GeekBrains

Основы диаграмм состояний

Диаграммы состояний

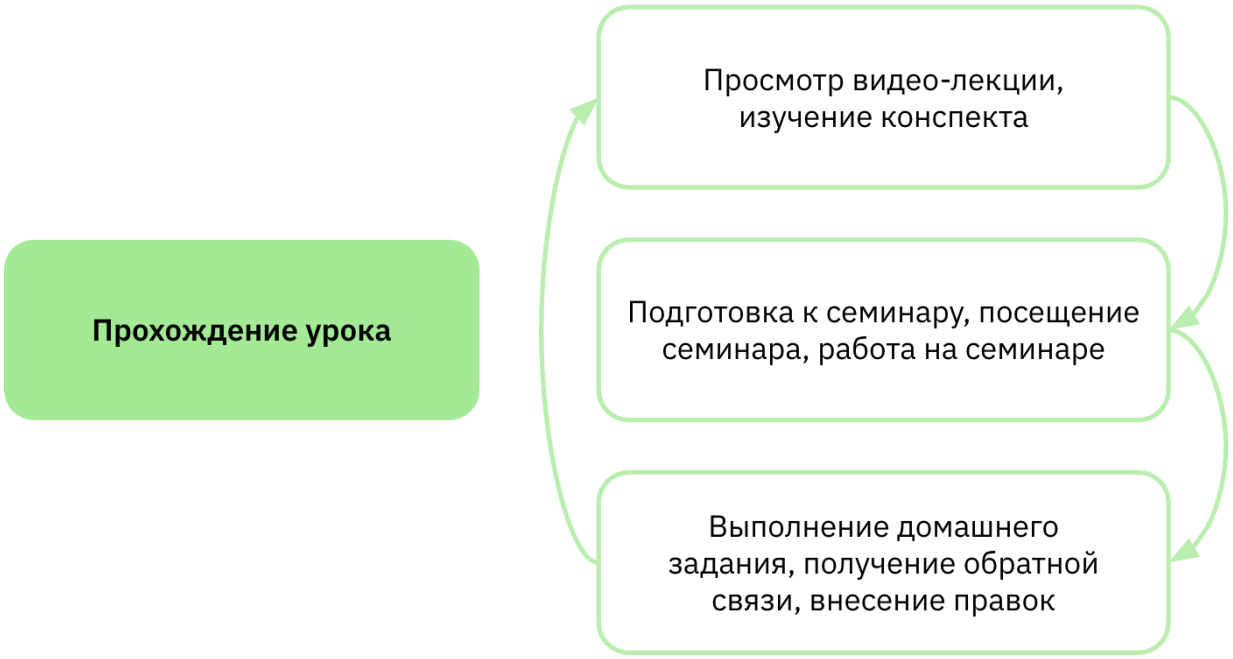
Оглавление

Введение

Все вокруг нас находится в состояниях, которые меняются, создавая сложную систему. Например, когда мы изучаем урок на платформе GeekBrains, проходим несколько этапов:

1. Получаем доступ к видеолекции, конспекту и презентации. Изучаем их.
2. Переходим к практике: посещаем семинары и отрабатываем навыки с коллегами.
3. Приступаем к домашней работе: закрепляем материал и ждем обратную связь от преподавателей, чтобы доработать решение и перейти к следующему уроку.
4. Цикл повторяется для каждого урока.

Как устроен урок в Geekbrains



Таким образом, изучая материалы к каждому уроку, мы переходим из одного состояния в другое. Этот процесс можно представить в виде диаграммы состояний.

Диаграммы состояний помогают лучше понять процессы в нашей жизни: будь то рутинные обязанности, информационные системы или рабочие проекты. Наша

жизнь полна перемен и событий, которые мы не можем контролировать и отслеживать, как они влияют на разные процессы. Поэтому мы создали этот курс — он будет полезен и в повседневной жизни, и на работе.

На первых двух лекциях мы изучим инструменты, которые помогают описать нашу систему максимально детально. Учтем важные детали проектирования систем, научимся строить простые диаграммы состояний и проектировать конечные автоматы.

В третьем уроке курса разберемся с диаграммами состояний и будем их описывать при помощи нотации UML.

