Progress




Artificial Intelligence | Progress® OpenEdge® Platform | You

# The Future of Progress OpenEdge Development

## It's not alphabet soup anymore—it's a development stack…

**How I built a complete REST API with full tests using a free OpenEdge license and a $20 AI subscription!**

# How Development Has Changed

## 2010 Development

- Research approach
- Write code
- Compile
- Debug
- Write more code
- Debug more
- Write tests (maybe)
- Fix bugs tests found
- Document (maybe)
- Deploy
- Fix deployments bugs

## 2026 Development

- Describe what you want
- Review AI output
- Steer when needed
- Run test (AI wrote them)
- Ship

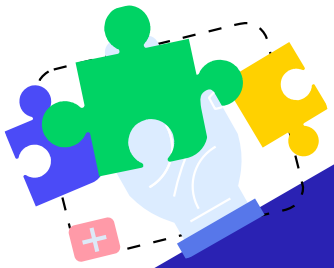
**Same quality. Fraction of the time.**


© 2026 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

# The Challenge

Build a modern REST API on PAS for OpenEdge for the Sports2020 database (24 tablets)

- Full CRUD operations(Create, Read, Update, Delete)
- Pagination, filtering, sorting, field selection
- Standard JSON response envelopes and error formats
- Comprehensive unit tests (ABL Unit)
- OpenAPI documentation and Postman test suite
- PAS for OpenEdge deployment with build automation



**I built it twice—the difference tells the whole story.**

Progress

© 2026 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

# The $20 AI Developer

What one developer can do with the right tools.

## The Stack

| | |
|---|---|
| **You**<br>The developer with domain expertise | **$0** |
| **Claude Code**<br>AI coding assistant (Anthropic) | **$20/mo** |
| **OpenEdge Education License**<br>Full OpenEdge development environment | **Free** |
| **OpenEdge AI Assistant**<br>AI integration with OpenEdge Platform | **Included*** |

