

UML для разработки

Кошкин В.А.

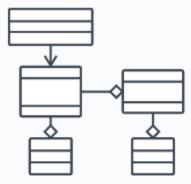
Что такое UML?

Унифицированный язык моделирования (UML) был разработан с целью обеспечить единый визуальный язык с богатой семантикой и развернутым синтаксисом для проектирования и внедрения программных систем со сложной структурой и комплексным поведением. Стоит отметить, что UML применяется не только в разработке программ, но и в других сферах, например, в схематизации потоков производственных процессов.



UML для генерации

UML не является языком программирования, однако на базе диаграмм UML можно сгенерировать код на разных языках, и для этого существует ряд специальных инструментов. Зато с объектно-ориентированным анализом и дизайном унифицированный язык моделирования связан напрямую.



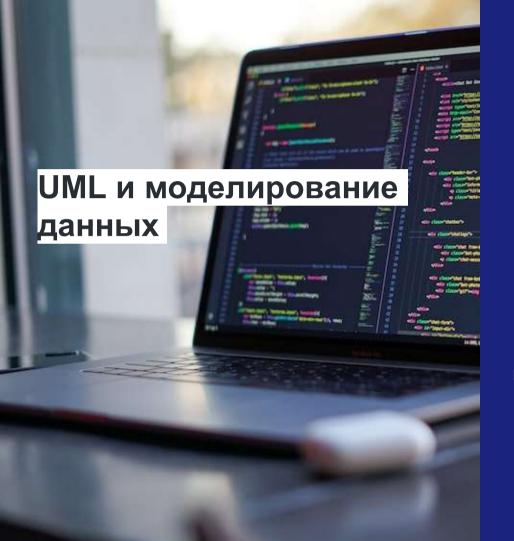
UML и его роль в объектноориентированном анализе и дизайне

Объектно-ориентированные языки преобладают в сфере программирования, так как моделируют объекты реального мира. UML сочетает в себе несколько разновидностей объектно-ориентированной нотации — объектно-ориентированный дизайн, технику объектного моделирования и разработку объектно-ориентированного программного обеспечения.

```
Project ~

    → * 1 - # index.ts × $1 env.local × ★ app.tsx ×

           - - /Projects/blog main / 9 4 1
                                          import { useEffect } from 'react'; 8.23 k8 (gzip: 3.33 k8)
                                             import Head from 'next/head';
   components
                                             import type { AppProps } from 'next/app'; 5.11 k8 (gzip: 2.16
    context
                                              import { ApolloProvider } from '@apollo/client'; 123.67 MB
      documentation
                                              import { ThemeProvider } from '@material-ui/core/styles'; ?
     > graphol
                                               import CssBaseline from '@material-ui/core/CssBaseline'; 61
     > In hooks
                                                import { Container } from '@material-ui/core'; 63.32 kB (g
                                                import { useApollo } from '../graphql/client';
      > mode modules library root
      v pages
                                                 import { lightTheme, darkTheme } from '../utils/theme';
         > admin
                                                 import useLocalStorage from '../hooks/useLocalStorage';
                                                  import MayBar from ' .. /components/NayBar';
                document tsx
                                                   function App({ Component, pageProps }: AppProps) {
              index.15x
                                                      const [currentTheme, setCurrentTheme] = useLocalStor
                                                      const apolloClient = useApollo(pageProps.initialApo
                                             18
               theme.ts
                                                       useEffect( effect () => {
                                                          const jssStyles = document.querySelector( selector
              Stabelet
             25 mov example
                                                          if (jssStyles) {
                                                             jssStyles.parentElement.removeChild(jssStyle
              any local
                                                         }, deps: (1);
                 next-eny dis
                Dackage Non
                                                           return (
                E SEADME md
                                                                   <title>ECU-DEV</title>
                                                                   <meta name="viewport" content="minimu
               El Scratches and Consoles
                                                                  «/Head>
                                                                  <ThemeProvider theme={currentTheme ===</pre>
                                                                     <ApolloProvider client={apolloClient</pre>
                                                                       <CssBaseline />
                                                                           Maritan material whom from
```



Язык UML получил широкое распространение в среде программистов, однако в целом мало используется в разработке баз данных. Одна из причин этого заключается в том, что создатели UML не ставили базы данных в центр внимания. Тем не менее, UML эффективно применяется в концептуальном моделировании данных высоких уровней и подходит для создания различных видов диаграмм UML.

Типы UML диаграмм

В языке UML есть 12 типов диаграмм:

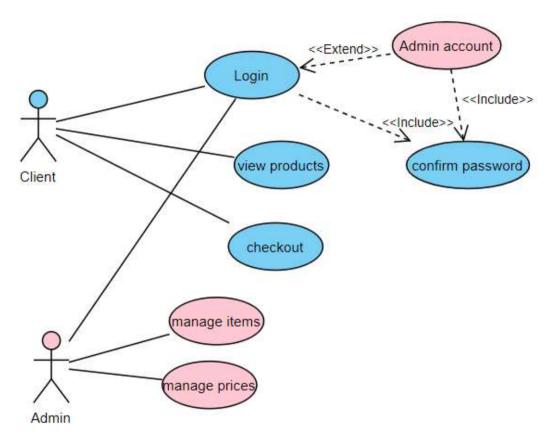
- 4 типа диаграмм представляют **статическую структуру** приложения;
- 5 типов представляют **поведенческие аспекты** системы;
- 3 представляют физические аспекты функционирования системы (диаграммы реализации).