Мы поработаем с рядом инструментов, среди которых конечные автоматы, mind map, диаграммы состояний и многое другое. Все они пригодятся вам в работе, помогут внести ясность в процессы и даже выстроить их.

Азы диаграмм состояний и основ моделирования будут полезны разным специалистам:

- **Бизнес-аналитикам**, так как рассматриваемые инструменты — это фундамент при оформлении спецификаций на продукт.
- **Системным аналитикам**, так как их инструментарий пересекается с бизнес-анализом, а часто системный аналитик и бизнес-аналитик — один и тот же человек в компании.
- **Тестировщикам**, так как понимание состояний объекта и системы существенно помогает при тестировании программного обеспечения: для составления тест-кейсов и при проверке пограничных состояний.
- **Разработчикам**, так как они разрабатывают программное обеспечение согласно информации от аналитиков. Кроме того, более опытным разработчикам иногда нужно продумывать архитектуру программы. Результат их работы должен быть зафиксирован с помощью артефактов, которыми часто выступают диаграммы состояний.
- **Владельцу продукта**, так как он должен уметь читать документацию по проекту и согласовывать наработки, которые зафиксировали аналитики. Понимание принципов проектирования систем может ускорить процесс разработки продукта.
- **Менеджеру проектов**, так как он часто выступает связующим звеном между бизнесом и командой разработки. А формализованным языком общения между этими двумя командами являются артефакты бизнес-анализа, в

частности диаграммы состояний. Важно в них разбираться, чтобы грамотно управлять взаимодействием со стейкхолдерами проекта.

- **Маркетологам**, так как они работают с воронками продаж, а воронки продаж со статусами и состояниями клиентов — это и есть диаграмма состояний.

На этой лекции вы узнаете:

- Что такое состояние объекта и как с ним работать.
- Что такое диаграммы состояний.
- Как различные роли взаимодействуют с диаграммами состояний.
- Как можно визуализировать полученную информацию.

Термины, используемые в лекции

Диаграммы состояний — это инструмент, помогающий описать различные состояния объекта, системы или продукта. Часто для его разработки используют нотации (UML, BPMN и другие), чтобы описывать состояние стандартизировано.

Требования — это спецификация того, что должно быть реализовано. В них описано поведение системы, ее свойства или атрибуты. Они могут служить ограничениями в процессе разработки системы.

Стейкхолдер — лицо, заинтересованное в проекте и его результатах или просто имеющее возможность воздействовать на проект.

Что такое состояние и диаграмма состояний

Состояние — это текущее положение объекта или системы. Может быть физическим или логическим. Описывается с помощью свойств и параметров.

Иногда нам нужно реализовать простейшую программу с несколькими состояниями: например, программу для работы со шлагбаумом. Шлагбаум может находиться в двух состояниях: «открыт» и «закрыт». Этого достаточно, чтобы выполнить его основную задачу. Похожий пример — чайник с двумя режимами: «включен» и «выключен».

Более сложными состояниями можно описать машину: например, ее скорость, направление движения, количество топлива в баке и так далее. Состояние компьютера может быть описано как нагрузка на процессор, доступное количество оперативной памяти и так далее.

В некоторых случаях состояние может меняться во времени, на него могут влиять внешние условия или входные данные. Состояние машины будет меняться в зависимости от того, как она используется (например, движется или стоит) и от того, как изменяется уровень топлива в баке.

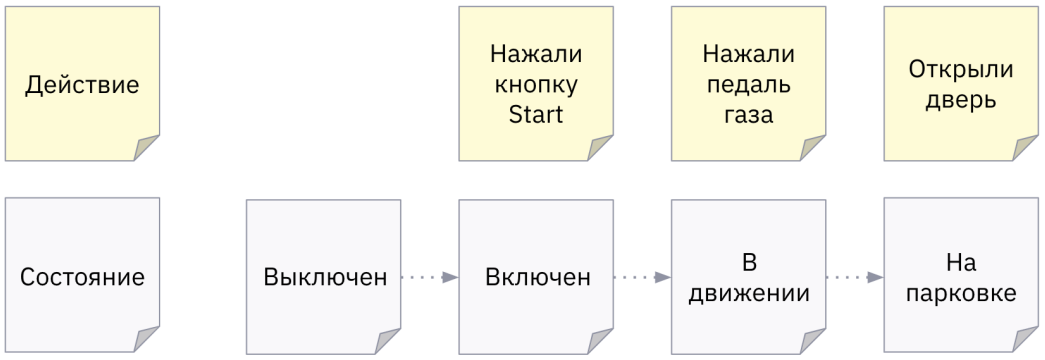
Для описания всех состояний используется **диаграмма состояний** — один из мощнейших инструментов для моделирования и анализа систем, которые изменяются во времени. Диаграмма состояний позволяет визуализировать возможные состояния системы и переходы между ними, а также описывать действия, которые происходят в каждом состоянии. Все это помогает лучше понять поведение системы и принять более эффективные решения при проектировании и исправлении ошибок.