Progress

© 2026 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

# What Is Claude Code?

Your AI pair programmer in the terminal: https://claude.com

### Reads Your Codebase

Understands project structure

Reads files and dependencies

Follows your CLAUDE.md instructions

### Write & Edits Code

Create new hire

Edits existing code

Follows your patterns

### Runs Commands

Compiles your project

Runs unit tests

Manages build tools

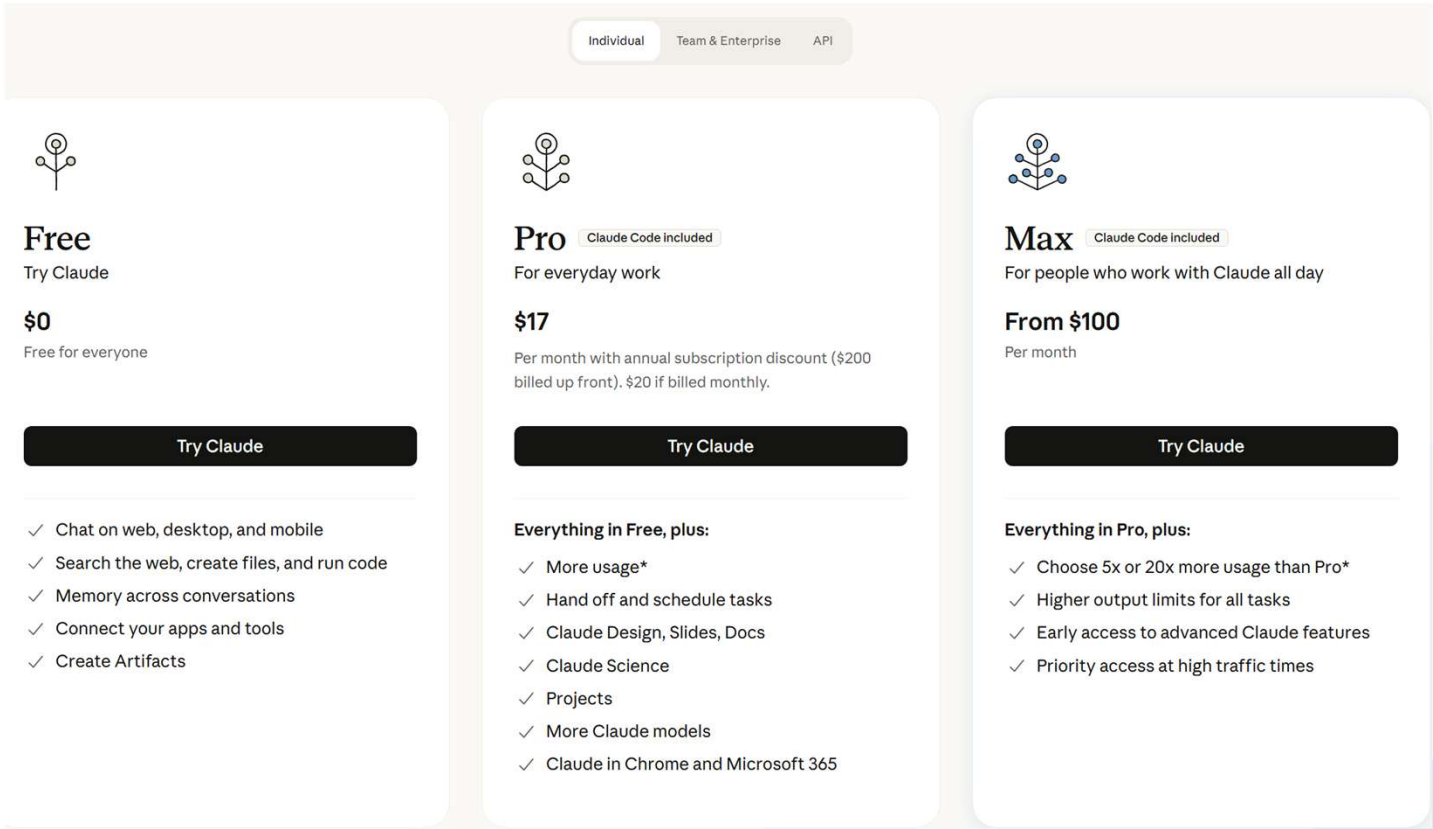
### CLAUDE.md = Your AI's Brain

A project-level instruction file Claude reads at the start of every session. Include architecture decisions, naming conventions, API signature, build commands and known pitfall. Every minute spent on CLAUDE.md saves hours of correction later.

Progress

© 2026 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

# claude.com/pricing




© 2026 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

# Project Walkthrough

- Vs Code plugin
  - openedge.project.json
  - .vscode folder
- CLAUDE.MD - /init
- .claude folder
- /docs, /src, /target
- Build.xml
- Terminal: Claude
- Github repo




© 2026 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

# Additional reference context found

- 1st project:
  - OpenEdge Application Design
    https://docs.progress.com/bundle/openedge-modernize-guide-122/page/OpenEdge-application-design.html
  - API First Development
    https://docs.progress.com/bundle/openedge-abl-develop-services-122/page/API-first-development.html
  - Sports 2020 Database info
    https://docs.progress.com/bundle/openedge-database-introduction/page/Sports-2020-database.html
  - ABLUnit documentation
    https://community.progress.com/s/article/ABLUnit-testing-with-the-ABLUnit-testing-framework
- 2nd project:
  - Ant PCT Guide
    https://wiki.rssw.eu/pct/Home

Progress
© 2026 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

# Project 1: The Learning Curve

Sports2020: My first AI coding project

1

## Day 1 — Planning

Gave Claude 3 reference docs. Got: CLAUDE.md, project plan, ERD. Planning is an AI sweet spot — this went smoothly.

2

## Day 2 — First Major Pivot

Claude built DAOs returning JSON directly through all layers.
I intervened: "Shouldn't we use temp-tables and ProDataSets?"
Claude immediately pivoted to the OpenEdgeA pattern.
Also corrected: "Use built-in ROW-STATE, not manual fields."

3

## Days 3–5 — Build, Test, Debug

Reserved keyword errors, 6 critical ABLUnit bugs, primary index detection. Built validation framework, service layer. Reached 64 tests passing.

## Key Insight

AI follows your lead. It will build whatever seems reasonable—but the developer has domain knowledge.

When I suggested temp-tables, Claude understood the "why" and pivoted immediately.

The human provides the "what" and "why." AI provides the "how."

