STANISLAV KISELEVSKII

.NET SOFTWARE DEVELOPER

CONTACT INFO

Email: st.kiselevskii@gmail.com

Telegram: @Stanislav_Kiselevskii
Mobile Phone: +49 176 75 494 593

Preferred communication methods: Telegram, Email

Accessibility:	On-site, Remote, Hybrid.
Current location:	Ludwigshafen - Rheinland-Pfalz.
Start New Job Time:	Since May 2026, after current contract completion.
Work Permit:	Germany (Blue Card)

Linked In profile:	https://www.linkedin.com/in/st-kiselevskii/
GitHub profile:	https://github.com/K-S-K/CV/

The Platforms, Technologies, I'm experienced in:	Windows, Linux, Docker, Dotnet, .Net Framework, C#, .Net 10, ASP.Net, REST API, WEB Services, EF Core, Dapper WinForms, WPF, Windows Services MS SQL Server – T-SQL – DDL, DML, Performance Optimization

Industry domains, I'm experienced in:	Aerospace – the development of the physics-based Software Digital Twin of Gaia telescope Attitude and Orbit Control System. Electric power – software development for billing, power network modeling. Fintech – software development for trading automation for fiat and crypto. Telecom – software development for billing, monitoring, data converting. Railway – Blackbox data analysis.

EDUCATION:

Ural State Technical University (Ekaterinburg, Russia),
Engineer's degree, Automated control of electric power systems, 1992-1998.

Languages: English(my current working language), Russian (native), und Ich lerne Deutz.

JOB EXPERIENCE (MAIN):

05.2024 – 12.2025 **ASTRONOMISCHES RECHEN-INSTITUT (HEIDELBERG UNIVERSITY)**

Job Purpose: Software digital twin development.

Responsibilities: Requirements analysis; proposing the project architecture; development of all elements of the product; bringing the ideas of scientists to the form of a tool which can be used by them for making computational experiments with modeled system.

Business project description: We created a physics-based software digital twin of Gaia telescope Attitude and Orbit Control System. It is a tool for testing and tuning spacecraft attitude control algorithms - the fast, configurable, production-ready framework to prove algorithms for the next-generation telescope.

Technical project description: The whole project is organized in following components, which can be used separately, and can be connected as an end-user tool:

- The set of “Library” assemblies with common mathematical abstractions and infrastructure tools.
- The set of “Modules” – assemblies with implementations of different spacecraft equipment parts.
- The “Laboratory” – special module which holds all different modules together and provides communication between them.
- UI Service, which allows user to configure experiment, see the progress, browse the results and download experiment data in CSV form.
- Rest API based service which holds the Laboratory module.
- Optional separately executable service for Star Catalog which, in some experiments, consume more memory than normally can be allocated on the workstation.

Languages, tools and technologies, used in the project: C#/.Net10, Rest API, Blazor, Docker, Linux, VSCode.

Achievements: During working in the position, I’ve done next duties:

- Translating various concepts and requirements into a format appropriate for software product development.
- Organizing the development flow for bringing models and prototypes to the usable instrument.
- Development the modules from the ideas were originally prototyped in Python and Java by my colleagues and adjusted after primary implementations.

The description of this project can be seen here:
https://github.com/K-S-K/CV/blob/main/Articles/36_GaiaSDT/Article.md

05.2022 – 11.2023 **EPAM SYSTEMS**

Job Purpose: Software products support for EPAM Customers.

Business project description: The software is designed for stock market assets processing.
A tremendous and well-featured desktop tool with a long history of evolvement

Technical project description: database, server, and desktop application (Windows Forms).

Responsibilities: Support, bug fixing, and implementing new features.

Languages, tools and technologies, used in the project: C#, T-SQL, MS Visual Studio

06.2020 – 04.2022 **SOFTMEDIALAB**

Software development for different customers.
There were several projects here.

Copy Trading.
Business project description: Providing the following trading by the follower traders after the leader trader on the Binance crypto exchange.

Technical project description: Listening to the Binance exchange signals for the "originator trader" account and repeating orders for the "duplicator trader" account with proportional order size correction regarding each duplicator trader wallet amount.

Responsibilities: collecting requirements from the customer, RND, architecture planning, implementation and deploying of MVP version, technical support, evolving of the product

Languages, tools and technologies, used in the project: .Net5, WPF, MS Visual Studio, Binance.Net library by JKorf, and some domain area knowledge.

The description of this project can be seen here:
https://github.com/K-S-K/CV/blob/main/Articles/27_CopyTrading/Article.md

Bank portal.
Project business description: Data exchange between the company and the bank.

Responsibilities:

- Added CSV files import to the database.
- Added XLSX files import to the database.
- Added report-creating functionality as Google Table.
- Added file downloading from the Google Drive functionality.

