Знать

* Более глубокие знания архитектур (многослойная, многоуровневая, MVC: Model-View-Controller, клиент-серверная, файл-серверная, облачная, событийно-ориентированная, микроядерная, модульный монолит, peer-to-peer и т.д.)
* Особенности реализации web, desktop и мобильных приложений. Кроссплатформенная разработка.
* Расширенные знания нотации BPMN и UML.
* Знание других нотаций, стандартов, фреймворков (IDEF0, IDEF3, EPC, DMN, VAD, SIPOC, BABOK и т.п.).
* Модель С4 архитектуры программного обеспечения.

Уметь

* Описывать более сложные процессы в нотации BPMN.
* Описывать более сложные процессы в нотации UML, использовать прочие виды диаграмм (Deployment Diagram, Component Diagram и т.д.).

**4. Модели данных**

**Обязательно**

Знать

* Что такое концептуальная, логическая и физическая модели данных.
* Структурированные, неструктурированные и слабоструктурированные данные.
* Что такое и как между собой связаны: сущности (объекты), атрибуты, связи.
* Основные принципы ООП. Понятия «класс», «объект», «экземпляр».
* Типы баз данных (реляционные, объектно-реляционные, нереляционные - NoSQL, колоночные, текстовые). Когда какие используются. Популярные систем управления баз данных (СУБД) для каждого типа.
* Как организованы реляционные базы данных, правила проектирования:
  + Основные принципы реляционных баз данных
  + Типы данных
  + Способы реализации связей
  + Нормализация: что это, зачем нужна, 3 формы
  + Первичный ключ, составной первичный ключ, внешний ключ, суррогатный ключ
  + NULL и пустые значения
  + Ограничения (NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK, DEFAULT, INDEX, AUTO INCREMENT)

Уметь

* Создавать ER-диаграммы.
* Проектировать простые базы данных.

**Продвинутый уровень**

Знать

* Более глубокие знания в проектировании реляционных БД
  + Представление (виртуальная таблица). Виды представлений.
  + Индексы
  + Триггеры
  + Транзакции. ACID-требования к транзакциям и CAP-теорема.
  + Временные таблицы (таблицы для промежуточных данных)
  + Хранимые процедуры
  + Сериализация данных
  + Что такое подмножества и какие бывают (DDL, DQL, DML, DCL, TCL).
  + SQL-инъекции
  + Чем отличаются и когда какую СУБД лучше выбрать (PostgreSQL, MySQL, Oracle, MS SQL, MongoDB, ClickHouse, DB2, Greenplum, SQLite, Elasticsearch, Cassandra и т.д.)
* Правила проектирования нереляционных баз данных.
* Понятие Object-relational mapping - ORM (доступ к реляционной базе данных с помощью объектно-ориентированного языка)

Уметь

* Описывать классы в UML: Class Diagram.
* Проектировать более сложные реляционные базы данных.
* Проектировать нереляционные базы данных.
* Составлять логические модели по существующей базе данных.
* Осуществлять реинжиниринг модели данных.

**5. Пользовательский интерфейс (UI/UX)**

**Обязательно**

Знать

* Основы UI/UX, правила построения интерфейсов (принципы, этапы разработки, критерии качества).
* Типовые элементы (экранные формы, модальные окна, хлебные крошки, пагинация и т.д.).
* Понимание принципов адаптивной верстки.
* Понимание, что такое «клиентский путь».

Уметь

* Создавать наброски и схемы графических интерфейсов системы.
* Взаимодействовать с UI/UX специалистами.

**Продвинутый уровень**

Знать

* Расширенные знания UI/UX. В том числе в разных направления (web, app и т.д.).

Уметь

* Создавать макеты графического интерфейса системы (с помощью Figma или других специальных инструментов).
* Создавать интерактивные прототипы графических интерфейсов.
* Разрабатывать карты клиентского пути (CJM - Customer Journey Map).