Early correction = minimal rework
Late correction = days of rework


© 2026 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

# Project 1: Steering Moments

## Redirecting Claude

### 1. Architecture: JSON vs Temp-Tables

*Claude's default:* JSON through all layers
*My fix:* "Use temp-tables with TRACKING-CHANGES"
*Impact:* Prevented weeks of rework

### 2. Row-State Tracking

*Claude:* Added manual RowState field
*My fix:* "Use built-in ROW-STATE attribute"
*Impact:* Simpler, native pattern

### 3. Reserved Keywords

*Claude:* Used "debug" as method name—compile error
*My fix:* Pointed to ABL keyword reference
*Impact:* Added keyword list to CLAUDE.md

### 4. Primary Index Detection

*Claude:* Couldn't find primary indexes in metaschema
*My fix:* Linked Progress Community article
*Impact:* Fixed with _file._prime-index

### 5. WebHandler API Mismatch

*Claude: P*laceholder methods for wrong API version
*My fix:* Research OpenEdge 12.8 WebHandler API
*Impact:* Corrected method signatures

### 6. PCT Build System

*Claude:* Ant couldn't load PCT tasks
*My fix:* Pointed to RSSW installation docs
*Impact:* Copied PCT.jar to Ant lib


© 2026 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

# Project 1: The Testing Wall

6 critical ABLUnit discoveries

1. **NUM-RESULTS invalid on TEMP-TABLE**
   Must use QUERY handle, not TEMP-TABLE

2. **TRACKING-CHANGES only on TEMP-TABLE**
   Cannot set on DATASET — compile error

3. **Can't EMPTY while tracking enabled**
   Must disable tracking before EMPTY

4. **DAO must control tracking lifecycle**
   Read operations need explicit control

5. **OUTPUT TABLE not TABLE-HANDLE**
   Syntax matters for temp-table params

6. **DELETE removes from temp-table**
   Need BEFORE-TABLE for deleted records

**Every bug was documented. Every lesson was fed into Project 2.**

**Result:** 19KB testing guide (ablunit_testing_lessons_learned.md). Claude never made these mistakes again.


© 2026 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

# Project 1: PAS for OpenEdge Deployment & Summary

Days 6–9 were the biggest time sink

### 4 Days on Deployment

- ✗ Spring Security bean initialization errors
- ✗ Handler method signatures wrong for OpenEdge 12.8
- ✗ PCT build system classloader issues
- ✗ Full HTTP round-trip not validated end-to-end

### Project 1 Result

- 9 days → 1 table complete (vertical slice)
- 64 unit tests passing
- Architecture proven end-to-end
- Every pitfall documented for reuse

### Project 1 wasn't a failure—it was an investment.

The vertical slice proved the architecture. Every lesson was captured in CLAUDE.md and skills documents. Think of it as training both me AND the AI for round two.


© 2026 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

# Project 2: Armed with Lessons

sports_dynamic: The second attempt

## What Changed Between Projects

1

### Comprehensive CLAUDE.md from Day 1

All architecture decisions, reserved keywords, API signatures, and pitfalls—documented before a single line of code

2

### Created 7 reusable "Skills"

lessons-learned, dynamic-abl, webhandler, unit-test, Progress® Application Server (PAS) for OpenEdge, build, openapi—like training manuals for the AI

3

### Dynamic architecture: one handler for ALL tables

No per-entity classes, dynamic buffers handle any table at runtime via schema introspection

4

### Build automation from the start

Ant/PCT compile, database init, PAS for OpenEdge start/stop—no manual steps, no deployment surprises


© 2026 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

# Project 2: Three Days to Complete

## Day 1: Foundation

Feb 1

- Complete project structure
- 8 Claude Code skills
- 40+ ABL reference files
- Build system configured
- CLAUDE.md with full architecture

**162 files of preparation**

## Day 2: Implementation

Feb 2

- 8 production classes
- GenericRepository for ALL tables
- CrudHandler serves 24 tables
- QueryBuilder with filter/sort
- 130 unit tests — ALL passing

**Zero to working API in ONE DAY**

## Day 3: Polish & Deploy

Feb 3

- Postman test collection
- Build automation (start/stop)
- OpenAPI 3.0 specification
- Session history documented
- Final deployment to PAS for OpenEdge