Типы UML диаграмм

Некоторые из видов диаграмм специфичны для определенной системы и приложения. Самыми доступными из них являются:

- Диаграмма прецедентов (Use-case diagram);
- Диаграмма классов (Class diagram);
- Диаграмма активностей (Activity diagram);
- Диаграмма последовательности (Sequence diagram);
- Диаграмма развёртывания (Deployment diagram);
- Диаграмма сотрудничества (Collaboration diagram);
- Диаграмма объектов (Object diagram);
- Диаграмма состояний (Statechart diagram).





- 1) Actor (участник) множество логически связанных ролей, исполняемых при взаимодействии с прецедентами или сущностями. Участником может быть человек, роль человека в системе или другая система, подсистема или класс, которые представляют нечто вне сущности.
- 2) Use case (прецедент) описание отдельного аспекта поведения системы с точки зрения пользователя. Прецедент не показывает, "как" достигается некоторый результат, а только "что" именно выполняется.

ДИАГРАММА КЛАССОВ(CLASS DIAGRAM)

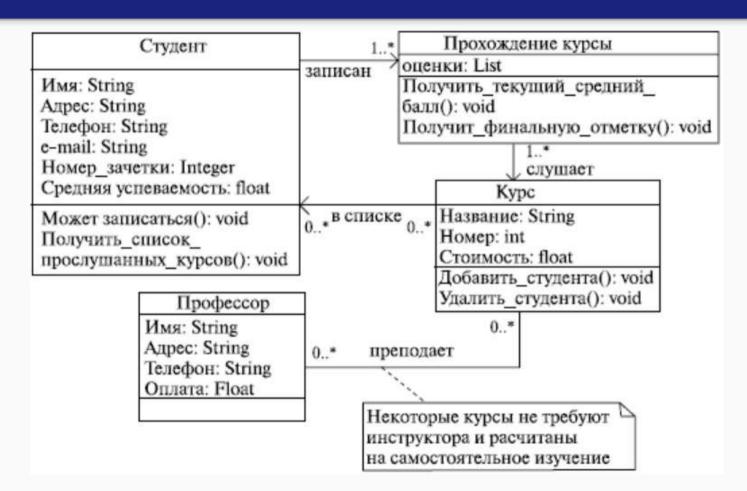


ДИАГРАММА AKTUBHOCTEЙ (ACTIVITY DIAGRAM)

Диаграмма активностей описывает динамические аспекты поведения системы в виде блок-схемы, которая отражает бизнес-процессы, логику процедур и потоки работ переходы от одной деятельности к другой. По сути, мы рисуем алгоритм действий (логику поведения) системы или взаимодействия нескольких систем.

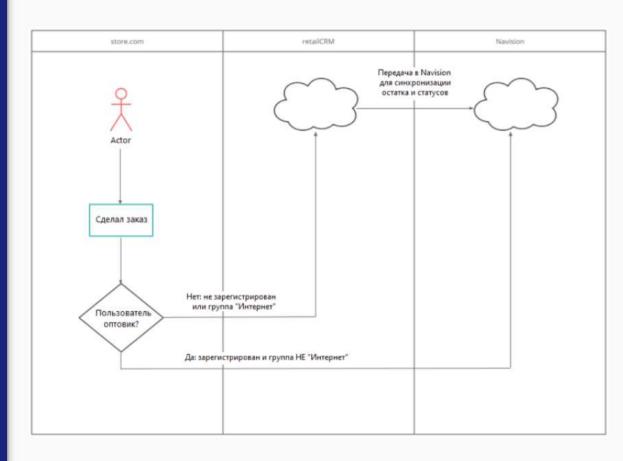


ДИАГРАММА ПОСЛЕДОВАТЕЛЬНОСТ И (SEQUENCE DIAGRAM)

Используется для уточнения диаграмм прецедентов описывает поведенческие аспекты системы. Диаграмма последовательности отражает взаимодействие объектов в динамике, во времени. При этом информация принимает вид сообщений, а взаимодействие объектов подразумевает обмен этими сообщениями в рамках сценария.

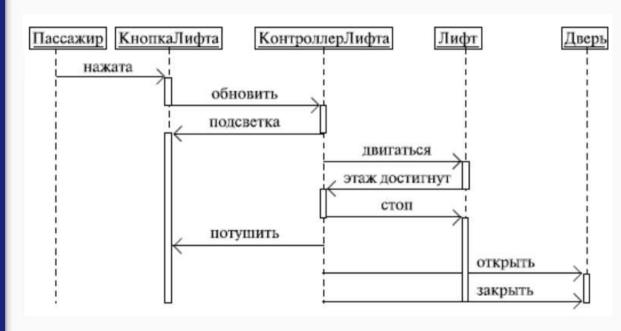


ДИАГРАММА РАЗВЁРТЫВАНИЯ(DEPLOYMENT DIAGRAM)



Диаграмма развертывания отображает графическое представление инфраструктуры, на которую будет развернуто приложение: топологию системы и распределение компонентов по ее узлам, а также соединения маршруты передачи данных между узлами. Диаграмма помогает более рационально организовать компоненты, от чего зависит в числе прочего и производительность системы, а также решить вспомогательные задачи, например, связанные с безопасностью.



UML для разработки

Кошкин В.А.