Work Time Recorder — システム概要

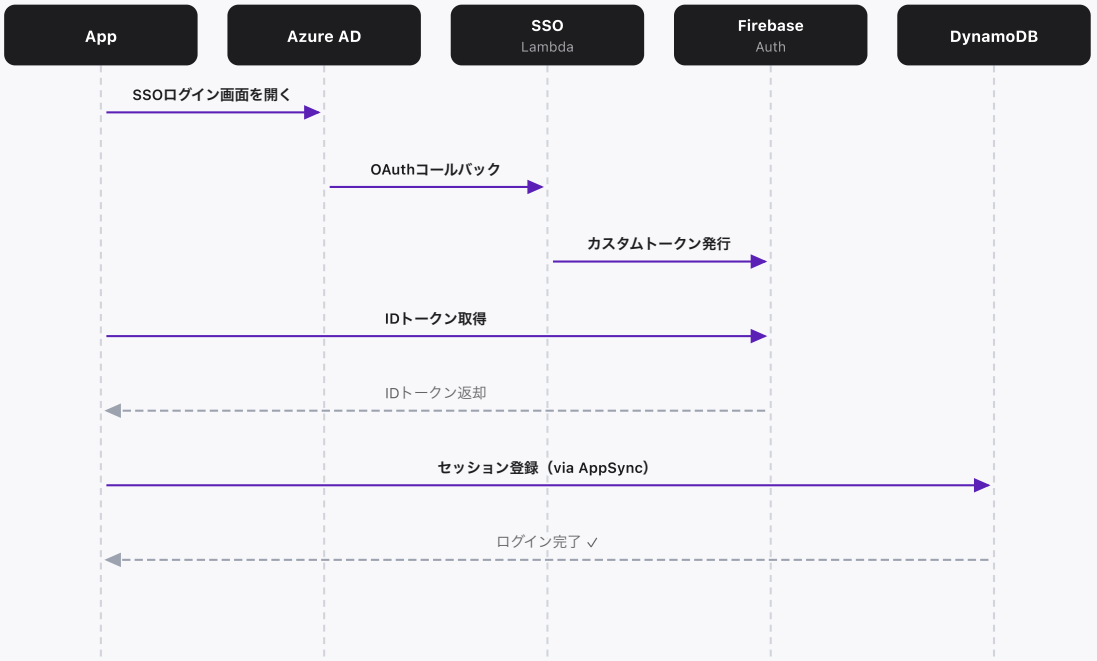
対象: IT部門・インフラ担当　作成日: 2026-04-15　機密: クライアント社内限り

1 バックエンドアーキテクチャ図

フロー図　構成図

主要処理フローの通信シーケンスを表示。タブで切り替えてください。

① ログイン　② 出退勤打刻　③ 受信トレイ　④ トークン自動更新



2 主要データフロー

ログイン　出退勤打刻　受信トレイ（タスク取得）　トークン自動更新

1 アプリ → Microsoft Azure AD（SSOログイン画面を開く）

sessionId（UUID）を生成しOAuth stateパラメータとして渡す。prompt=select_accountで共有デバイスにも対応。

2 バックグラウンドポーリング（最大180回 × 1.5秒間隔）で認証完了を待機
Microsoft → ServiceNowOAuthCallbackLambda → DynamoDBにトークンを一時保存。

3 ServiceNowSSOLoginLambda → Microsoft Graph APIでユーザー情報取得

4 Lambda → Firebase Auth（カスタムトークン発行）

5 アプリ → Firebase signInWithCustomToken → IDトークン取得

6 アプリ → setActiveSessionLambda（DynamoDB ActiveSessionsにセッション登録）
1デバイス1セッション制御。別デバイスからのログインを自動検知・強制ログアウト。

7 以降すべてのGraphQLリクエストはappsyncAuthで認証（60秒グレースウィンドウ）
Firebase IDトークン検証 + DynamoDBセッション一致確認。不一致は即時拒否。

3 テスト概要

層 1 — ユニット・統合テスト（Jest）

ツール **Jest（Meta、オープンソース、無償）** 規模 **62スイート / 1,247テスト** 実行時間 **〜3秒**

- ✓ 残業計算（日次・月次・年次の集計、端数処理）
- ✓ トークン処理（BTP / Firebase認証トークンの検証・更新・有効期限）
- ✓ セッション管理（打刻セッションの開始・終了・日またぎ補正）
- ✓ データフォーマット（時刻変換、ServiceNowフィールドマッピング）
- ✓ GraphQLクエリ（AppSyncクエリ・ミューテーション構造の検証）
- ✓ コンテキスト配線（全14プロバイダーの正しい接続を保証）