**Ready for stakeholder review**

Progress

© 2026 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

# Why Dynamic Architecture Won

## One handler, all tables

### Project 1: Per-Entity Pattern

For EACH of 24 tables:

```
StateDAO.cls          (data access)
StateManager.cls      (business logic)
StateService.cls      (service layer)
StateHandler.cls       HTTP handler)
StateTempTable.i       temp-table def)
```

**= 5 files x 24 tables = 120 files needed**

### Project 2: Generic Pattern

For ALL 24 tables:

```
GenericRepository.cls   (dynamic CRUD)
CrudService.cls         (generic service)
CrudHandler.cls         (generic handler)
TableMetadata.cls       (schema inspect)
QueryBuilder.cls        (query building)
JsonSerializer.cls      (serialization)
```

**= 6 files total for ALL tables**

120 files vs. 6 files—same functionality, 20x less code to write, test and maintain


© 2026 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

# The Numbers

Side-by-side comparison

| Metric | Project 1 (sports2020) | Project 2 (sports_dynamic) |
|---|---|---|
| **Duration** | ~9 working days | **3 days** |
| **Tables Implemented** | 1 of 24 | **24 of 24** |
| **Unit Tests** | 64 passing | **130 passing** |
| **Production Classes** | 8+ (per entity) | **8 (generic)** |
| **Architecture Pivots** | 3 major | **0** |
| **Steering Moments** | 6+ | **2–3 minor** |
| **Deployment Time** | ~4 days | **Hours** |
| **PAS for OpenEdge Working** | Partial | **Yes** |
| **API Documentation** | Partial | **Full OpenAPI** |
| **Build Automation** | Basic | **Complete** |

Progress

© 2026 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

# Prompt Patterns That Worked

## How to talk to your AI pair programmer

1

### Provide Reference Material First

*"Here are 3 docs on our architecture, API standards and DB schema. Design a REST API following these."*

*“Add the list of progress keywords to the cluade.md file..."*

2

### Ask for validation and recommendations

*""What information do you need from me for you to implement phase 1?“*

*"I think we may want to change the project plan to finish the entire State project start to finish... thoughts?“*

*"How many classes have we created? Is that too much bloat?*

3

### Point to Specific Resources

*"Go read this Progress Community article for how to find the primary index of a table."*

*"I think we need to build ant/pct scripts to compile, build the abl app, and deploy."*

4

### Ask for Verification

*"Verify status after every command to see if each worked. 'Test them all starting with pasoe-stop.'"*

5

### Let AI Document Lessons

*"Pull a history of all prompts and put in the docs folder." AI writes surprisingly good documentation.*

Progress

© 2026 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

# Honest Assessment

What AI does well—and what it doesn't

## AI excels at:

- ✓ Generating boilerplate and repetitive code
- ✓ Following established patterns consistently
- ✓ Writing comprehensive unit tests
- ✓ Creating documentation (API specs, README)
- ✓ Scaffolding project structures
- ✓ Implementing standard REST/CRUD patterns
- ✓ Refactoring and code cleanup

## AI needs human help with:

- ⚠ Architecture decisions (builds whatever you ask)
- ⚠ Domain-specific knowledge (ABL quirks)
- ⚠ Environmental / deployment issues
- ⚠ Validating that commands actually worked
- ⚠ Knowing when to stop vs. keep going
- ⚠ Niche language patterns not in training data

***Sweet Spot:*** Human provides architecture + domain knowledge + validation.
AI provides implementation + testing + documentation at speed.


© 2026 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

# Lessons Learned

What I'd tell someone starting today

1

### Start with a plan, not code

Let AI generate the project plan first.
Review before implementation.
This is where YOUR expertise matters most.

2

### Invest heavily in CLAUDE.md

Document decisions, conventions, gotchas.
Include correct API signatures.
Highest-leverage investment you can make.

3

### Build a vertical slice first

One entity, all layers, fully tested.
Validate architecture before scaling.
Discover issues early. *(Project 1)*

4

### Then go generic

Use dynamic patterns once architecture is proven.
One class serving N entities beats N classes.
*(Project 2)*


© 2026 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

# Lessons Learned (continued)

### 5 Correct the AI early and specifically

"Use temp-tables" beats letting it build 3 layers with JSON.
Point it to specific docs.
Earlier = less rework.

