

The screenshot shows a VS Code editor with a project named 'cs-253p-hw-hampr-machine-service'. The Explorer sidebar on the left shows the file structure, including 'src' and 'test' directories. The main editor window displays the 'api.ts' file, which defines an 'ApiHandler' class. The terminal window at the bottom shows the output of running tests, including a table of resource usage and console logs for 'simulation.test.ts'.

Resource Usage Table:

(index)	Resource	Run 1 Units	Run 1 %	Run 2 Units	Run 2 %	Run 3 Units	Run 3 %	Run 4 Units	Run 4 %
0	'IdentityProviderClient'	3840256	'69.18%'	3840256	'69.39%'	3840256	'69.12%'	3840256	'69.24%'
1	'SmartMachineClient'	32256	'0.58%'	32256	'0.58%'	32256	'0.58%'	32256	'0.58%'
2	'MachineStatsTable'	1651824	'29.74%'	1651824	'29.83%'	1651824	'29.72%'	1651824	'29.77%'
3	'DataCache'	22956	'0.41%'	22956	'0.41%'	22956	'0.41%'	22956	'0.41%'

Cache Hits Table:

(index)	Run	Cache Hits	Cache Misses	Hit Rate
0	1	3059	2222	'63.46%'
1	2	3919	2139	'64.69%'
2	3	3834	2122	'64.37%'
3	4	3867	2150	'64.27%'

Hit/Access Ratio Table:

(index)	Run	Cache Hits	DB Accesses	Hit/Access Ratio
0	1	3059	6633	'0.5018'
1	2	3919	6633	'0.5908'
2	3	3834	6633	'0.5780'
3	4	3867	6633	'0.5830'

Test Results:

```
Test Suites: 2 passed, 2 total
Tests: 12 passed, 12 total
Snapshots: 0 total
Time: 0.798 s, estimated 1 s
Run all test suites.
```

I'm a little surprised that the Identity Provider Client usage is so high, because theoretically the only thing I coded that used that was the validity checker. I think that I could have done different code to have more usage on the data cache, particularly in the `handleGetMachine`. I'm not quite sure what ratio is good for the cache and db hits, but I think it's good that I am consistently getting over 50% for both.