層 2 — E2Eデバイステスト（Maestro）

ツール **Maestro（オープンソース、無償）** 規模 **18フロー** 実行環境 **iOSシミュレーター**

- ✓ SSOログイン → 待機画面
- ✓ 出勤打刻 → メイン画面
- ✓ タスク開始（受信トレイ → スワイプ）
- ✓ タスク切替・中断・再開
- ✓ 退勤打刻

✓ 残業カウンター表示

✓ アプリ強制終了 → 再起動（データ永続性確認）

✓ その他11フロー（お気に入り、学習、コラボレーション 等）

層 3 — 大規模負荷テスト（k6）

ツール **k6（Grafana、オープンソース、無償）** 規模 **1,200ユーザー同時接続** テスト時間 **25分間**

✓ パート1: Lambda認証レイヤー（✅ 完了 — p95 103ms）

✓ パート2: DynamoDB 読み込み（✅ 完了 — p95 280ms）

✓ パート3: DynamoDB 書き込み（✅ 完了 — p95 524ms）

✓ パート4: ServiceNow・Salesforce連携（✅ 完了 — p95 276ms）

4 負荷テスト状況

パート 1 — Lambda認証レイヤー

✅ 完了 2026-04-14

指標	結果	目標値
レスポンスタイム（p95）	103ms	3,000ms以下
Lambdaスロットル	0件	0件
ピーク同時接続数	1,200 / 1,200人	1,200人
総リクエスト数	214,014件	—

パート 2 — DynamoDB 読み込み

✅ 完了 2026-04-15

指標	結果	目標値
残業サマリー取得（p95）	280ms	3,000ms以下
エラー率	0.00%	1%未満
Lambdaスロットル	0件	0件

パート 3 — DynamoDB 書き込み

✅ 完了 2026-04-15

指標	結果	目標値
打刻書き込み（p95）	524ms	3,000ms以下
エラー率	0.00%	1%未満

Lambdaスロットル	0件	0件

パート 4 — ServiceNow・Salesforce連携

✅ 完了 2026-04-15

指標	結果	目標値
タスク取得（p95）	276ms	3,000ms以下
エラー率	0.00%	1%未満
Lambdaスロットル	0件	0件

テスト実施: 2026-04-15（ユーザー利用なし）。総リクエスト数: 330,568件、テスト時間: 25分間。全4パート合格。

5 Ad Hoc デバイステスト

ビルド・配布プロセス

アーカイブ方法	Xcode 手動アーカイブ（方法B）
署名	Ad Hoc プロビジョニングプロファイル
出力	IPAファイルをエクスポート
配布	テストデバイスに直接インストール

プロビジョニング情報

プロファイル名	WTR-Internal-Adhoc-Testing
Bundle ID	com.benben0505.Project01App20250531
有効期限	2026年10月27日
登録デバイス数	26台（上限100台 / 年）

OTA アップデートチャンネル

internal **内部開発テスト用**
開発チームのみ。最新ビルドを随時配信。

preview **クライアントテスト用**
Ad Hoc IPAインストール済みデバイスへ配信。

OTA更新はEAS Update経由で配信。アプリ起動時に自動チェック・ダウンロード。ネイティブコード変更がない限りストア再申請不要。

将来計画

配布方式	Apple Business Manager + カスタムアプリ配信
メリット	UDID登録制限なし（本番展開対応）
対象	全社員への展開

6 外部システム連携設定 — ServiceNow / Salesforce / SAP BTP

アプリは3つの外部エンタープライズシステムと連携しています。タブで各システムの認証設定・API・Lambda 構成・トラブル対処を確認できます。

ServiceNow | Salesforce | SAP BTP / IAS

役割 タスク一覧の取得・状態更新・作業時間（metric_instance）の管理。Flow Designer 経由でテーブルを更新。

認証 — OAuth 2.0 Client Credentials

項目	値 / 説明
OAuth アプリ名	Mobile App Flow API（ServiceNow Application Registry に登録）
Grant Type	`client_credentials`
Token エンドポイント	`POST {SN_URL}/oauth_token.do`
トークン有効期限	60分（Lambda 内 60秒バッファ付きキャッシュ）
Lambda 環境変数	`SN_URL` / `SN_CLIENT_ID` / `SN_CLIENT_SECRET`（全5 Lambda に設定）