Подробнее рассмотрим еще несколько состояний машины:

- выключена,
- включена,
- в движении,
- на парковке.

Диаграмма состояния машины будет описывать переходы между этими состояниями, а также действия, которые можно выполнить в каждом состоянии. Например, когда машина в состоянии «Включена», можно нажать кнопку старта, чтобы перейти в состояние «В движении».

Пример. Серые стикеры — это состояния, желтые — действия, которые переводят в состояние.



Когда машина в состоянии «На парковке», можно открыть дверь, чтобы выйти из нее. Тогда мы перейдем в другое состояние системы.

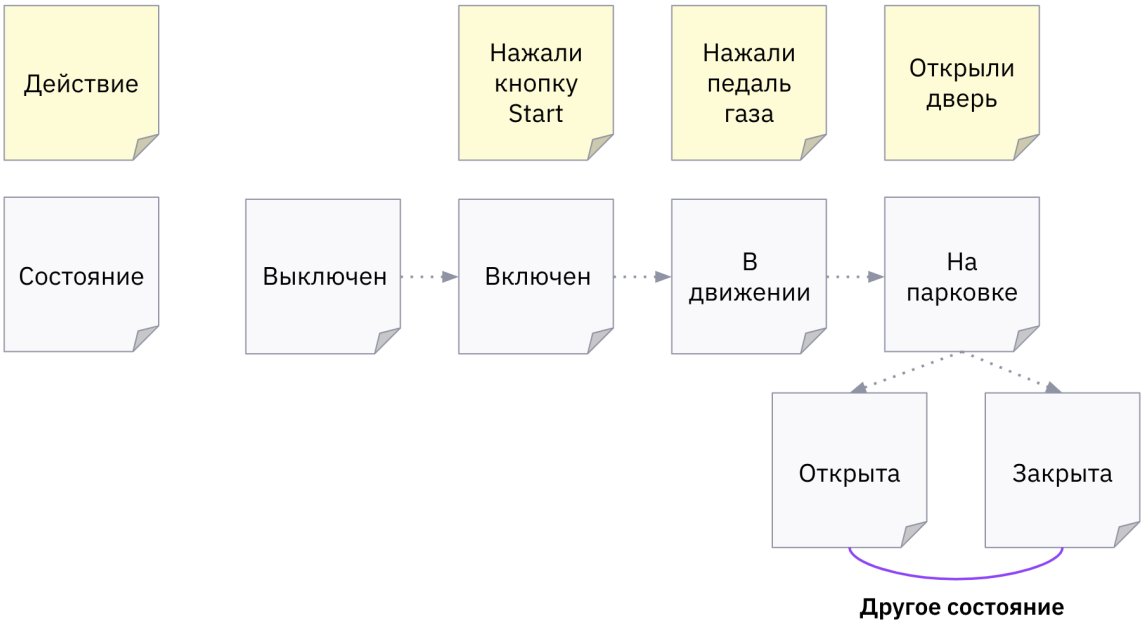


Диаграмма состояний может помочь обозначить все возможные переходы и действия. Это полезно при проектировании и разработке системы или продукта. Один и тот же продукт можно описать с разных точек зрения, используя разные состояния. Все зависит от того, какую часть системы мы хотим реализовать.

Рассмотрим на примере информационной системы. Она может находиться в нескольких состояниях:

- остановлена,
- запущена,
- перезагружается,
- обновляется,
- ошибка.

Например, когда система в состоянии «Остановлена», можно нажать кнопку «Запустить» чтобы перейти в состояние «Запущена». А когда система в состоянии «Обновляется», можно отслеживать процесс обновления и перейти в состояние «Запущена» после завершения обновления.