**6. Интеграция систем**

**Обязательно**

Знать

* Виды интеграций информационных систем (API, шина данных - ESB/MOM/MQ, общая база данных, файловый обмен). Их описание, преимущества и недостатки разных способов, что когда используется.
* Pull-модель (первоначальный запрос производится клиентом) и push-модель (данные поступают от поставщика к пользователю на основе установленных контрактов).
* Синхронное, асинхронное и реактивное взаимодействие.
* Концепции stateful и stateless (с сохранением и без сохранения состояния на стороне сервера).
* Основы синтаксиса JSON и XML.
* Типы API (REST, SOAP, JSON:API, GraphQL, RPC, API нативных библиотек), их общее описание.
* Веб-сервисы. Webhook.
* Протокол HTTP
  + структура запроса и ответа (стартовая строка, HTTP-заголовки, тело сообщения)
  + основные методы в HTTP запросах, их назначение (GET, POST, PUT, DELETE, PATCH), концепция CRUD
  + структура URL (протокол, хост, порт, путь до ресурса, запрос/параметры)
  + коды состояния (1хх, 2хх, 3xx, 4xx, 5xx)
  + как передать html документ, json-файл
  + что такое HTTPS, отличие от HTTP

Уметь

* Проектировать интеграционные взаимодействия:
  + диаграммы потоков данных (DFD)
  + диаграммы последовательности (Sequence Diagram)
  + описание передаваемых и принимаемых данных
  + обработка ошибок и нештатных ситуаций, журналирование
  + ограничения на интеграцию
  + требования к качеству интеграций
  + описание преобразований данных
  + регламент передачи данных
* Разрабатывать требования к интеграции систем через файловый обмен.
* Разрабатывать требования к интеграции систем через общую базу данных.

**Продвинутый уровень**

Знать

* Уровни сетевого взаимодействия: модель OSI, TCP/IP, UDP, FTP, SSH, SFTP, WebSocket и т.п.
* Какие есть сложности и проблемы при интеграции систем. Закон дырявых абстракций.
* Что такое идемпотентность. Какие HTTP методы являются идемпотентными, а какие нет, почему.

Уметь

* Разрабатывать требования интеграции систем через GraphQL и JSON:API.

**7. Интеграция систем: архитектура REST**

**Обязательно**

Знать

* Принципы REST архитектуры.
* Отличие от SOAP.
* Особенности применения REST в HTML (нельзя отправлять PUT и DELETE запросы из HTML-формы).
* JSON Schema.

Уметь

* Понимать документацию с описанными REST API (в том числе в Swagger).
* Проектировать и описывать интеграции REST API.
* Работать с Postman

**Продвинутый уровень**

Знать

* Отличие от JSON:API.

Уметь

* Описывать интеграции REST API через OpenAPI и Swagger
* Работать с cURL.

**8. Интеграция систем: протокол SOAP**

**Обязательно**

Знать

* Основы протокола SOAP. Структура сообщения (XML файла). WSDL
* XSD схема (XML Schema).

Уметь

* Понимать документацию с описанными SOAP API.

**Продвинутый уровень**

Знать

* Расширенные знания протокола SOAP (пространство имен, индикаторы элементов, XSLT, XPATH и т.д.).

Уметь

* Проектировать и описывать интеграции SOAP API, XSD схемы.
* Работать с SOAP UI и Postman (для SOAP).

**9. Интеграция систем: шина данных (ESB, MOM, MQ)**

**Обязательно**

Знать

* Что из себя представляет ESB (Enterprise Service Bus - сервисная шина предприятия), MOM (Message-oriented Middleware - ПО, ориентированное на обмен сообщениями в распределенном окружении), MQ (Message Queue - очередь сообщений, брокер сообщений).
* Отличие ESB от MQ. Отличие ESB от ETL.
* Понятия «топик» (издатель-подписчик) и «очередь» (отправитель-получатель).
* Какие брокеры сообщений чаще всего используются (RabbitMQ и Kafka).

Уметь

* Понимать документацию с описанием интеграций через шину.
* Описать документацию при использовании интеграции через шину.

**Продвинутый уровень**

Знать

* Разница между RabbitMQ и Kafka. Когда что лучше выбрать.
* Как брокер сообщений (RabbitMQ и Kafka) гарантирует доставку сообщений.

**10. Анализ данных**

**Обязательно**

Знать

* Основы анализа данных.
* Знать что такое Big Data, BI, Data Science, ML.