Used technologies: PostgreSQL, .Net Web Server, EF Core, Dapper, Google Drive, Google Documents.

03.2011 – 03.2016 **EASTWIND**

Software development for telecom billing systems.
It were three projects there.

EWReliability.
Project business description: Reliability analysis system for the cell operator billing system.
Practical application: To measure software's availability factor in production and the CI cycle.
Project purpose: The analysis of the program's complex reliability parameters and status.

Project technical description: Application health data collection and analysis. Availability factor measuring. Tool for investigation of the evolving of accidents. Software for load monitoring and emergencies prediction. Tool for loading testing of billing software after every sprint.

Initially, we had one point of interest: to measure the availability factor of our software (a cell operator billing system) to be sure it meets the requirements of our customer (cell operator). They asked us to provide an availability factor = 0.99995. We've researched methodology for measuring a time interval we can qualify as a "working interval" and a time interval we can call a "not working interval". The ratio of a calculated sum of the working interval to the whole observation time is the availability factor we need.

Project Roles: Software developer, Product Owner, Team leader.

Responsibilities:

- Researched the possibilities of availability factor measuring.
- Developed a technology of software monitoring health.
- Created the DLL for other software developers to emit the health data to the custom Performance counters.
- Created the software for collecting health data in the MS SQL Database.
- Formalized the requirements for dividing operation time into the periods, classified as "working," "hanging," and "failures" (this part was implemented by another software developer whom I shared part of my job with)
- Formalized the requirements for the graphical representation of the classified periods for all observing applications on the common time scale for the developer whom I shared this part of the job with.
- Developed the software tool to view these intervals in the form of a table and the graphical form for the emergencies evolving investigation.
- Developed the software for automatic periodical reporting.
- Developed the software for generating different traffic types for the load testing of the billing software.
- Organized and performed the load testing of the billing software after every sprint (we have not approved the release if its availability factor was significantly worse than the availability factor of the previous version).

Team management: 1 to 4 Software Developers

Evolving features of the project we implemented with wonderful people I worked with:

- Configurable deployment over the cluster subsystem.
- Many different traffic models for the load testing subsystem.
- Persistent queue of measures to survive the database server unavailability periods without keeping all measures in memory (sometimes it was extremely necessary).
- Several additional measuring subsystems like SQL Server availability observation component and MongoDB Health monitoring component, because SQL Server and MongoDB don't naturally provide data we need to observe them.
- Reliability Interval Marking on DB subsystem which is running on SQL Server by the SQL Server Agent and makes ready-to-use intervals of work for every observing software instance.
- Tree-like table interval representation for the UI and the reports.
- Reliability diagram - a scalable graphical representation of the reliability intervals on a common time scale for manual visual analysis of emergency situations evolution process.
- Dangerous situations detector on the database side - a module that finds time periods when applications are overloaded simultaneously (so-called "Pillars") to show them on the Reliability diagram and in the report tables.
- Emailing subsystem which hourly sends reports with reliability diagrams and dangerous situations tables.
- Online Diagram - the visualization of the real-time state of the observing applications.

Tools: Microsoft Visual Studio, Microsoft SQL Server Management Studio, SQL Server Profiler for bottlenecks research and for DB stuff optimizing, WinDBG for collecting the evidence of the tested software memory leaks and deadlocks.

Technologies: .Net Framework, Windows Forms, Windows Services, Application Domains, Desktop Applications, MS SQL Server (SP, UDF, Triggers, Jobs, Indexes, etc.).
Languages: C#, T-SQL (DDL, DML).

The description of the project can be seen here:
https://github.com/K-S-K/CV/blob/main/Articles/05_EWReliability/Article.md

ASN.1 Format driver generator.
Project business description: Tool for creating ASN.1 data files reading and writing drivers based on the notation.

Project technical description:

- ASNData assembly (DLL) that provides base types and reading and writing functionality.
- ASNProcessor assembly (DLL) that provides ASN notation parsing and creating data model source code in C#.
- ASNTools assembly (desktop WinForms application), which opens ASN.1 notation file, displays its structure, creates a .Net project with ASNData assembly and source code generated by ASNProcessor assembly, and calls MSBuild to create a new DLL assembly which is a format driver for the ASN.1 notation file processed.

Responsibilities: Created two assemblies of this project:

- ASNProcessor assembly (notation parsing, data model representation in the normalized form)
- ASNTools assembly (UI, displaying the data structure as a tree, generating format file driver assembly)

Languages, tools and technologies, used in the project: .Net, C#, MSBuild, MS Visual Studio, ASN.1.

EWMediation.
Project business description: Automatic data exchange between cell operators.

Project technical description: The data packet queue is processed by data converters and uploaders.