Как выглядит диаграмма состояния

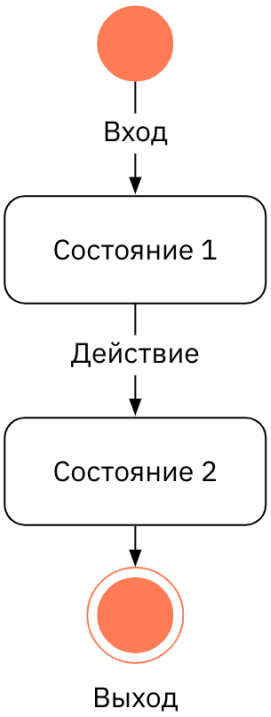
Простая диаграмма состояний состоит из следующих элементов:

Состояние — фаза или конфигурация, в которой находится объект или система в определенный момент. В диаграмме представляется в виде прямоугольника со скругленными углами и с названием состояния.

Переход — механизм, который позволяет перейти из одного состояния в другое. Обозначается стрелкой.

Вход — условие, которое может вызвать переход из одного состояния в другое. Это некий старт событий или точка входа в состояние. Например, при нажатии кнопки «Старт» система может перейти из состояния «Остановлена» в состояние «Запущена». Вход в этом случае — событие «Нажатие кнопки старт».

Выход — условие, которое может вызвать переход из одного состояния в другое. Завершающий этап всех состояний. Например, когда система переходит из состояния «Запущена» в состояние «Остановлена».



На следующих уроках мы разработаем полноценную диаграмму состояний и посмотрим ее применение на разных объектах в системе.

Диаграмма состояний в контексте жизненного цикла разработки

Чтобы понять, как разные роли работают с разными состояниями системы, посмотрим на весь процесс разработки программного обеспечения с точки зрения SDLC-модели.

Жизненный цикл разработки ПО (SDLC, System Development Life Cycle) — это методология разработки, описывающая шаги, которые нужно пройти для создания, внедрения и поддержки информационной системы.

Жизненный цикл разработки SDLC состоит из этапов:

1. **Анализ требований и планирование** — изучение потребностей пользователей и анализ их возможных решений.
2. **Проектирование и реализация** — создание спецификаций, диаграмм и других документов, описывающих архитектуру и дизайн системы.
3. **Тестирование** — проверка и исправление найденных ошибок и недостатков.
4. **Развертывание и поддержка** — установка продукта у заказчика и его поддержка.

В каждом из этих этапов участвуют разные роли, отвечающие за стадию продукта: продакт-менеджер, проджект-менеджер, аналитики, разработчики, тестировщики. Вы можете заметить, что эта схема сама по себе является диаграммой состояния для программного продукта.

Теперь рассмотрим, как каждая роль взаимодействует с диаграммами состояний.

Продакт- и проджект-менеджерам диаграммы состояний помогают охватывать состояния, которые принимает продукт, и действия, которые можно произвести в каждом состоянии. Это полезно, например, чтобы описать поведение приложения или сайта, в котором пользователь может интерактивно перемещаться между разными экранами или состояниями. Диаграммы состояний помогают обозначить,

как состояние меняется, что происходит при переходе из одного состояния в другое, какие действия могут быть вызваны в определенном состоянии.

Системным и бизнес-аналитикам диаграммы состояний помогут в анализе состояний и переходов в бизнес-процессах. Диаграммы состояний будут полезны для описания и анализа работы бизнес-процессов: например, процесса продажи товаров или услуг, процесса обработки заявок или процесса клиентского обслуживания. Бизнес-аналитики могут использовать диаграммы состояний, чтобы визуализировать работу процесса, выявить проблемные участки и понять, как можно улучшить его эффективность.

Тестировщики с помощью диаграммы состояний могут визуализировать состояния и события, которые происходят в системе. Это помогает лучше понять, как работает система и какие сценарии тестирования нужно проверять.

Например, диаграмма состояния показывает, что система может находиться в состояниях «Вход в систему», «Режим просмотра», «Режим редактирования». Тестировщик может использовать эту информацию, чтобы определить, какие тесты нужно проводить для каждого состояния: например, проверить правильность перехода из одного состояния в другое или найти ошибки в состояниях.

