Стань QA специалистом с нуля

Раздел 6

Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

Базы данных и SQL

Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

Материалы для изучения

Учебник по SQL с возможностью выполнения команд:
http://www.sql-tutorial.ru/
Упражнения по SQL разной степени сложности
http://www.sql-ex.ru/
Справочник по SQL с возможностью выполнения команд и упражнениями:
https://www.w3schools.com/sql
Серия видеоуроков “SQL scripting for beginners” на youtube.com:
https://www.youtube.com/watch?v=cYmQr8yeALA
https://www.youtube.com/watch?v=1sMR2ApQVvw
https://www.youtube.com/watch?v=deegPjmasq8
https://www.youtube.com/watch?v=vHE-EeLaYsI

Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

Базы данных и тестирование

Умение работать с базами данных крайне полезный навык при тестировании приложения.

Возможные преимущества при работе непосредственно с базами данных:

- тестирование на большой выборке данных
- редактирование данных напрямую в базе
- обнаружение багов на уровне БД
- обнаружение недопустимых/некорректных данных

Что необходимо знать?

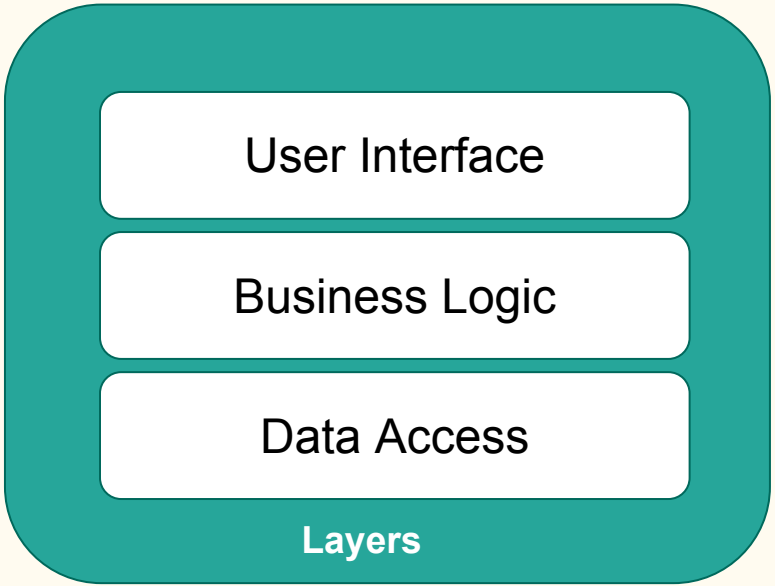
- Основы реляционных баз данных
- SQL

Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

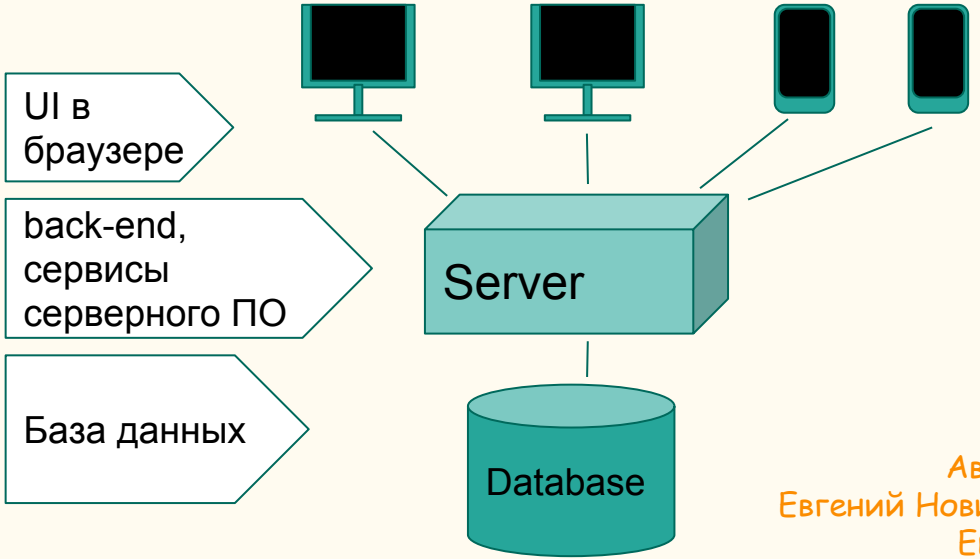
Архитектура приложения

Пример многослойной (многоуровневой) архитектуры:

- пользовательский интерфейс зависит от бизнес логики, а бизнес логика - от доступа к данным.



