**правила проверки корректности email**

Правила проверки корректности email-адреса основаны на стандартах и общепринятых практиках. Основным стандартом, определяющим синтаксис email-адресов, является **RFC 5322**. Ниже приведены основные правила проверки:

1. **Наличие символа "@":** Email-адрес должен содержать один и только один символ "@", разделяющий локальную и доменную части.
2. **Локальная часть:**
   * Может содержать буквы (a–z, A–Z), цифры (0–9) и специальные символы (! # $ % & ' \* + - / = ? ^ \_ { | } ~`).
   * Точка (.) допускается, но не может быть первым или последним символом и не может идти подряд.
   * Длина локальной части не должна превышать 64 символа.
3. **Доменная часть:**
   * Должна соответствовать правилам доменных имен, определенным в **RFC 1035** и **RFC 1123**.
   * Может содержать буквы, цифры и дефисы (-), но дефис не может быть первым или последним символом.
   * Каждая метка домена не должна превышать 63 символа, а общая длина домена — 255 символов.
4. **Общая длина:** Общая длина email-адреса не должна превышать 254 символов.
5. **Недопустимые символы:** Пробелы, запятые, двоеточия, точка с запятой и другие специальные символы недопустимы вне кавычек.
6. **Использование кавычек:** Локальная часть может быть заключена в кавычки, что позволяет использовать специальные или недопустимые символы, но это редко используется и может не поддерживаться всеми почтовыми серверами.
7. **Валидация домена:** Рекомендуется проверить наличие MX-записи в DNS для домена, чтобы убедиться, что домен может получать почту.

**Подтверждение фактами:**

* **RFC 5322**: Определяет формат сообщений электронной почты, включая синтаксис адресов.
* **RFC 1035**: Описывает реализацию доменных имен в системе DNS.
* **RFC 1123**: Дополняет и уточняет спецификации протоколов интернет.

**Пример корректного email-адреса:**

|  |
| --- |
| [example.user+label@sub-domain.example.com](mailto:example.user+label@sub-domain.example.com) |

**Примечание:** Несмотря на строгие стандарты, на практике некоторые почтовые сервисы могут иметь собственные ограничения или допуски. Поэтому при проверке корректности email-адреса важно учитывать как официальные стандарты, так и реалии их применения.

**регулярное выражение для проверки корректности email**

Составление регулярного выражения для проверки корректности email-адреса может быть сложной задачей из-за разнообразия допустимых форматов согласно стандарту **RFC 5322**. Однако можно создать выражение, которое охватывает наиболее распространенные и практичные случаи.

Вот пример регулярного выражения для проверки email-адреса:

|  |
| --- |
| ^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$ |

**Объяснение этого выражения:**

* ^ и $ — начало и конец строки, обеспечивают проверку всей строки целиком.
* [A-Za-z0-9.\_%+-]+ — локальная часть:
  + Состоит из одного или более символов: букв (A–Z, a–z), цифр (0–9), точки (.), подчеркивания (\_), процента (%), плюса (+), дефиса (-).
* @ — разделитель между локальной и доменной частями.
* [A-Za-z0-9.-]+ — доменная часть:
  + Состоит из одного или более символов: букв, цифр, точки и дефиса.
* \. — обязательная точка перед доменным суффиксом.
* [A-Za-z]{2,} — доменный суффикс:
  + Состоит минимум из двух букв (например, com, ru, info).

**Важно отметить:**

* **Ограничения данного выражения:**
  + Не учитывает все возможные варианты, допустимые по стандарту RFC 5322.
  + Не позволяет использовать кавычки в локальной части.
  + Не учитывает IP-адреса в качестве доменной части (например, user@[192.168.0.1]).
  + Не проверяет длину локальной и доменной частей согласно ограничениям стандарта.
* **Дополнительные рекомендации:**
  + Для более точной проверки можно использовать библиотеки, специально разработанные для валидации email-адресов, которые учитывают все нюансы стандарта.
  + Помимо синтаксической проверки, рекомендуется выполнять проверку существования домена и MX-записей через DNS.

**Заключение:**

Предоставленное регулярное выражение подходит для большинства практических случаев и обеспечивает базовую проверку корректности email-адреса. Для критически важных систем, где требуется полное соответствие стандартам, следует применять специализированные решения.

**На C# существует несколько библиотек и подходов для проверки корректности email-адресов, а также для проверки существования домена и MX-записей через DNS. Вот несколько популярных решений:**

### 1. **MailKit**

**MailKit** — это мощная библиотека с открытым исходным кодом для работы с протоколами электронной почты, включая IMAP, POP3 и SMTP. Она также поддерживает валидацию email-адресов и проверки DNS.