Разработчикам диаграммы состояний помогают описывать состояния и переходы между состояниями системы или компонента. Это полезно для разработки комплексных или изменяющихся систем со множеством состояний и переходов между ними. Используя диаграмму состояний, разработчики могут визуально отображать и сохранять информацию о состояниях системы и переходах между ними. Все это помогает лучше понимать и контролировать работу системы.

Диаграммы состояний также полезны для описания и документирования поведения системы для других разработчиков или стейкхолдеров.

Здесь также стоит рассказать про конечные автоматы.

Конечный автомат (Finite State Machine, FSM) — это математическая модель, которая используется для описания и имитации динамических систем с ограниченным количеством состояний.

Конечный автомат можно использовать для моделирования и реализации различных систем, включая компьютерные программы, электронные устройства и механические системы. Диаграмма состояний помогает схематично отрисовать конечный автомат. Подробнее эту тему мы изучим на следующем уроке.

В программировании конечные автоматы используются для создания конечных автоматных машин, которые могут отслеживать состояния и изменять их в

зависимости от входных данных или событий. Их можно реализовать в виде кода на любом языке программирования, например, на C++, Java или Python. Это огромный пласт задач, которые решают разработчики.

Инструменты для сбора сведений об объекте

Мы узнали, что такое состояния и зачем они нужны. Теперь разберем, откуда берется информация о состояниях.

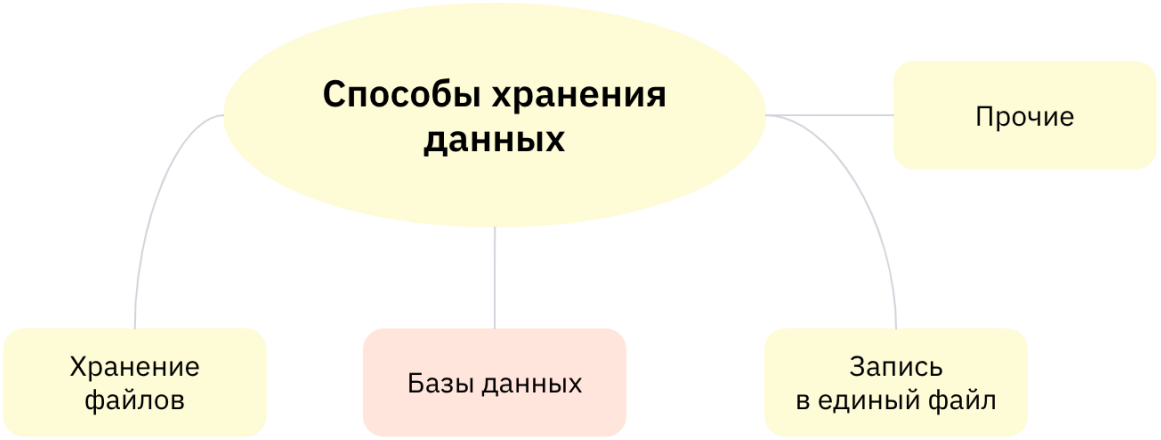
Есть несколько инструментов:

- **Логика и открытые источники.** Вместе с командой, мы можем сгенерировать множество идей о том, как должна выглядеть система и что она должна делать. После этого идеи с описанием состояний нужно верифицировать у заказчика. А лучше — пригласить представителя заказчика поучаствовать в генерации идей.
- **Интервью со стейкхолдерами**. Стейкхолдеры — это все заинтересованные в проекте лица. Например, для создания системы CRM стейкхолдерами могут быть:
    - сотрудники отдела продаж;
    - процессный офис, который оценивает эффективность процессов в компании;
    - владельцы смежных систем, с которыми взаимодействует наша CRM;
    - поставщик, чьим программным продуктом мы будем пользоваться;
    - руководство ИТ-отдела, которое выделяет бюджет на проект;
    - и так далее.

  Помните: «Стейкхолдеров всегда на одного больше, чем вы знаете, а те, которых вы знаете, имеют минимум на одну потребность больше, чем вам сейчас известно» (автор цитаты — Том Гилб).
- **Анализ бизнес-процессов**. Как правило, в компании есть регламенты или сложившиеся практики. Мы можем проанализировать их и учесть в разработке нашей системы.
- **Составление mind map**. Это метод визуализации информации. Основное преимущество в том, что визуализация концепций помогает найти

недостающие звенья и дополнить их. Подробнее о mind map — в дополнительных материалах.