Клиент-серверное (client-server) приложение



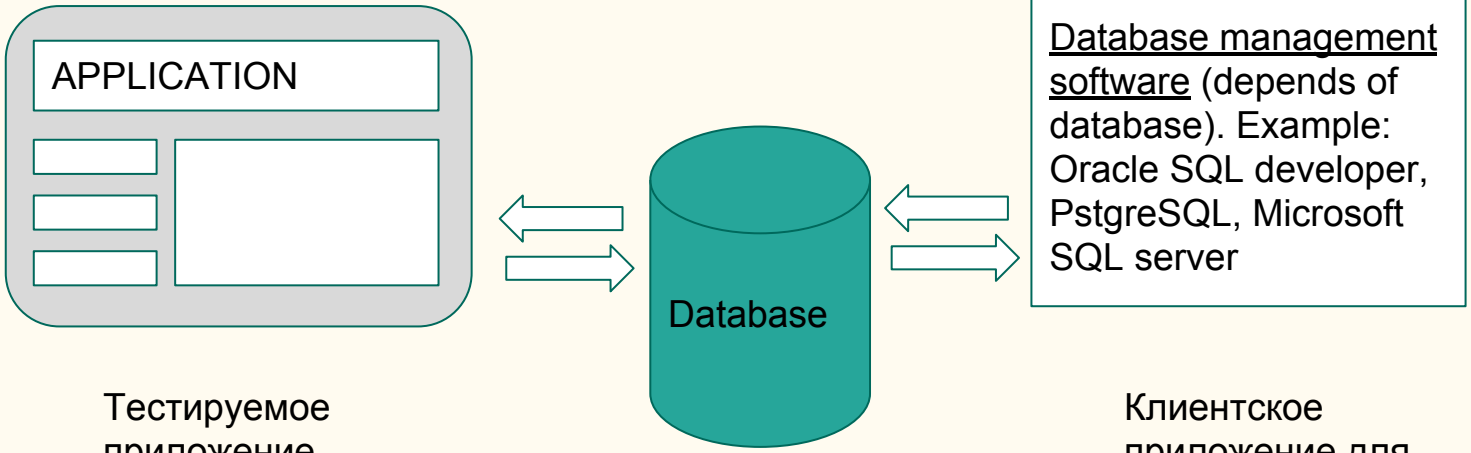
Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

Базы данных

- При разработке программного обеспечения возникает необходимость отделения логики приложения и данных, причем как back-end, так front-end
- База данных (БД) — совокупность данных, хранимых в соответствии со схемой данных.
- Система управления базами данных (СУБД) - набор интерфейсов для использования и создания данных
- Данные могут использоваться как приложением, так и пользователем (например QA инженер), при наличии доступа к базе данных
- CRUD (create, read, update, delete) - создание, чтение, запись и удаление, это типичные действия с базой данных

Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

Базы данных



Тестируемое приложение

База данных приложения

Клиентское приложение для управления базой данных

Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

Реляционные базы данных и SQL

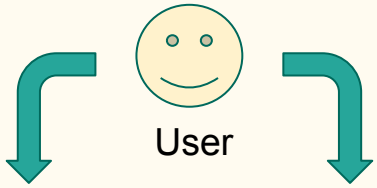
Реляционные базы данных
Это базы данных, построенные на основе реляционной модели - набора взаимосвязанных таблиц

SQL (structured query language) - язык структурированных запросов
Это язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных

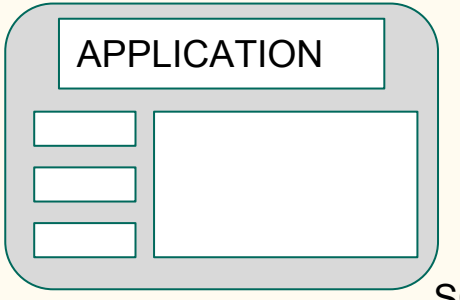
Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

SQL

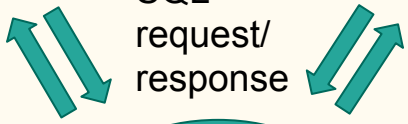
Пользователь работает с пользовательским интерфейсом программы, а программа общается с базой данных когда это необходимо



SQL (structured query language) - язык структурированных запросов
Это язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных.



Database management software - система управления базами данных - позволяет писать запросы или скрипты на языке SQL и получать ответы от базы данных



- **Запрос (Query, request)**
  показать, удалить, добавить или изменить данные
- **Ответ (Output, response)**
  вывод содержимого таблицы, response code (код):
  - код 0 (успех операции),
  - код ошибки

Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

Реляционные базы данных

Это базы данных, построенные на основе реляционной модели. Таблицы внутри базы связаны по полям через внешние ключи. Преимущество таких баз данных - информационная и структурная независимость.

employees		
employee_id	employee_name	employee_surname
1	Ivan	Petrov
2	Alex	Smith
3	Vasily	Ivanov
4	Tom	Smith

employee_phones		
phone_id	phone_number	employee_id
1	10203040	1
2	11223344	1
3	111222333	3
4	222333444	4

Ниже, для сравнения, приводится пример плохо структурированной базы:

employees				
employee_id	employee_name	employee_surname	phone_1	phone_2
1	Ivan	Petrov	10203040	11223344
2	Alex	Smith	null	null
3	Vasily	Ivanov	111222333	null

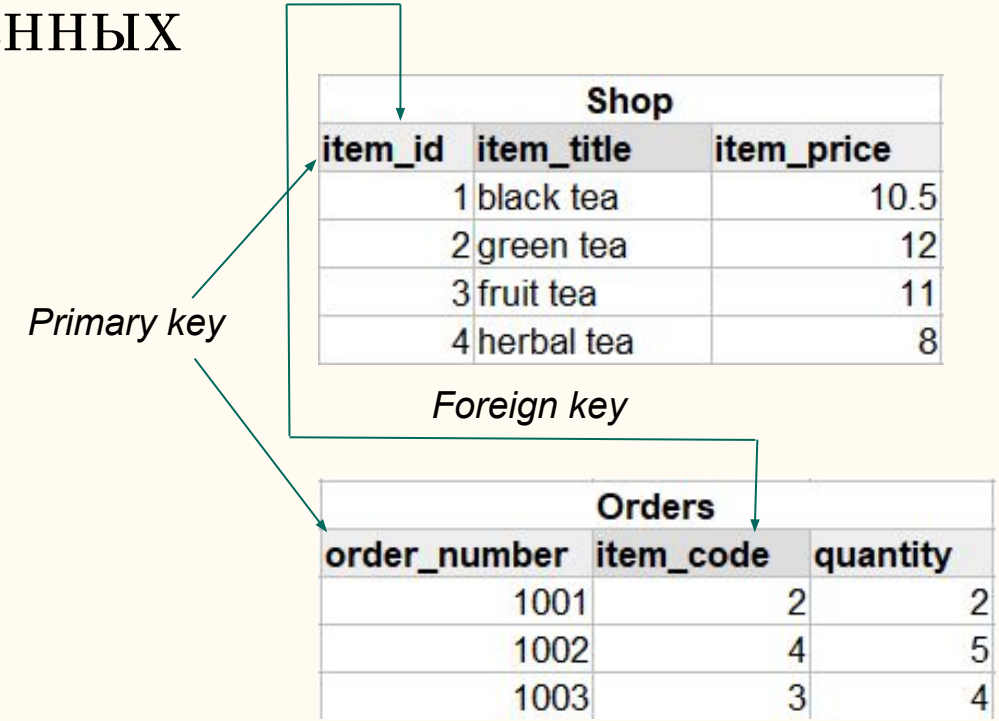
Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

