1. Key Trace Metrics

Trace Name	Samples	Avg. Duration	7-day Δ
_app_start	6	5.59 s	+0 %
home_screen_load	13	3.38 s	+0 %
poi_list_load	25	2.20 s	+0 %
poi_details_load	11	3.15 s	+0 %
my_reviews_load	15	6.82 s	+0 %
_app_in_background	5	8.07 s	−92 %
_app_in_foreground	13	1.4 min	−97 %
filter_operation	15	0.74 ms	+0 %
app_cold_start	12	0.66 ms	+0 %

Longest user-facing loads:
my_reviews_load (6.82 s)
_app_start (5.59 s)

Mid-range loads:
home_screen_load (3.38 s)
poi_details_load (3.15 s)
poi_list_load (2.20 s)

2. Statistical Observations

Stability:
All key traces show 0 % drift over the past week, indicating consistent performance.

Foreground vs. Background:
Foreground time has dropped dramatically (−97 %), likely due to reduced user sessions or instrumentation change.
Background work is now minimal (8 s), so focus should remain on startup and screen-loads.

Micros:
Your in-memory filters (filter_operation) and cold-start wrapper (app_cold_start) are negligible—no action needed there.

3. High-Level Causes & Light Recommendations

Trace	Possible Cause	Quick Win
_app_start	JS bundle parsing & native-module init	Enable Hermes & inlineRequires
my_reviews_load	Many Firestore reads + image downloads	Aggregate calls / use cached thumbnails
home_screen_load	Multiple small queries + full re-render	Enable offline persistence & placeholders
poi_list_load	Unpaginated fetch of all items	Implement Firestore cursors + FlatList
poi_details_load	Full-size image fetch on navigation	Fetch low-res thumbnails first

Next Steps:

We'll track the impact of each optimization by reviewing the **Avg. Duration** (as well as 95th-percentile timings and any emerging error spikes) in the Firebase Performance dashboard after every deployment. To gain deeper insights, we'll instrument our code with custom JS-side performance marks immediately before and after critical operations—namely, each Firestore query and primary component render. By correlating those measurements, our team can distinguish network-fetch time from UI-render time, compare pre- and post-optimization traces, and systematically identify any residual performance bottlenecks for further refinement.