Уметь

* Анализировать данные, находить ответы на вопросы, формировать отчеты, уметь их презентовать.
* Работать с простыми SQL запросами
  + CRUD-операции: SELECT, UPDATE, INSERT, DELETE
  + Создание и удаление таблиц: CREATE, DROP, TRUNCATE
  + Ограничение и сортировка: WHERE, LIMIT, DISTINCT, ORDER BY
  + Дополнительные условия: = != <> > < AND, OR, BETWEEN, IN, IS NULL, IS NOT NULL, LIKE, NOT LIKE
  + Арифметические операции: + - \* / %
  + Агрегирование данных: GROUP BY, AS, AVG, COUNT, MAX, MIN, SUM, HAVING
  + Вложенные запросы (в части SELECT, FROM, WHERE)
  + Объединение запросов: UNION, UNION ALL
  + Объединение таблиц: JOIN (INNER, LEFT, RIGHT, FULL, CROSS)
  + Комментарии

**Продвинутый уровень**

Знать

* Основы статистики.
* Основы продуктовой аналитики (продуктовые и маркетинговые метрики, CustDev, А/В тестирование, Unit-экономика и т.д.)
* Что такое OLTP (обработка транзакций в реальном времени), OLAP (интерактивный анализ данных - кубы), ETL (извлечение, преобразование, загрузка), ELT (извлечение, загрузка, преобразование), DWH (хранилище данных).
* Понимание принципов построения хранилищ данных.
* Основные системы аналитики данных.

Уметь

* Проводить более сложный анализ данных, находить инсайты, аномалии. Формировать дашборды, визуализировать данные.
* Работать с более сложными SQL запросами
  + Сложные составные запросы (в том числе с EXISTS, ANY, ALL, CASE, IF)
  + Встроенные функции: ROUND, DATE, TIME, DATETIME, SUBSTR и т.д.
  + Оконные функции (OVER)
* Анализировать данные с помощью Excel.
* Работать с BI системами (Power BI, Tableau и т.п.).
* Работать с системами web и app аналитики (Google Analytics, Яндекс.Метрика и т.п.).
* Работать с Python для анализа данных (в том числе со специальными библиотеки NumPy, Pandas и т.д.).

**11. Безопасность**

**Обязательно**

Знать

* Что такое аутентификация, примеры способов аутентификации (пароль, ЭЦП, SMS, push уведомление, биометрия, многофакторная), что такое идентификация.
* Что такое авторизация, ролевая модель информационной системы.
* Что такое хеширование, как и где применяется (особенно для паролей).
* Что такое электронная подпись. Зачем и как используется.

**Продвинутый уровень**

Знать

* Что такое криптография, для чего используется. Симметричное и асимметричное шифрование. Открытый и закрытый ключи. TLS/SSL в HTTPS.
* Контрольная сумма: что это, как используется.
* Основные схемы и протоколы аутентификации (базовая аутентификация, аутентификация по cookies, аутентификация по предъявлению цифрового сертификата, аутентификация с помощью ключа API, OpenID/OAuth/JWT и т.д.)
* Что такое верификация и валидация.
* Принципы работы электронной подписи, виды (простая, неквалифицированная, квалифицированная), их отличия.
* Основные уязвимости веб сервисов и мобильных приложений.

**12. Тестирование**

**Обязательно**

Знать

* Процесс тестирования, тест кейсы и чек листы.

Уметь

* Разрабатывать критерии и процесс проведения приемочного тестирования.
* Организовывать и проводить приемочное тестирование.

**Продвинутый уровень**

Знать

* Виды, подходы, инструменты тестирования.

Уметь

* Написать тест кейсы и чек листы (все позитивные и основные негативные сценарии).
* Тестировать функционал (все позитивные и основные негативные сценарии).

**13. Основы программирования**

**Обязательно**

Знать

* Какие основные языки программирования существуют, где применяются.
* Основы программирования (переменные, операторы, ветвление, циклы, функции и т.п.).
* Основы языка разметки документов HTML.

Уметь

* Написать простую программу на любом ООП языке программирования.

**Продвинутый уровень**

Знать

* Более глубокое знание хотя бы одного из популярных ООП языков.
* Понимание основ HTML/CSS/JavaScript.
* Понимание ООП в программировании.

Уметь

* Написать более сложную программу на любом ООП языке программирования.