Пример mind map:



Сбор информации проводится не только в начале разработки ПО, но и в любой другой момент — для обновления и дополнения состояния объекта.

Давайте рассмотрим диаграмму состояния на примере работника из отдела продаж. Представим, что нам нужно реализовать программный продукт для этого специалиста. Диаграмма состояния сотрудника отдела продаж может использоваться для отслеживания и анализа его работы на разных этапах процесса продаж. Может включать следующие состояния:

- **Привлечение клиентов:** сотрудник занимается поиском и привлечением новых клиентов для компании.
- **Предварительный контакт:** сотрудник осуществляет первичный контакт с клиентом, обсуждает его потребности и предлагает подходящие продукты или услуги.
- **Презентация:** сотрудник представляет продукты или услуги клиенту, отвечает на его вопросы.
- **Заключение сделки:** сотрудник заключает сделку с клиентом и закрывает продажу.
- **Послепродажное обслуживание:** сотрудник выполняет послепродажное обслуживание клиента.

Чтобы собрать эти данные, мы можем:

- Спросить у сотрудника из отдела продаж.
- Провести **интервью со стейкхолдером**.

- Проанализировать нормативные документы компании по работе с клиентами.
- Провести **анализ бизнес-процессов.**
- Использовать **логику и открытые источники**.

Все это можно визуализировать с помощью **mind map**.

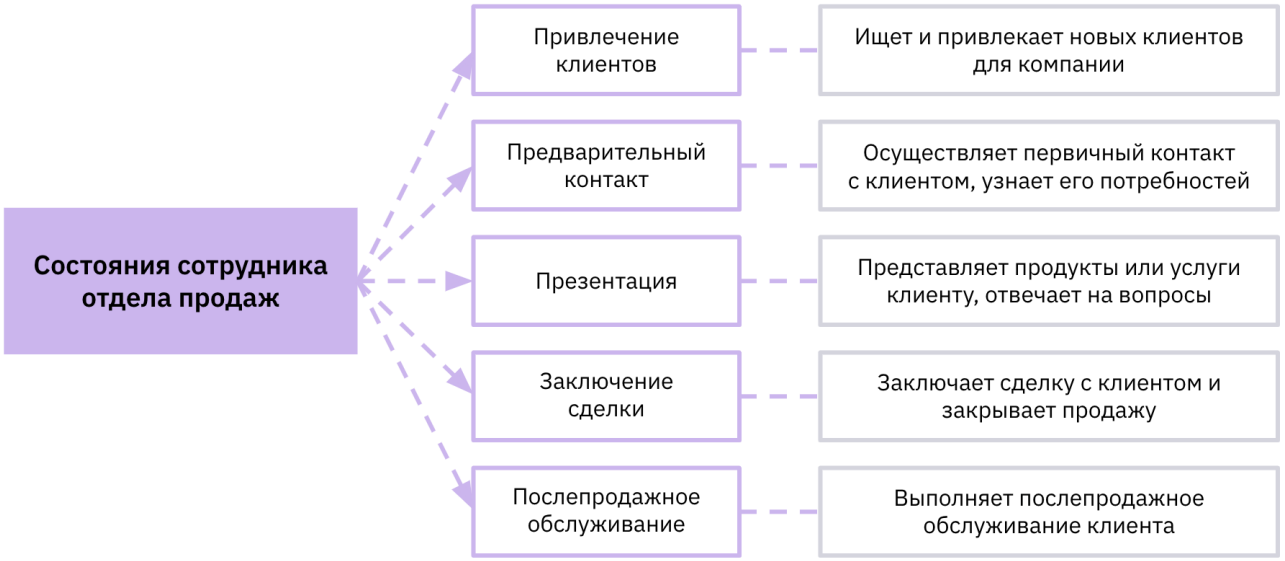


Диаграмма состояний с точки зрения разных ролей

Любую систему и процесс можно представить в виде диаграммы состояний: будь то обучение в GeekBrains, работа сотрудника продаж или программа записи к врачу.

