This is an exciting startup concept that addresses a real need in the market! Creating a platform for non-technical users to build custom AI agents could be quite valuable. Let me walk through some thoughts on your tech stack choices and suggest some considerations for building this product effectively.

Tech Stack Analysis

Your proposed tech stack has solid foundations:

1. **Frontend/Website**: Laravel + Vue.js + Inertia.js
    - This gives you a robust MVC framework with Laravel
    - Vue.js provides reactive UI components
    - Inertia.js bridges them nicely without requiring a separate API layer
2. **Basic Agents**: Laravel Neuron AI package
    - Good for simpler AI integrations directly within your Laravel app
3. **Advanced Agents**: AgentForge + FastAPI
    - AgentForge provides agent orchestration frameworks
    - FastAPI creates Python microservices to connect with Laravel

Suggestions for Implementation

Architecture Considerations

Consider structuring your system in layers:

1. **User Interface Layer** (Laravel + Vue)
    - Visual agent builder with drag-and-drop components
    - Agent configuration forms and templates
    - Agent monitoring and analytics dashboards
2. **Middleware Layer** (Laravel)
    - Authentication and user management
    - Agent configuration storage
    - Simpler agent execution through Neuron AI
3. **Agent Execution Layer** (AgentForge + FastAPI)
    - Complex agent logic execution
    - LLM interactions
    - Tool/API integrations

User Experience Recommendations

For non-technical users to successfully create AI agents, focus on:

1. **Agent Templates**: Provide pre-built templates for common use cases (customer service, content creation, data analysis)

2. **Visual Builder**: Create an intuitive interface where users can:
    - Define agent goals and constraints
    - Configure agent personality and behavior
    - Connect to data sources and tools
    - Set up trigger conditions and workflows
3. **Testing Environment**: Let users test their agents in a sandbox before deployment

Technical Implementation Considerations

1. **API Management**
    - You'll need a robust system to handle integration with external APIs that agents might need to access
    - Consider an API gateway pattern to manage this complexity
2. **Agent Execution Engine**
    - Plan for asynchronous execution of longer-running agent tasks
    - Consider using Laravel's queue system or RabbitMQ for job management
3. **Scalability**
    - Containerize your Python services with Docker for easier scaling
    - Consider using Kubernetes for orchestration as you grow
4. **Security**
    - Implement proper isolation between different customers' agents
    - Create security policies to limit what agents can do

Specific Technical Components

1. **For the Laravel + Python Bridge**:
    - Your FastAPI approach is solid
    - Consider implementing a message queue (like Redis or RabbitMQ) between Laravel and FastAPI for better resilience
2. **For Agent Configuration Storage**:
    - Consider using a flexible schema like JSON fields in PostgreSQL
    - Or explore document databases like MongoDB for more complex agent configurations
3. **For the Visual Builder**:
    - Look into Vue component libraries like Vuetify or PrimeVue for UI components
    - Consider using a flow-based programming interface (like vue-flow) for visual agent creation
4. **For Monitoring and Analytics**:

    - Implement logging of all agent actions
    - Create visualization dashboards for agent performance
    - Set up alerting for problematic agent behaviors

Initial Implementation Roadmap

1. **MVP (Minimum Viable Product)**:
    - Start with a small set of agent templates
    - Focus on a clean, intuitive UI for configuration
    - Support basic integrations with common APIs
2. **Growth Stage**:
    - Add more sophisticated agent capabilities
    - Implement an agent marketplace
    - Enhance analytics and monitoring
3. **Maturity Stage**:
    - Support agent collaboration and workflows
    - Implement more advanced LLM capabilities
    - Create an ecosystem for third-party extensions

Would you like me to elaborate on any particular aspect of this guidance? Or would you like me to explore how you might implement a specific component of your platform in more detail?