

Brazilian CNPJ

Brazilian company tax ID — 14 digits.

```
^\d{2}\.\d{3}\.\d{3}\\/\d{4}-\d{2}$
```

Token-by-token breakdown

Token	Meaning
^	start of string (or line in multiline mode)
\d{2}	exactly 2 times: digit (0–9)
\.	literal "."
\d{3}	exactly 3 times: digit (0–9)
\.	literal "."
\d{3}	exactly 3 times: digit (0–9)
\/	literal "/"
\d{4}	exactly 4 times: digit (0–9)
-	literal text "-"
\d{2}	exactly 2 times: digit (0–9)
\$	end of string (or line in multiline mode)

Quick usage in 7 languages

JavaScript	<pre>const re = /^\d{2}\.\d{3}\.\d{3}\\/\d{4}-\d{2}\$/; re.test(value);</pre>
Python	<pre>import re re.match(r"^\d{2}\.\d{3}\.\d{3}\\/\d{4}-\d{2}\$", value)</pre>
Java	<pre>Pattern.compile("^\d{2}\\.\d{3}\\.\d{3}\\/\d{4}-\d{2}\$").matcher(value).matches();</pre>
C# / .NET	<pre>Regex.IsMatch(value, @"^\d{2}\.\d{3}\.\d{3}\\/\d{4}-\d{2}\$");</pre>
Go	<pre>regexp.MustCompile(`^\d{2}\.\d{3}\.\d{3}\\/\d{4}-\d{2}\$`).MatchString(value)</pre>
Ruby	<pre>/^\d{2}\.\d{3}\.\d{3}\\/\d{4}-\d{2}\$/ .match?(value)</pre>
PHP	<pre>preg_match('~^\d{2}\.\d{3}\.\d{3}\\/\d{4}-\d{2}\$~', \$value);</pre>

Common pitfalls

- **Anchored to the whole string.** This pattern uses ^ and \$, so it requires the entire input to match. To find it inside a longer text, drop the anchors and use the global (g) flag.
- **Escape it correctly per language.** In Java and JavaScript strings each backslash must be doubled (\\d); in Python, Go, and C# use raw/verbatim strings so the backslashes survive.
- **Validate beyond format.** Matching the format doesn't guarantee the value is real. Confirm the brazilian cnpj against a source of truth (database, API, or checksum) where it matters.