Диаграмма состояний показывает все состояния, в которых может находиться система или процесс, а также переходы между ними. Все это позволяет лучше понять, как они работают и как изменения влияют на них.

Попробуем рассмотреть варианты использования диаграммы состояний с точки зрения разных ролей. Для начала посмотрим на кейс, с которым будем работать на текущей лекции и семинаре.

Кейс 1: программа для записи к врачу

Программное обеспечение для онлайн-записи к врачу позволяет пациентам записываться на прием через интернет без необходимости обращаться в медицинское учреждение лично.

Функционал, который может включать программа:

- просмотр доступных врачей и их расписания,
- выбор времени и даты приема,
- подтверждение записи через электронную почту или SMS,
- отмена или изменение записи,
- напоминание о записи,
- предоставление доступа с информацией о медицинской карте пациента и истории записей.

Вы можете дополнять этот кейс недостающей информацией.

В системе много функционала и объектов, чьи состояния мы можем описать. Для начала возьмем роль **пользователя программы** (клиента клиники) и опишем для него диаграмму состояния с точки зрения продуктового менеджера — специалиста, ответственного за разработку и управление продуктом.

Обязанности продуктового менеджера:

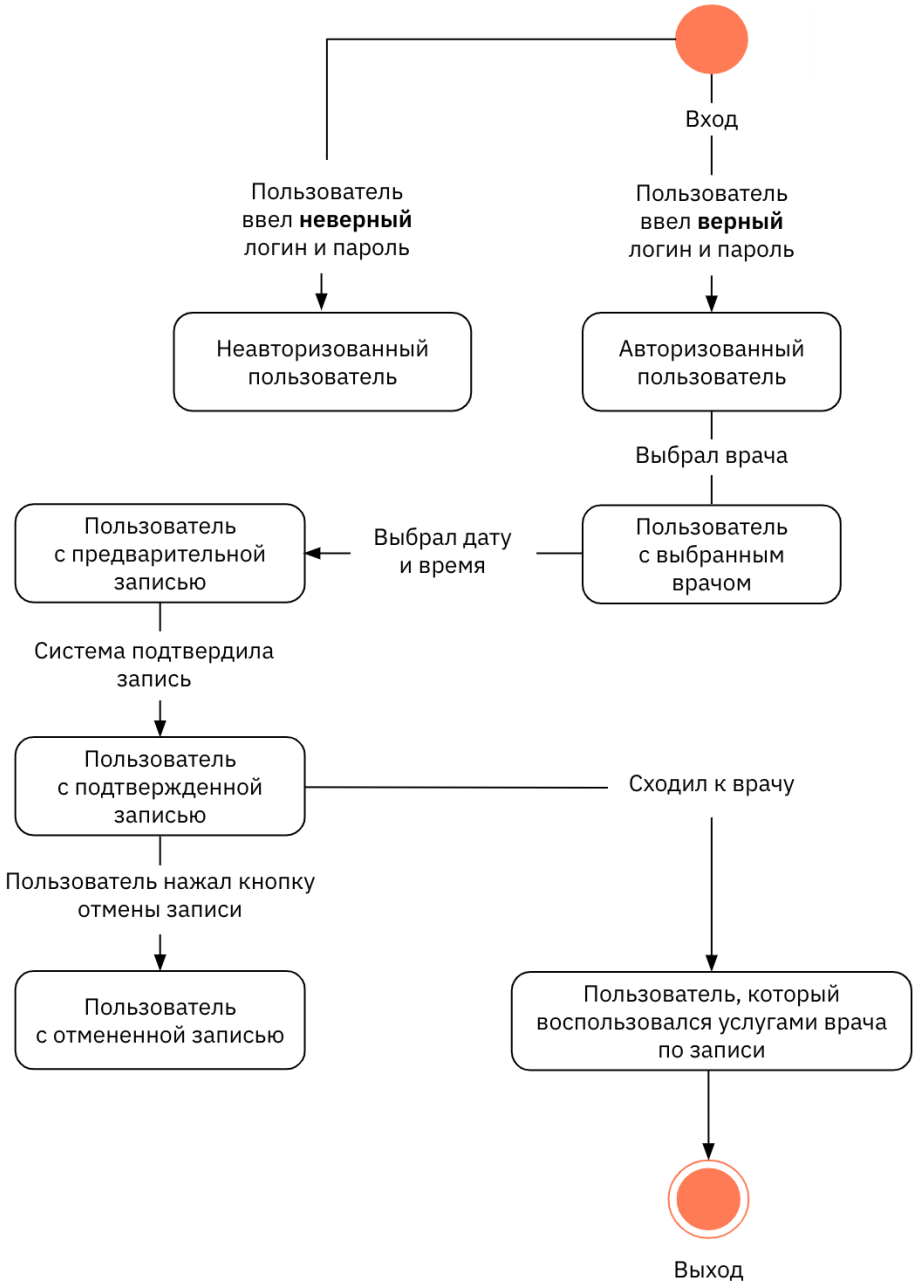
- определение потребностей клиентов и рынка для разработки продукта,
- создание стратегии и плана развития продукта,
- коммуникация с командой разработки и многое другое.