使用 Lambda・テーブル・エンドポイント

Lambda	テーブル / エンドポイント	操作
ServiceNowLambda	`u_work_task`	担当タスク一覧取得（最大1,000件）
ServiceNowUpdateStatusLambda	Flow: `workstateupdate_ejecmobileapp`	タスク状態更新（state フィールド書き込み）
ServiceNowMetricInstanceLambda	`metric_instance`	作業時間レコード取得（直近3日間）
ServiceNowUpdateMetricTimeLambda	Flow: `mobile_edit_workflow_time`	作業開始・終了時刻の更新
PMLambda	`u_work_task`	プロジェクト管理者向けタスク取得

Flow Designer — 2フロー

フロー名	トリガー	処理	重要な注意点
`workstateupdate_ejecmobileapp`	REST PATCH（非同期）	入力: `record_sys_id`, `state` → `u_work_task.state` を更新	⚠ スクリプトステップは必ず **Global スコープ** で実行。EJEC Mobile App スコープのみでは `u_work_task` の ACL に阻まれ `.update()` が `null` を返す
`mobile_edit_workflow_time`	REST PATCH（非同期）	入力: `metric_instance_sys_id`, `new_start`, `new_end` → duration を自動計算して更新	

トラブルシューティング

症状	原因	対処
タスク取得が空 / 401	OAuth トークン失効 or 環境変数未設定	Lambda 環境変数を確認。CloudWatch で `[ServiceNowLambda]` ログ参照
状態更新が反映されない（エラーなし）	Flow のスクリプトが EJEC スコープで動作し ACL にブロックされている	Flow Designer のスクリプトステップを Global スコープに変更
Flow の入力値が空	トリガーボディの変数がフローにワイヤリングされていない	Input Variable に `inputs.record_sys_id` 等を正確にマッピング
OAuth App の User フィールドが空	ServiceNow UI では非表示のフィールド	REST API で `PATCH /api/now/table/oauth_entity/{sys_id}` に `{"user":"<user_sys_id>"}` を送信

7 監視体制

本アプリの稼働状況は 3 つの手段で継続的に監視しています。それぞれ対象読者と目的が異なります。

① 業務活動ダッシュボード

対象読者	EJEC 管理職・人事担当
URL	activity-dashboard-1ef.pages.dev
更新頻度	60 秒ごとに自動リロード
目的	現在のチーム稼働状況を

② 日次 SLACK レポート

対象読者	EJEC IT 担当者・マリンスフィア
配信先	Slack `#mobile-daily-monitor`
配信時刻	毎日 08:00 JST（自動）
目的	前日の利用状況と Lambda エラーを朝一

リアルタイムで把握

主な表示内容:

- 現在アクティブな作業員一覧（名前・現在タスク・経過時間）
- 日次アクティビティヒートマップ（過去30日）
- CSV エクスポート（出勤記録・欠勤レポート）
- タスク見積もり精度レポート

番に確認

レポート内容:

- アクティブユーザー数 / 全員数（例: 28 / 35 名）
- 前日の活動記録件数・合計作業時間
- 前日アプリ未使用ユーザー一覧
- 12 重要 Lambda の 24 時間エラー件数（0 件なら ✅ 表示）

③ CLOUDWATCH アラーム（即時通知）

対象読者	マリンスフィア 開発担当
通知先	Slack `#mobile-lambda-alert` + メール
発火条件	5分以内に Lambda エラー ≥ 3 件（認証系は ≥ 1 件）
目的	障害を発生後5分以内に検知し、即座に対応開始

監視対象 Lambda（12 関数）:

- 認証系: appsyncAuth / BTPSSOLambda / SSOBTPIASLambda / ServiceNowSSOLoginLambda 他
- 打刻系: SSORefreshToken / BTPTokenRefresherLambda / OTupdaterLambda
- タスク系: ServiceNowLambda / ServiceNowUpdateStatusLambda / dailyActivityLogSaver