Реляционные базы данных

Терминология

- **Первичный ключ (Primary key)**
  уникальный идентификатор записи (строки) в таблице. Может состоять из одного или нескольких полей (столбцов)

- **Внешний ключ (Foreign key)**
  одно или несколько полей таблицы, используемых для установления логической связи между 2 таблицами внутри базы данных.



Shop

item_id	item_title	item_price
1	black tea	10.5
2	green tea	12
3	fruit tea	11
4	herbal tea	8

Orders

order_number	item_code	quantity
1001	2	2
1002	4	5
1003	3	4

Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

Задание 1

Реляционные базы данных

Для лучшего понимания реляционных баз данных, посмотрите видеоуроки:
https://www.youtube.com/watch?v=YWniTsFYhMw
https://www.youtube.com/watch?v=P40Qi9QKTaQ

Ответьте на вопросы:

1) Что такое реляционная база данных?
2) Что такое SQL и для чего он применяется?
3) Чем отличаются и для чего предназначены первичный ключ и внешний ключ?

Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

SQL DML

SQL DML (Data Manipulation Language)

Виды запросов:

- SELECT - вывод данных из одной или нескольких таблиц
- INSERT - вставка строки/строк в таблицу
- UPDATE - изменение одной или нескольких (уже существующих) строк
- DELETE - удаление строки/строк из таблицы

SQL DML запрос всегда начинается с ключевого слова select, update, insert или delete

Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

Select: простейший запрос

Выводим содержимое всех полей таблицы employees:
SELECT * **FROM** employees;

employee_id	employee_name	employee_surname
1	Ivan	Petrov
2	Alex	Smith
3	Vasily	Ivanov
4	Tom	Smith

Выводим содержимое выбранных (через запятую) полей:
SELECT employee_name, employee_surname **FROM** employees;

employee_name	employee_surname
Ivan	Petrov
Alex	Smith
Vasily	Ivanov
Tom	Smith

Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

Select all VS distinct

Выводим полное содержимое employee_surname
SELECT employee_surname **FROM** employees;

employee_surname
Petrov
Smith
Ivanov
Smith

Выводим содержимое employee_surname без повторов, благодаря distinct
SELECT DISTINCT employee_surname **FROM** employees;

employee_surname
Petrov
Smith
Ivanov

Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

Select - WHERE

Если необходимо написать запрос с условием, используем WHERE:

SELECT * **FROM** employees **WHERE** employee_id >= 3;

employee_id	employee_name	employee_surname
3	Vasily	Ivanov
4	Tom	Smith

SELECT employee_surname **FROM** employees **WHERE** employee_name = 'Ivan';

employee_surname
Petrov

Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

Select - WHERE

Условие с отрицанием будет содержать оператор NOT:

SELECT * **FROM** employees **WHERE NOT** employee_surname = ‘Smith’;

employee_id	employee_name	employee_surname
1	Ivan	Petrov
3	Vasily	Ivanov

Условие с перечислением можно написать при помощи IN:

SELECT employee_surname **FROM** employees **WHERE** employee_id **IN** (1, 4);

employee_surname
Petrov
Smith

Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

Select - WHERE

WHERE позволяет комбинировать условия при помощи операторов AND и OR

SELECT employee_surname **FROM** employees **WHERE** employee_id <=3
AND NOT employee_name = 'Alex';

employee_surname
Petrov
Ivanov

SELECT * **FROM** employees **WHERE** employee_name = 'Alex'
OR employee_surname = 'Ivanov';

employee_id	employee_name	employee_surname
2	Alex	Smith
3	Vasily	Ivanov

Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

Select - LIKE

Like позволяет искать данные по маске.