Project Role: Software developer

Responsibilities:

- Developed different format converters
- Implemented each format converter as DLL, which implements the interface required by user software
- Implemented tests for every format converter (as for mine converters as for those created before me)
- The formats were: text, XML, binary, and CSV.

Languages, tools and technologies, used in the project: .Net, C#, MS Visual Studio, Windows Services, Application domains.

JOB EXPERIENCE (OTHER):

2011 – 2018 **SMALL PRIVATE FINTECH STARTUP**

Project business description: Automatic trading, different trading approaches testing.

Project technical description: The project contains an exchange trading history database, exchange emulator, real exchange connector, main trading desktop program with UI, configurations manager, and several united trading algorithms implementations, which can be used with the main system.

Project Role: Software developer

Responsibilities:

- Requirements collection, normalization and formalization
- Created the software for the trading algorithm testing on the historical trading data (RnD)
- Created the software for trading on the exchange by the implemented trading algorithm (RnD)
- Created the database and the exchange emulator
- Created the Exchange connector, which can transparently restore context after restoring the connection after the connection was lost to external reasons
- Created the trading tool which can be connected to the exchange emulator as to exchange connector
- Implemented many trading algorithms
- Developed the configurations optimizer tool
- Developed the configuration changer based on a schedule and the market situation measurement

The final project contains these components:

- The SQL Server database with trading history data in ticks
- The Import tool for filling the database with trading history data from .qsh files
- The Exchange Emulator
- The Exchange Connector (connector to SmartCOM connector), which can transparently restore the context after the broken connection is restored
- The Trading Engine, which can be connected to the Exchange Connector or the Exchange Emulator
- The set of different Indicators which can be connected to the Trading Engine by configuration
- The scheduling module with switching the indicator configurations on the schedule base
- The Desktop Application with the UI that allows to monitor and configure the trading process
- The quick calculations tool that is used to emulate trade sessions quickly without visualization simultaneously for different configurations on the historical data to do experiments with algorithms and their configurations
- The reporting module
- The New UI module on WPF was started to be implemented, but it is rest as a concept only
- A lot of small tools for different experiments.

Languages, tools and technologies, used in the project: C#, T-SQL, WinForms, WPF, MS Visual Studio.

https://github.com/K-S-K/CV/blob/main/Articles/04_TDATrading/Article.md

2000 – 2007 **PROSOFT-SYSTEMS**

Developing electricity metering software and power billing software.

Project purpose: Electric power meters can provide data exchange by different hardware and software APIs. The server application collects data from the power meters and puts data into the SQL Server database. SQL Server Agent executes jobs that calculate result data from the primary data with measurement transformer factor and power grid topology history (bypass switches commutation history, measurement transformers replacement history e, etc.)

Project description: My application provides the possibility of proper data storing corresponding to the place on the power grid where measures were collected from. In other words, my application is the editor for creating the power network model in the database to store data in the right way to process data by jobs and retrieve data for the reporting. My application contains a T-SQL script that can actualize any version of the database to the actual version with all necessary data conversions, COM object which provides API and UI controls for the data structure editor, and for the different applications that work with data: viewers, report generators, etc., and desktop editor for the tree-shaped representation of the electric energy system.

My part in this project: Requirements preparation, integration with existing architecture, development of the architecture, implementing, testing, deployment to the first customer, technical support, collecting and implementing additional customer requirements after they got some user experience.

Implementation technologies: MFC SDI Application, ATL COM object for data management level, OLE DB, T-SQL, and some subject area knowledge.

Applied technologies: C++, MSVS, ATL COM object for data management level, MFC, MFC SDI Application, Windows API, OLEDB, CHM Help, SQL (MS SQL Server), and some subject area knowledge.
Developing database infrastructure and tools for editing the electric network counting scheme and solving rules calculated by MS SQL Agent. Database infrastructure to represent a hierarchical model of the part of the electric power system. User interface tools for the editing of the model (MFC). Different tools for the scheme data synchronization. The history of the scheme changes, which affects the summary power calculation (Current transformer replacement history, power meter replacement history, etc.)

https://github.com/K-S-K/CV/blob/main/Articles/03_ESphere/Article.md

1999 – 2001 **URAL STATE TECHNICAL UNIVERSITY TELEPHONE COMMUTATOR ADMINISTERING**

Billing software development. From saving CDR data to printing toll billing orders for customers.

Languages, tools and technologies, used in the project: C++, MSVS, ATL, MFC. No DB, file-based data storage.

1998 – 2000 **RESEARCH INSTITUTE OF THE AUTOMATION OF RAILWAY TRANSPORT. (REMOTELY)**

Development of the railway black box analyzer.

My part in this project was UI, graphics, and infrastructure.

Languages, tools and technologies, used in the project: C++, BC Builder, MSVS.

https://github.com/K-S-K/CV/blob/main/Articles/01_Railway_BB/Article.md