使い分けまとめ

📊 **活動ダッシュボード** — EJEC 管理職が「今日誰が稼働中か」を確認する日常ツール

📨 **日次 Slack レポート** — EJEC IT 担当者が毎朝前日の利用率とシステム健全性を確認

🚨 **CloudWatch アラーム** — 障害発生時にマリンスフィアへ即時通知。EJEC IT から連絡が来る前にマリンスフィアが把握

1 フロントエンドアーキテクチャ — 4層構造

React Native（Expo Router）のレイヤー構成。画面層からバックエンドへの依存は一方向。各層は Context API 経由で疎結合に接続。

画面層 Expo Router 画面 — app/ ディレクトリ

- grey.tsx 打刻前・時刻表示
- main.tsx アクティブタスク
- inbox.tsx タスク受信トレイ
- newClock.tsx 打刻確認
- learning.tsx 学習アクティビティ
- Login.tsx SSO認証
- activityLog.tsx 日別ログ確認

↓ useContext() で読み書き（単方向データフロー）

Context層 14 グローバルプロバイダー — src/components/

- GlobalTimeContext 打刻状態・BTPステータス
- TaskContext タスク一覧・活動記録
- OvertimeContext 残業時間計算・表示
- UserProfileContext 表示名・個人設定
- ToastContext 通知トースト
- ErrorContext エラーモーダル制御
- SessionContainer セッション監視
- + 7 その他 Auth / Favorites / Notification 等

↓ サービス関数・カスタムフックを呼び出し

Service / Hook層 ビジネスロジック — src/services/ + src/hooks/

- punchHandler 打刻ロジック・レート制限
- clockOutService 退勤処理オーケストレーション
- dailyActivityLogService ログ生成・保存
- btpSsoAuth BTP SSO認証・再認証
- syncService ServiceNow状態同期
- breakService 休憩管理
- useModalStates モーダル開閉制御
- useSwipeModal スワイプ操作
- useBreakTask 休憩タスク判定

↓ AppSync GraphQL mutation / query（主要永続化経路）

永続化層 クラウド永続化 + セッションキャッシュ

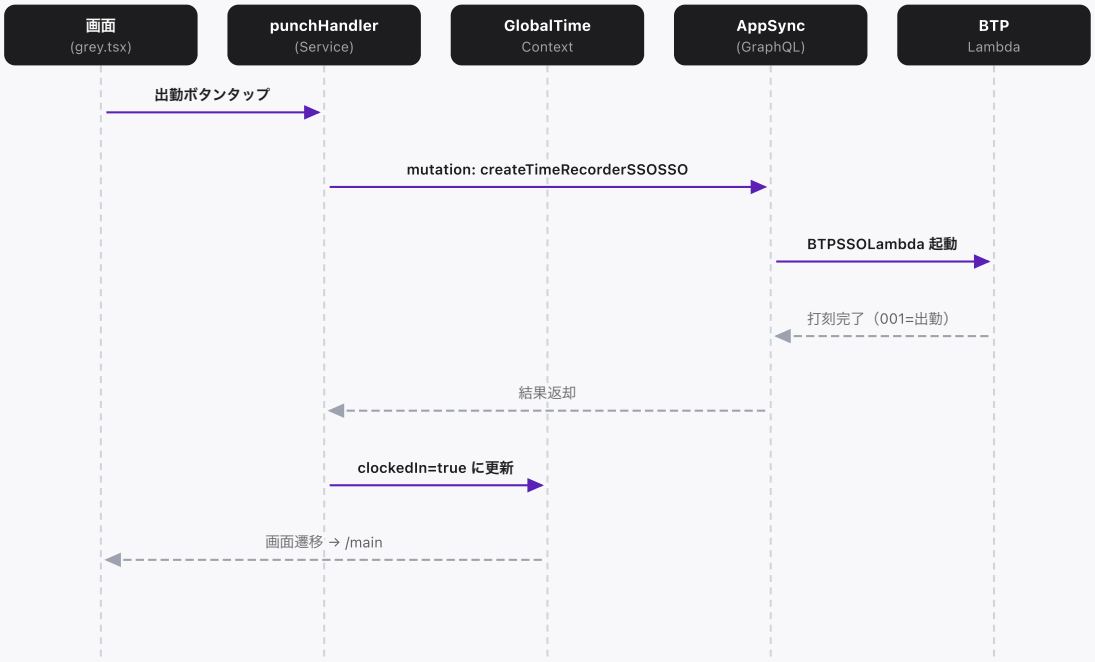
- AppSync GraphQL クラウド永続化の主経路
- DynamoDB 5テーブル 全重要データの保管先
- AsyncStorage セッションキャッシュのみ（ログアウト時クリア）