_ - любой одиночный символ
% - ноль, один или группа символов

Пример того, как можно вывести данные для сотрудников, фамилии которых заканчиваются на “ov”:

SELECT * FROM employees WHERE employee_surname LIKE ‘%ov’;

employee_id	employee_name	employee_surname
1	Ivan	Petrov
3	Vasily	Ivanov

Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

Select - ORDER BY

Восходящая сортировка ASCENDING:
SELECT employee_surname **FROM** employees **WHERE** employee_id < 4
ORDER BY employee_surname;

employee_surname
Ivanov
Petrov
Smith

Нисходящая сортировка DESCENDING:
SELECT * **FROM** employees **ORDER BY** employee_id **DESC**;

employee_id	employee_name	employee_surname
4	Tom	Smith
3	Vasily	Ivanov
2	Alex	Smith
1	Ivan	Petrov

Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

Select - JOIN

Для чего необходимо соединение?
Чтобы вывести данные сразу из двух или более таблиц.
Как соединить таблицы?
На основе внешнего ключа (foreign key).
В данном примере это поле employee_id
Пример соединения:
SELECT * FROM employees **JOIN** employee_phone **ON** employees.employee_id = employee_phone.employee_id;

Таблица слева от **JOIN** - левая таблица
Таблица справа от **JOIN** - правая таблица

employee_phones

phone_id	phone_number	employee_id
1	10203040	1
2	11223344	1
3	111222333	3
4	222333444	4

employees

employee_id	employee_name	employee_surname
1	Ivan	Petrov
2	Alex	Smith
3	Vasily	Ivanov
4	Tom	Smith

Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

Select - JOIN

Виды соединений

FULL JOIN

соединяет данные из левой и правой таблиц, даже если нет соответствия по внешнему ключу

RIGHT JOIN

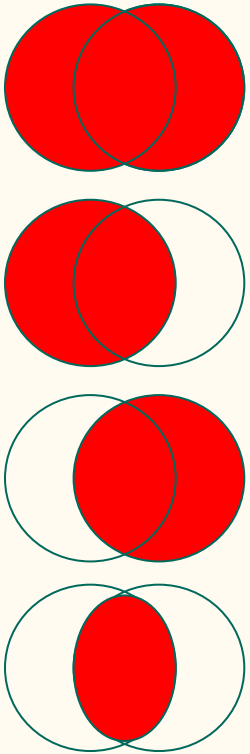
из правой таблицы в результат попадают только те строки где есть соответствие ключей в обеих таблицах, из левой - все данные

LEFT JOIN

из левой таблицы в результат попадают только те строки где есть соответствие ключей в обеих таблицах, из правой - все данные

INNER JOIN

соединяет данные из таблиц только при полном соответствии ключей



Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

Select - JOIN

SELECT * FROM employees **LEFT JOIN** employee_phones **ON** employees.employee_id = employee_phones.employee_id;

employee_id	employee_name	employee_surname	phone_id	phone_number
1	Ivan	Petrov	1	10203040
1	Ivan	Petrov	2	11223344
2	Alex	Smith	null	null
3	Vasily	Ivanov	3	111222333
4	Tom	Smith	4	222333444

SELECT * FROM employees **INNER JOIN** employee_phones **ON** employees.employee_id = employee_phones.employee_id **ORDER BY** phone_id;

employee_id	employee_name	employee_surname	phone_id	phone_number
1	Ivan	Petrov	1	10203040
1	Ivan	Petrov	2	11223344
3	Vasily	Ivanov	3	111222333
4	Tom	Smith	4	222333444

Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

Select - агрегатные функции

Это математические функции, используемые в SQL запросе:
COUNT - считает количество записей по выбранному полю или всей таблице.
MIN - выводит минимальное значение из таблицы или сгруппированных данных.
MAX - выводит максимальное значение из таблицы или сгруппированных данных.
SUM - выводит сумму значений выбранного поля таблицы.