### 6 Document everything

Session history captures what worked.
Lessons-learned docs prevent repeated mistakes.
AI writes great docs.

### 7 Provide extra context for niche technologies

ABL has limited training data.
Provide reference files, code examples, API docs.
Skills bridge the gap.

### 8 Verify, don't trust

Always run tests after AI generates code.
Check that commands succeeded.
AI can be confidently wrong.


© 2026 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

# Getting Started

## Your first AI coding session

1. **Get the tools**
   Claude Code ($20/mo) + OpenEdge Education License (free) + OpenEdge AI Assistant

2. **Create CLAUDE.md**
   Project purpose, architecture, conventions, known pitfalls, build commands

3. **Start with planning**
   "Design a project plan for [your goal]", review before implementing

4. **Build incrementally**
   One component at a time, run tests after each, correct AI early

5. **Document as you go**
   Update CLAUDE.md with lessons, create skills for domain patterns


© 2026 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

# AI Prompts

Tasks you can ask AI to do

1. **Build/Maintain ABLUnit tests**
2. **Document: plans, projects, programs, end users, technical specs or Executive Summary PPTs**
3. **Write OpenAPI specs and Postman collections**
4. **Code Review, search codebase for security flaws, performance improvements, potential memory leaks, potential areas that cause a reported defect (“needle in a haystack”)**
5. **Build automated CI/CD processes**

Progress

© 2026 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

# AI-Native Software Engineering Evolution

From copilots to autonomous software factories

| | 2026 – 2027 | 2027 – 2029 | 2029 – 2032 | 2032+ |
|---|---|---|---|---|
| | **AI COPILOT**<br>AI assists humans | **AI TEAMMATE**<br>AI collaborates with humans | **AI ENGINEERING TEAMS**<br>AI agents orchestrate delivery | **AUTONOMOUS SOFTWARE FACTORIES**<br>AI runs the SDLC end-to-end |
| | **Assist** | **Collaborate** | **Orchestrate** | **Optimize** |
| HUMAN ROLE | Developer | AI Supervisor | AI Orchestrator | Business Strategist |
| PRIMARY AI ROLE | Suggestions and assistance | Task completion with human in the loop | Multi-agent delivery of complex solutions | End-to-end execution and continuous optimization |
| PRIMARY FOCUS | Productivity and developer efficiency | Engineering speed and quality | System optimization and scale | Business outcomes and value creation |
| TYPICAL DELIVERABLES | • Code completion<br>• Unit tests<br>• Documentation<br>• Code reviews | • Entire features<br>• Test suites<br>• Pull requests<br>• Security fixes | • Complete epics<br>• Architecture<br>• Deployments<br>• Operations | • Entire products<br>• Self-healing systems<br>• Continuous evolution<br>• Autonomous maintenance |
| HUMAN OVERSIGHT | **Very High**<br>Humans approve and validate everything | **High**<br>Humans review and guide AI-generated outputs | **Moderate**<br>Humans orchestrate and oversee outcomes | **Strategic Only**<br>Humans set direction, guardrails and governance |

INCREASING AI AUTONOMY

DECREASING MANUAL ENGINEERING EFFORT

**The future belongs to engineers who architect AI-powered solutions, not those who just write code.**

Progress

© 2026 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

# AI will not replace OpenEdge developers—

## but developers using AI will replace those who don't.




## The Future of OpenEdge Development =

✓ $20/month AI subscription + FREE OpenEdge license = Production-quality REST APIs

✓ 9 days to learn → 3 days to deliver → Every project gets faster

✓ Your domain expertise is the multiplier → AI amplifies what you know

*Try it. Document everything. Get faster every time.*


© 2026 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

# News You Can Use

- Learn how to access the **OpenEdge AI Assistant**
- Get started with the **OpenEdge MCP Server**
- Book a demo of **Progress Agentic RAG**
- Modernize to **OpenEdge 12.8** today
- Explore what's new in **OpenEdge 13.0**
- Work with **Progress Professional Services** to accelerate adoption
- Learn about **security capabilities** across OpenEdge
- Explore the **developer tools** available for OpenEdge today
- Connect with us: **Discord, Community, CVP and more**



Progress

© 2026 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.


Progress®