↕ バックエンドへ接続 — AppSync → Lambda → SAP BTP / ServiceNow 「バックエンド」タブで詳細を確認

2 主要機能フロー — フロントエンド視点

各機能で画面・Context・Service・バックエンドがどう連携するかをシーケンス図で表示。タブで切り替えてください。

① 出退勤打刻 ② タスク開始 ③ タスク切替 ④ 残業トラッキング ⑤ 日別ログ保存 ⑥ SSOログイン



3 データ永続化アーキテクチャ

全ての重要データは **AppSync GraphQL → Lambda → DynamoDB** 経路で直接クラウドに保存されます。AsyncStorage はセッション中のUIキャッシュ用途に限定されており、ログアウト時にクリアされます。

主要データフロー（AppSync 直接経路）

操作	AppSync Mutation / Query	Lambda	保存先
出退勤打刻	`createTimeRecorderSSO`	BTPSSOLambda	SAP BTP
退勤時ログ保存	`saveDailyActivityLog`	dailyActivityLogSaver	DynamoDB: dailyActivityLog
残業時間送信	`updateOvertime`	OTupdaterLambda	DynamoDB: OvertimeSummary
タスク状態更新	`updateWorkTaskStatus`	ServiceNowUpdateStatus	ServiceNow
活動ログ記録	`putActivityLog`	activityLog Lambda	DynamoDB: ActivityLog (TTL 30日)
個人設定保存	`saveUserProfile`	PersonalConfigDB Lambda	DynamoDB: PersonalConfigDB

AsyncStorage — セッションキャッシュのみ

キー	用途	スコープ
`@EmailStatusStorage`	受信トレイタスク一覧（セッション中の表示キャッシュ）	ログアウト時クリア
`@TimeRecordsStorage`	アクティビティ時刻記録（退勤時にDynamoDBへ送信）	退勤後クリア
`inbox_breakRecords`	休憩セグメント（退勤ログに統合して送信）	退勤後クリア
`inbox_breakTaskId / _breakStartTime`	現在の休憩タスクID・開始時刻	休憩終了時クリア
`ACTIVE_SESSION_ID`	デバイスセッション識別子（AppSync認証に使用）	ログアウト時クリア
`lastBtpUserEmail`	前回ログインユーザー（SSO経路選択に使用）	デバイス保持
`@TaskFavorites`	お気に入りタスク一覧（PersonalConfigDB Lambdaとも同期）	ユーザースコープ

退勤時のデータ確定フロー

1. 退勤ボタンタップ → BTPSSOLambda 経由で SAP BTP に退勤打刻（002）を直接送信。
2. セッション中の timeRecords + breakRecords を取得し、DailyActivityLog ペイロードを生成。
   dailyActivityLogService がローカルキャッシュを読み込んでログを構築。

3 AppSync `saveDailyActivityLog` mutation → dailyActivityLogSaver Lambda → DynamoDB に永続保存。

4 ローカルキャッシュをクリア、AsyncStorage を初期化してログイン前状態に戻す。

4 アプリ配布・OTA アップデート

iOS Ad Hoc 配布と EAS OTA（Over The Air）アップデートにより、ストア審査なしで迅速な機能リリースを実現。

現在の配布形態

プロビジョニング	Ad Hoc（テスト配布）
プロファイル名	WTR-Internal-Adhoc-Testing
有効期限	2026年10月27日
登録デバイス数	26台（上限100台 / 年）
OTAチャンネル	preview（クライアント向け）

OTA 更新の仕組み

配信プラットフォーム	EAS Update（Expo）
チェックタイミング	アプリ起動時に自動チェック
適用タイミング	次回起動時（バックグラウンドDL）
適用条件	JS・アセット変更のみ（ネイティブ変更は再ビルド必要）

将来の配布計画

配布方式	Apple Business Manager
メリット	UDID登録制限なし（本番展開対応）
対象	全社員への展開

テスト概要（詳細はバックエンドタブ 3 参照）　Jest ユニット / 統合テスト: **1,247件（62スイート）** — サービス・Context・ユーティリティをカバー。　Maestro E2E テスト: **18フロー** — 出退勤打刻・タスク操作・SSO 認証等の主要フローを実機シミュレーターで検証。