SELECT COUNT(employee_id) **FROM** employees;

COUNT(employee_id)
4

SELECT MAX(employee_id) **FROM** employees;

COUNT(employee_id)
4

Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

Select - GROUP BY & HAVING

Группировка (GROUP BY) с опциональным условием группировки (HAVING), обычно применяются вместе с агрегатными функциями, то есть когда необходимо сделать расчеты в сгруппированных данных.
Например необходимо найти сотрудников, у которых два и более телефонных номеров:

employee_phones		
phone_id	phone_number	employee_id
1	10203040	1
2	11223344	1
3	111222333	3
4	222333444	4

SELECT COUNT(phone_id), employee_id **FROM** employee_phones **GROUP BY** employee_id **HAVING** COUNT(phone_id) >= 2;

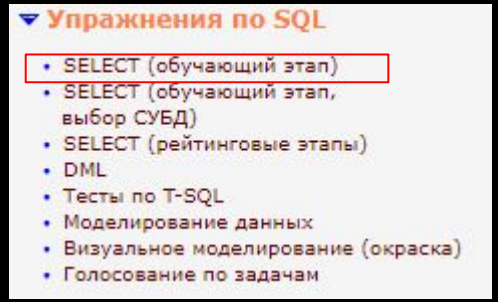
COUNT(phone_id)	employee_id
2	1

Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

Задание 2

SELECT запросы

1) Зарегистрируйтесь на http://www.sql-ex.ru/
2) Откройте секцию SELECT(обучающий этап)



3) Выполните все упражнения 1 уровня сложности
4) Выполните максимальное число заданий 2 уровня сложности

Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

Insert

Insert используется для добавления новой записи в таблицу:
INSERT INTO employees (employee_id, employee_name, employee_surname)
VALUES (5, ‘Jack’, ‘Stone’);

При успешном выполнении insert запрос возвращает 0. Чтобы проверить как добавились данные, выполняем select запрос:
SELECT * FROM Employees;

employee_id	employee_name	employee_surname
1	Ivan	Petrov
2	Alex	Smith
3	Vasily	Ivanov
4	Tom	Smith
5	Jack	Stone

Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

Update

Update применяется для модификации данных в талице:
UPDATE employees **SET** employee_name = ‘John’ **WHERE** employee_id = 5;

При успешном выполнении update запрос возвращает 0. Чтобы проверить как добавились данные, выполняем select запрос:
SELECT * **FROM** employees;

employee_id	employee_name	employee_surname
1	Ivan	Petrov
2	Alex	Smith
3	Vasily	Ivanov
4	Tom	Smith
5	John	Stone

Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

Delete

Delete используется для удаления записей из таблицы:
DELETE FROM employees **WHERE** employee_id = 5;

При успешном выполнении delete запрос возвращает 0. Чтобы проверить как добавились данные, выполняем select запрос:
SELECT * **FROM** employees;

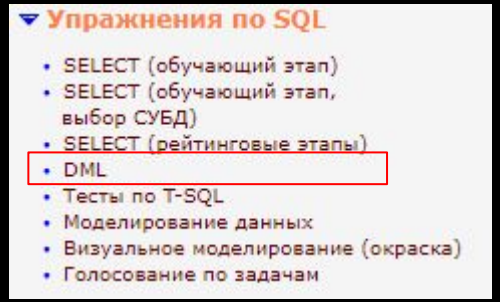
employee_id	employee_name	employee_surname
1	Ivan	Petrov
2	Alex	Smith
3	Vasily	Ivanov
4	Tom	Smith

Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

Задание 3

INSERT, UPDATE, DELETE

1) Зарегистрируйтесь на http://www.sql-ex.ru/
2) Откройте секцию DML



3) Выполните по несколько упражнений на INSERT, UPDATE и DELETE

Автор:
Евгений Новиков
Email:
xevgnov@gmail.com

Спасибо!

Курс:
Стань QA специалистом с нуля
Лекция 6

Автор:
Евгений Новиков

Email:
xevgnov@gmail.com