Наши клиенты — основные пользователи системы, поэтому продакт-менеджер должен точно представлять, в каком состоянии они могут находиться. Это поможет понять, как нужно улучшать продукт и какой профит это улучшение принесет.

1. Для начала опишем все состояния пользователя в системе:
    - **Не авторизован:** пользователь не авторизован в приложении и не имеет доступа к функции записи к врачу.
    - **Авторизован:** пользователь авторизован в приложении и имеет доступ к функции записи к врачу.
    - **Выбор врача:** пользователь выбирает врача из списка доступных врачей.
    - **Выбор даты и времени:** пользователь выбирает дату и время для записи к врачу.
    - **Подтверждение записи:** пользователь подтверждает запись к врачу и получает подтверждение записи.

- **Отмена записи:** пользователь отменяет запись.
- **Поход к врачу:** пользователь сходил к врачу.

2. Сформулируем действия в системе, которые переводят пользователя из одного состояния в другое:
    - **Не авторизован:** пользователь не зарегистрировался в приложении.
    - **Авторизован:** пользователь ввел логин и пароль в систему.
    - **Выбор врача:** пользователь нажал на профиль врача и выбрал врача.
    - **Выбор даты и времени:** пользователь нажал кнопку выбора даты и времени.
    - **Подтверждение записи:** врач зашел в систему и нажал кнопку «Подтвердить запись».
    - **Отмена записи:** пользователь нажал кнопку «Отменить запись».
    - **Поход к врачу:** врач оставил в системе информацию о приеме.
3. Нарисуем диаграмму состояния:


Вход
Пользователь ввел **неверный** логин и пароль
Пользователь ввел **верный** логин и пароль
Неавторизованный пользователь
Авторизованный пользователь
Выбрал врача
Пользователь с предварительной записью
Выбрал дату и время
Пользователь с выбранным врачом
Система подтвердила запись
Пользователь с подтвержденной записью
Сходил к врачу
Пользователь нажал кнопку отмены записи
Пользователь с отмененной записью
Пользователь, который воспользовался услугами врача по записи
Выход

4. Определим задачи с точки зрения продакт-менеджера: что мы хотим добавить на том или ином этапе.
    - **Не авторизован:** онбординг и регистрация для пользователей, которые еще не авторизованы.
    - **Авторизован:** улучшение системы входа для уже зарегистрированных пользователей (вход по телефону).
    - **Выбор врача:** предоставление пользователям возможности выбирать и просматривать профили разных врачей.
    - **Выбор даты и времени:** предоставление пользователям возможности выбирать дату и время для записи.
    - **Подтверждение записи:** внедрение функции подтверждения, чтобы врачи могли одобрить или отклонить запрос на запись.
    - **Отмена записи:** разрешение пользователям отменять встречи, если это необходимо.
    - **Поход к врачу:** отслеживание и информация о визитах к врачу.

С помощью диаграммы состояния мы довольно точно передали основной процесс в системе для нашей роли.

Остальные роли и их работу рассмотрим на семинаре.

Что можно почитать еще?

1. Mindmap: что это и как составить интеллект-карту | Blog Roistat
2. Интеллект-карты: как правильно составить наглядный план для любой задачи | РБК Тренды
3. Mind Mapping, или как заставить свой мозг работать лучше
4. Project managing the SDLC

Используемая литература

1. Википедия: Диаграмма состояний (UML)
2. UML-гайд с различными диаграммами