**Пример использования для проверки существования домена и MX-записей:**

|  |
| --- |
| using MailKit.Net.Smtp; using DnsClient;  var lookup = new LookupClient(); var result = lookup.Query("example.com", QueryType.MX); if (result.Answers.MxRecords().Any()) {  Console.WriteLine("Домен поддерживает получение почты."); } else {  Console.WriteLine("MX-записи не найдены."); } |

**MailKit** позволяет также отправлять тестовые SMTP-запросы для проверки доставки почты.

2. **System.Net.Mail**

Класс MailAddress из пространства имен System.Net.Mail может использоваться для базовой проверки синтаксиса email-адреса:

|  |
| --- |
| using System.Net.Mail;  try {  var address = new MailAddress("example@domain.com");  Console.WriteLine("Корректный email-адрес."); } catch (FormatException) {  Console.WriteLine("Некорректный email-адрес."); } |

Однако этот метод проверяет только синтаксис и не гарантирует, что домен существует.

### 3. **DnsClient.NET**

[DnsClient.NET](https://github.com/MichaCo/DnsClient.NET) — это популярная библиотека для выполнения DNS-запросов. С ее помощью можно проверять наличие MX-записей у домена:

|  |
| --- |
| using DnsClient;  var client = new LookupClient(); var queryResult = client.Query("example.com", QueryType.MX);  if (queryResult.Answers.MxRecords().Any()) {  Console.WriteLine("MX-записи найдены, домен поддерживает почту."); } else {  Console.WriteLine("MX-записи не найдены."); } |

4. **ARSoft.Tools.Net**

[ARSoft.Tools.Net](https://github.com/alexreinert/ARSoft.Tools.Net) — библиотека для выполнения DNS-запросов, поддерживающая работу с различными типами DNS-записей, включая MX-записи.

Пример использования для проверки существования MX-записи:

|  |
| --- |
| using ARSoft.Tools.Net.Dns;  var response = DnsClient.Default.Resolve("example.com", RecordType.Mx); if (response?.AnswerRecords?.Any(record => record.RecordType == RecordType.Mx) == true) {  Console.WriteLine("MX-записи найдены, домен поддерживает почту."); } else {  Console.WriteLine("MX-записи не найдены."); } |

Вывод

* **MailKit**, **DnsClient.NET**, и **ARSoft.Tools.Net** могут использоваться для выполнения проверки существования домена и его MX-записей через DNS.
* MailAddress из System.Net.Mail подходит для базовой проверки синтаксиса email-адреса.

Эти библиотеки позволяют проводить как синтаксическую проверку, так и проверку существования домена и его способности принимать электронные письма, что особенно важно для обеспечения надежности работы с email-адресами.

**FluentValidation — это популярная библиотека для создания правил валидации на языке C#. Она позволяет писать декларативные правила проверки для объектов, делая код более чистым и поддерживаемым. Хотя FluentValidation в первую очередь предназначена для проверки объектов и их свойств, ее также можно использовать для проверки email-адресов.**

Для проверки email-адресов с помощью **FluentValidation** можно воспользоваться встроенными методами, такими как EmailAddress(), которые обеспечивают базовую проверку корректности синтаксиса email.

Пример использования FluentValidation для проверки email-адреса:

|  |
| --- |
| using FluentValidation; using FluentValidation;  public class User {  public string Email { get; set; } }  public class UserValidator : AbstractValidator<User> {  public UserValidator()  {  RuleFor(user => user.Email)  .NotEmpty().WithMessage("Email не должен быть пустым.")  .EmailAddress().WithMessage("Некорректный формат email-адреса.");  } } |

**Объяснение:**

* NotEmpty() — Проверяет, что значение не является пустым.
* EmailAddress() — Выполняет проверку формата email-адреса, соответствующего стандартам (например, наличие символа @ и корректная доменная часть).

**Ограничения:**

* FluentValidation выполняет только синтаксическую проверку email-адресов. Она не проверяет, существует ли домен или MX-записи для домена. Для выполнения таких проверок необходимо использовать дополнительные библиотеки, такие как **DnsClient.NET** или **MailKit**.
* Метод EmailAddress() использует базовые правила, схожие с регулярным выражением, что подходит для большинства практических случаев, но может не охватить все особенности стандарта **RFC 5322**.

**Когда использовать FluentValidation:**

Когда нужно создать проверку email-адреса в контексте проверки модели, например, при проверке данных, поступающих от пользователя через форму.

* Когда важно следовать декларативному стилю написания правил проверки